

libstdc++

Generated by Doxygen 1.7.1

Sun Oct 10 2010 21:13:30



# Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Module Documentation</b>	<b>7</b>
2.1	Extensions . . . . .	7
2.1.1	Detailed Description . . . . .	8
2.2	SGI . . . . .	8
2.2.1	Detailed Description . . . . .	11
2.2.2	Function Documentation . . . . .	12
2.2.2.1	__median . . . . .	12
2.2.2.2	__median . . . . .	13
2.2.2.3	_Find_first . . . . .	13
2.2.2.4	_Find_next . . . . .	13
2.2.2.5	_Unchecked_flip . . . . .	14
2.2.2.6	_Unchecked_reset . . . . .	14
2.2.2.7	_Unchecked_set . . . . .	14
2.2.2.8	_Unchecked_set . . . . .	14
2.2.2.9	_Unchecked_test . . . . .	15
2.2.2.10	compose1 . . . . .	15
2.2.2.11	compose2 . . . . .	15
2.2.2.12	constant0 . . . . .	15
2.2.2.13	constant1 . . . . .	15
2.2.2.14	constant2 . . . . .	15
2.2.2.15	copy_n . . . . .	16

---

2.2.2.16	distance	16
2.2.2.17	identity_element	17
2.2.2.18	identity_element	17
2.2.2.19	iota	17
2.2.2.20	is_heap	17
2.2.2.21	is_heap	17
2.2.2.22	is_sorted	18
2.2.2.23	is_sorted	18
2.2.2.24	lexicographical_compare_3way	18
2.2.2.25	power	19
2.2.2.26	power	19
2.2.2.27	random_sample	19
2.2.2.28	random_sample	20
2.2.2.29	random_sample_n	20
2.2.2.30	random_sample_n	20
2.2.2.31	uninitialized_copy_n	21
2.3	Containers	22
2.3.1	Detailed Description	22
2.4	Sequences	23
2.4.1	Detailed Description	24
2.5	Associative	25
2.5.1	Detailed Description	25
2.6	Unordered Associative	26
2.6.1	Detailed Description	26
2.7	Diagnostics	27
2.7.1	Detailed Description	27
2.8	Concurrency	28
2.8.1	Detailed Description	28
2.9	Exceptions	29
2.9.1	Detailed Description	31
2.9.2	Typedef Documentation	31

---

---

2.9.2.1	terminate_handler . . . . .	31
2.9.2.2	unexpected_handler . . . . .	31
2.9.3	Function Documentation . . . . .	32
2.9.3.1	__verbose_terminate_handler . . . . .	32
2.9.3.2	copy_exception . . . . .	32
2.9.3.3	current_exception . . . . .	32
2.9.3.4	rethrow_exception . . . . .	32
2.9.3.5	rethrow_if_nested . . . . .	32
2.9.3.6	rethrow_if_nested . . . . .	33
2.9.3.7	set_terminate . . . . .	33
2.9.3.8	set_unexpected . . . . .	33
2.9.3.9	terminate . . . . .	33
2.9.3.10	throw_with_nested . . . . .	33
2.9.3.11	uncaught_exception . . . . .	33
2.9.3.12	unexpected . . . . .	34
2.10	Time . . . . .	34
2.10.1	Detailed Description . . . . .	34
2.11	Complex Numbers . . . . .	34
2.11.1	Detailed Description . . . . .	39
2.11.2	Function Documentation . . . . .	39
2.11.2.1	abs . . . . .	39
2.11.2.2	acos . . . . .	40
2.11.2.3	acosh . . . . .	40
2.11.2.4	arg . . . . .	40
2.11.2.5	arg . . . . .	40
2.11.2.6	asin . . . . .	40
2.11.2.7	asinh . . . . .	41
2.11.2.8	atan . . . . .	41
2.11.2.9	atanh . . . . .	41
2.11.2.10	conj . . . . .	41
2.11.2.11	cos . . . . .	41

---

2.11.2.12	cosh	41
2.11.2.13	exp	42
2.11.2.14	fabs	42
2.11.2.15	log	42
2.11.2.16	log10	42
2.11.2.17	norm	42
2.11.2.18	operator!=	43
2.11.2.19	operator!=	43
2.11.2.20	operator!=	43
2.11.2.21	operator*	43
2.11.2.22	operator*	43
2.11.2.23	operator*	43
2.11.2.24	operator*= operator*+=	44
2.11.2.25	operator*+=	44
2.11.2.26	operator+	44
2.11.2.27	operator+	44
2.11.2.28	operator+	44
2.11.2.29	operator+	44
2.11.2.30	operator+=	45
2.11.2.31	operator-	45
2.11.2.32	operator-	45
2.11.2.33	operator-	45
2.11.2.34	operator-	45
2.11.2.35	operator-=	45
2.11.2.36	operator/	46
2.11.2.37	operator/	46
2.11.2.38	operator/	46
2.11.2.39	operator/=	46
2.11.2.40	operator/=	46
2.11.2.41	operator<<	46
2.11.2.42	operator=	47

---

---

2.11.2.43	operator=	47
2.11.2.44	operator==	47
2.11.2.45	operator==	47
2.11.2.46	operator==	47
2.11.2.47	operator>>	48
2.11.2.48	polar	48
2.11.2.49	pow	48
2.11.2.50	pow	48
2.11.2.51	pow	48
2.11.2.52	sin	49
2.11.2.53	sinh	49
2.11.2.54	sqrt	49
2.11.2.55	tan	49
2.11.2.56	tanh	49
2.12	Condition Variables	50
2.12.1	Detailed Description	50
2.12.2	Enumeration Type Documentation	50
2.12.2.1	cv_status	50
2.13	Futures	51
2.13.1	Detailed Description	53
2.13.2	Enumeration Type Documentation	53
2.13.2.1	future_errc	53
2.13.3	Variable Documentation	53
2.13.3.1	future_category	53
2.14	I/O	53
2.14.1	Detailed Description	56
2.14.2	Typedef Documentation	57
2.14.2.1	filebuf	57
2.14.2.2	fstream	57
2.14.2.3	ifstream	57
2.14.2.4	ios	57

---

2.14.2.5	<code>iostream</code>	57
2.14.2.6	<code>istream</code>	57
2.14.2.7	<code>istringstream</code>	57
2.14.2.8	<code>ofstream</code>	58
2.14.2.9	<code>ostream</code>	58
2.14.2.10	<code>ostringstream</code>	58
2.14.2.11	<code>streambuf</code>	58
2.14.2.12	<code>stringbuf</code>	58
2.14.2.13	<code>stringstream</code>	58
2.14.2.14	<code>wfilebuf</code>	58
2.14.2.15	<code>wfstream</code>	59
2.14.2.16	<code>wifstream</code>	59
2.14.2.17	<code>wios</code>	59
2.14.2.18	<code>wiostream</code>	59
2.14.2.19	<code>wistream</code>	59
2.14.2.20	<code>wstringstream</code>	59
2.14.2.21	<code>wofstream</code>	59
2.14.2.22	<code>wostream</code>	60
2.14.2.23	<code>wstringstream</code>	60
2.14.2.24	<code>wstreambuf</code>	60
2.14.2.25	<code>wstringbuf</code>	60
2.14.2.26	<code>wstringstream</code>	60
2.15	Memory	61
2.15.1	Detailed Description	61
2.16	Pointer Abstractions	61
2.16.1	Detailed Description	64
2.16.2	Function Documentation	64
2.16.2.1	<code>allocate_shared</code>	64
2.16.2.2	<code>get_deleter</code>	64
2.16.2.3	<code>make_shared</code>	64
2.16.2.4	<code>operator&lt;&lt;</code>	65

---

---

2.17	Mutexes	65
2.17.1	Detailed Description	67
2.17.2	Function Documentation	67
2.17.2.1	call_once	67
2.17.2.2	lock	67
2.17.2.3	try_lock	67
2.18	Numerics	68
2.18.1	Detailed Description	68
2.19	Rational Arithmetic	69
2.19.1	Detailed Description	70
2.20	Threads	71
2.20.1	Detailed Description	71
2.21	Utilities	72
2.21.1	Detailed Description	72
2.22	Numeric Arrays	73
2.22.1	Detailed Description	83
2.22.2	Function Documentation	83
2.22.2.1	gslice	83
2.22.2.2	gslice	83
2.22.2.3	gslice	83
2.22.2.4	gslice_array	83
2.22.2.5	indirect_array	84
2.22.2.6	mask_array	84
2.22.2.7	slice	84
2.22.2.8	slice	84
2.22.2.9	slice_array	84
2.22.2.10	valarray	84
2.22.2.11	valarray	85
2.22.2.12	valarray	85
2.22.2.13	valarray	85
2.22.2.14	valarray	85

---

2.22.2.15 valarray . . . . .	85
2.22.2.16 valarray . . . . .	85
2.22.2.17 valarray . . . . .	86
2.22.2.18 valarray . . . . .	86
2.22.2.19 ~gslice . . . . .	86
2.22.2.20 apply . . . . .	86
2.22.2.21 apply . . . . .	86
2.22.2.22 cshift . . . . .	87
2.22.2.23 max . . . . .	87
2.22.2.24 min . . . . .	87
2.22.2.25 operator! . . . . .	88
2.22.2.26 operator%= . . . . .	88
2.22.2.27 operator%= . . . . .	88
2.22.2.28 operator%= . . . . .	88
2.22.2.29 operator&= . . . . .	88
2.22.2.30 operator&= . . . . .	88
2.22.2.31 operator&= . . . . .	89
2.22.2.32 operator*= . . . . .	89
2.22.2.33 operator*= . . . . .	89
2.22.2.34 operator*= . . . . .	89
2.22.2.35 operator+ . . . . .	89
2.22.2.36 operator+= . . . . .	89
2.22.2.37 operator+= . . . . .	90
2.22.2.38 operator+= . . . . .	90
2.22.2.39 operator- . . . . .	90
2.22.2.40 operator-= . . . . .	90
2.22.2.41 operator-= . . . . .	90
2.22.2.42 operator-= . . . . .	91
2.22.2.43 operator/= . . . . .	91
2.22.2.44 operator/= . . . . .	91
2.22.2.45 operator/= . . . . .	91

---

---

2.22.2.46 operator<<=	91
2.22.2.47 operator<<=	91
2.22.2.48 operator<<=	92
2.22.2.49 operator=	92
2.22.2.50 operator=	92
2.22.2.51 operator=	92
2.22.2.52 operator=	92
2.22.2.53 operator=	93
2.22.2.54 operator=	93
2.22.2.55 operator=	93
2.22.2.56 operator=	93
2.22.2.57 operator=	94
2.22.2.58 operator=	94
2.22.2.59 operator=	94
2.22.2.60 operator=	94
2.22.2.61 operator=	94
2.22.2.62 operator=	95
2.22.2.63 operator=	95
2.22.2.64 operator=	95
2.22.2.65 operator=	95
2.22.2.66 operator=	96
2.22.2.67 operator=	96
2.22.2.68 operator>>=	96
2.22.2.69 operator>>=	96
2.22.2.70 operator>>=	96
2.22.2.71 operator[]	97
2.22.2.72 operator[]	97
2.22.2.73 operator[]	97
2.22.2.74 operator[]	98
2.22.2.75 operator[]	98
2.22.2.76 operator[]	99

---

2.22.2.77	operator[]	99
2.22.2.78	operator[]	100
2.22.2.79	operator[]	100
2.22.2.80	operator^=	100
2.22.2.81	operator^=	101
2.22.2.82	operator^=	101
2.22.2.83	operator =	101
2.22.2.84	operator =	101
2.22.2.85	operator =	101
2.22.2.86	operator~	101
2.22.2.87	resize	102
2.22.2.88	shift	102
2.22.2.89	size	102
2.22.2.90	size	103
2.22.2.91	size	103
2.22.2.92	start	103
2.22.2.93	start	103
2.22.2.94	stride	103
2.22.2.95	stride	103
2.22.2.96	sum	103
2.23	Mathematical Special Functions	104
2.23.1	Detailed Description	107
2.23.2	Function Documentation	107
2.23.2.1	assoc_laguerre	107
2.23.2.2	assoc_legendre	107
2.23.2.3	beta	107
2.23.2.4	comp_ellint_1	108
2.23.2.5	comp_ellint_2	108
2.23.2.6	comp_ellint_3	108
2.23.2.7	conf_hyperg	108
2.23.2.8	cyl_bessel_i	108

---

---

2.23.2.9	<code>cyl_bessel_j</code>	108
2.23.2.10	<code>cyl_bessel_k</code>	109
2.23.2.11	<code>cyl_neumann</code>	109
2.23.2.12	<code>ellint_1</code>	109
2.23.2.13	<code>ellint_2</code>	109
2.23.2.14	<code>ellint_3</code>	109
2.23.2.15	<code>expint</code>	109
2.23.2.16	<code>hermite</code>	110
2.23.2.17	<code>hyperg</code>	110
2.23.2.18	<code>laguerre</code>	110
2.23.2.19	<code>legendre</code>	110
2.23.2.20	<code>riemann_zeta</code>	110
2.23.2.21	<code>sph_bessel</code>	110
2.23.2.22	<code>sph_legendre</code>	111
2.23.2.23	<code>sph_neumann</code>	111
2.24	Type Traits	111
2.24.1	Detailed Description	117
2.24.2	Typedef Documentation	117
2.24.2.1	<code>false_type</code>	117
2.24.2.2	<code>true_type</code>	117
2.25	Decimal Floating-Point Arithmetic	117
2.25.1	Detailed Description	117
2.26	Binder Classes	118
2.26.1	Detailed Description	119
2.26.2	Function Documentation	120
2.26.2.1	<code>bind</code>	120
2.26.2.2	<code>bind1st</code>	120
2.26.2.3	<code>bind2nd</code>	120
2.26.3	Variable Documentation	120
2.26.3.1	<code>_GLIBCXX_DEPRECATED_ATTR</code>	120
2.27	Algorithms	121

---

---

2.27.1 Detailed Description . . . . .	121
2.28 Mutating . . . . .	122
2.28.1 Function Documentation . . . . .	125
2.28.1.1 copy . . . . .	125
2.28.1.2 copy_backward . . . . .	125
2.28.1.3 copy_if . . . . .	126
2.28.1.4 copy_n . . . . .	126
2.28.1.5 fill . . . . .	127
2.28.1.6 fill_n . . . . .	127
2.28.1.7 generate . . . . .	128
2.28.1.8 generate_n . . . . .	128
2.28.1.9 is_partitioned . . . . .	129
2.28.1.10 iter_swap . . . . .	129
2.28.1.11 move . . . . .	130
2.28.1.12 move . . . . .	130
2.28.1.13 move_backward . . . . .	131
2.28.1.14 partition . . . . .	131
2.28.1.15 partition_copy . . . . .	132
2.28.1.16 partition_point . . . . .	132
2.28.1.17 random_shuffle . . . . .	133
2.28.1.18 random_shuffle . . . . .	133
2.28.1.19 remove . . . . .	134
2.28.1.20 remove_copy . . . . .	134
2.28.1.21 remove_copy_if . . . . .	135
2.28.1.22 remove_if . . . . .	135
2.28.1.23 replace . . . . .	136
2.28.1.24 replace_copy_if . . . . .	136
2.28.1.25 replace_if . . . . .	137
2.28.1.26 reverse . . . . .	137
2.28.1.27 reverse_copy . . . . .	138
2.28.1.28 rotate . . . . .	138

---

2.28.1.29 rotate_copy . . . . .	139
2.28.1.30 shuffle . . . . .	140
2.28.1.31 stable_partition . . . . .	140
2.28.1.32 swap . . . . .	141
2.28.1.33 swap_ranges . . . . .	141
2.28.1.34 transform . . . . .	142
2.28.1.35 transform . . . . .	142
2.28.1.36 unique . . . . .	143
2.28.1.37 unique . . . . .	143
2.28.1.38 unique_copy . . . . .	144
2.28.1.39 unique_copy . . . . .	144
2.29 Non-Mutating . . . . .	145
2.29.1 Function Documentation . . . . .	147
2.29.1.1 adjacent_find . . . . .	147
2.29.1.2 adjacent_find . . . . .	147
2.29.1.3 all_of . . . . .	148
2.29.1.4 any_of . . . . .	148
2.29.1.5 count . . . . .	149
2.29.1.6 count_if . . . . .	149
2.29.1.7 equal . . . . .	150
2.29.1.8 equal . . . . .	150
2.29.1.9 find . . . . .	151
2.29.1.10 find_end . . . . .	151
2.29.1.11 find_end . . . . .	152
2.29.1.12 find_first_of . . . . .	153
2.29.1.13 find_first_of . . . . .	153
2.29.1.14 find_if . . . . .	154
2.29.1.15 find_if_not . . . . .	154
2.29.1.16 for_each . . . . .	155
2.29.1.17 mismatch . . . . .	155
2.29.1.18 mismatch . . . . .	156

---

2.29.1.19	none_of	156
2.29.1.20	search	157
2.29.1.21	search	157
2.29.1.22	search_n	158
2.29.1.23	search_n	159
2.30	Sorting	160
2.30.1	Function Documentation	163
2.30.1.1	inplace_merge	163
2.30.1.2	inplace_merge	163
2.30.1.3	is_sorted	164
2.30.1.4	is_sorted	165
2.30.1.5	is_sorted_until	165
2.30.1.6	is_sorted_until	165
2.30.1.7	lexicographical_compare	166
2.30.1.8	lexicographical_compare	166
2.30.1.9	max	167
2.30.1.10	max	167
2.30.1.11	max_element	168
2.30.1.12	max_element	168
2.30.1.13	merge	169
2.30.1.14	merge	169
2.30.1.15	min	170
2.30.1.16	min	170
2.30.1.17	min_element	171
2.30.1.18	min_element	171
2.30.1.19	minmax	172
2.30.1.20	minmax	172
2.30.1.21	minmax_element	173
2.30.1.22	minmax_element	173
2.30.1.23	next_permutation	173
2.30.1.24	next_permutation	174

---

---

2.30.1.25	<code>nth_element</code>	174
2.30.1.26	<code>nth_element</code>	175
2.30.1.27	<code>partial_sort</code>	176
2.30.1.28	<code>partial_sort</code>	176
2.30.1.29	<code>partial_sort_copy</code>	177
2.30.1.30	<code>partial_sort_copy</code>	177
2.30.1.31	<code>prev_permutation</code>	178
2.30.1.32	<code>prev_permutation</code>	179
2.30.1.33	<code>sort</code>	179
2.30.1.34	<code>sort</code>	180
2.30.1.35	<code>stable_sort</code>	180
2.30.1.36	<code>stable_sort</code>	181
2.31	Set Operation	181
2.31.1	Detailed Description	183
2.31.2	Function Documentation	183
2.31.2.1	<code>includes</code>	183
2.31.2.2	<code>includes</code>	183
2.31.2.3	<code>set_difference</code>	184
2.31.2.4	<code>set_difference</code>	185
2.31.2.5	<code>set_intersection</code>	185
2.31.2.6	<code>set_intersection</code>	186
2.31.2.7	<code>set_symmetric_difference</code>	186
2.31.2.8	<code>set_symmetric_difference</code>	187
2.31.2.9	<code>set_union</code>	188
2.31.2.10	<code>set_union</code>	188
2.32	Binary Search	189
2.32.1	Detailed Description	190
2.32.2	Function Documentation	190
2.32.2.1	<code>binary_search</code>	190
2.32.2.2	<code>binary_search</code>	191
2.32.2.3	<code>equal_range</code>	191

---

2.32.2.4	<code>equal_range</code>	192
2.32.2.5	<code>lower_bound</code>	193
2.32.2.6	<code>lower_bound</code>	193
2.32.2.7	<code>upper_bound</code>	194
2.32.2.8	<code>upper_bound</code>	194
2.33	Allocators	195
2.33.1	Detailed Description	196
2.34	Atomics	196
2.34.1	Detailed Description	202
2.34.2	Define Documentation	202
2.34.2.1	<code>_GLIBCXX_ATOMIC_PROPERTY</code>	202
2.34.3	Typedef Documentation	202
2.34.3.1	<code>atomic_char</code>	202
2.34.3.2	<code>atomic_char16_t</code>	202
2.34.3.3	<code>atomic_char32_t</code>	203
2.34.3.4	<code>atomic_int</code>	203
2.34.3.5	<code>atomic_llong</code>	203
2.34.3.6	<code>atomic_long</code>	203
2.34.3.7	<code>atomic_schar</code>	203
2.34.3.8	<code>atomic_short</code>	203
2.34.3.9	<code>atomic_uchar</code>	203
2.34.3.10	<code>atomic_uint</code>	204
2.34.3.11	<code>atomic_ullong</code>	204
2.34.3.12	<code>atomic_ulong</code>	204
2.34.3.13	<code>atomic_ushort</code>	204
2.34.3.14	<code>atomic_wchar_t</code>	204
2.34.3.15	<code>memory_order</code>	204
2.34.4	Enumeration Type Documentation	204
2.34.4.1	<code>memory_order</code>	204
2.34.5	Function Documentation	205
2.34.5.1	<code>kill_dependency</code>	205

---

---

2.35 Hashes . . . . .	205
2.35.1 Detailed Description . . . . .	205
2.36 Locales . . . . .	206
2.36.1 Detailed Description . . . . .	208
2.37 Random Number Generation . . . . .	208
2.37.1 Detailed Description . . . . .	209
2.37.2 Function Documentation . . . . .	209
2.37.2.1 generate_canonical . . . . .	209
2.38 Function Objects . . . . .	210
2.38.1 Detailed Description . . . . .	211
2.38.2 Function Documentation . . . . .	212
2.38.2.1 mem_fn . . . . .	212
2.39 Arithmetic Classes . . . . .	212
2.39.1 Detailed Description . . . . .	213
2.40 Comparison Classes . . . . .	213
2.40.1 Detailed Description . . . . .	214
2.41 Boolean Operations Classes . . . . .	214
2.41.1 Detailed Description . . . . .	214
2.42 Negators . . . . .	215
2.42.1 Detailed Description . . . . .	215
2.42.2 Function Documentation . . . . .	216
2.42.2.1 not1 . . . . .	216
2.42.2.2 not2 . . . . .	216
2.43 Adaptors for pointers to functions . . . . .	216
2.43.1 Detailed Description . . . . .	217
2.43.2 Function Documentation . . . . .	217
2.43.2.1 ptr_fun . . . . .	217
2.43.2.2 ptr_fun . . . . .	218
2.44 Adaptors for pointers to members . . . . .	218
2.44.1 Detailed Description . . . . .	219
2.45 Heap . . . . .	219

---

---

2.45.1	Function Documentation	221
2.45.1.1	is_heap	221
2.45.1.2	is_heap	221
2.45.1.3	is_heap_until	221
2.45.1.4	is_heap_until	222
2.45.1.5	make_heap	222
2.45.1.6	make_heap	223
2.45.1.7	pop_heap	223
2.45.1.8	pop_heap	224
2.45.1.9	push_heap	224
2.45.1.10	push_heap	224
2.45.1.11	sort_heap	225
2.45.1.12	sort_heap	225
2.46	Iterators	226
2.46.1	Detailed Description	230
2.46.2	Function Documentation	230
2.46.2.1	__iterator_category	230
2.46.2.2	back_inserter	230
2.46.2.3	front_inserter	231
2.46.2.4	inserter	231
2.46.2.5	operator!=	231
2.46.2.6	operator==	232
2.46.2.7	operator==	232
2.47	Iterator Tags	232
2.47.1	Detailed Description	233
2.48	Strings	233
2.48.1	Typedef Documentation	234
2.48.1.1	string	234
2.48.1.2	u16string	234
2.48.1.3	u32string	234
2.48.1.4	wstring	234

---

2.49 Policy-Based Data Structures . . . . .	235
2.49.1 Detailed Description . . . . .	236
2.50 Random Number Generators . . . . .	236
2.50.1 Detailed Description . . . . .	238
2.50.2 Typedef Documentation . . . . .	239
2.50.2.1 minstd_rand . . . . .	239
2.50.2.2 minstd_rand0 . . . . .	239
2.50.2.3 mt19937 . . . . .	239
2.50.2.4 mt19937_64 . . . . .	239
2.50.3 Function Documentation . . . . .	240
2.50.3.1 operator!= . . . . .	240
2.50.3.2 operator!= . . . . .	240
2.50.3.3 operator!= . . . . .	241
2.50.3.4 operator!= . . . . .	241
2.50.3.5 operator!= . . . . .	242
2.50.3.6 operator!= . . . . .	242
2.50.3.7 operator<< . . . . .	243
2.51 Random Number Distributions . . . . .	243
2.52 Uniform . . . . .	244
2.52.1 Function Documentation . . . . .	245
2.52.1.1 operator!= . . . . .	245
2.52.1.2 operator!= . . . . .	245
2.52.1.3 operator<< . . . . .	245
2.52.1.4 operator<< . . . . .	246
2.52.1.5 operator== . . . . .	246
2.52.1.6 operator== . . . . .	246
2.52.1.7 operator>> . . . . .	247
2.52.1.8 operator>> . . . . .	247
2.53 Normal . . . . .	248
2.53.1 Function Documentation . . . . .	249
2.53.1.1 operator!= . . . . .	249

---

---

2.53.1.2	operator!=	249
2.53.1.3	operator!=	250
2.53.1.4	operator!=	250
2.53.1.5	operator!=	250
2.53.1.6	operator!=	250
2.53.1.7	operator!=	250
2.53.1.8	operator<<	251
2.53.1.9	operator==	251
2.53.1.10	operator>>	251
2.54	Bernoulli	252
2.54.1	Function Documentation	253
2.54.1.1	operator!=	253
2.54.1.2	operator!=	253
2.54.1.3	operator!=	254
2.54.1.4	operator!=	254
2.54.1.5	operator<<	254
2.54.1.6	operator<<	254
2.54.1.7	operator==	255
2.54.1.8	operator==	255
2.54.1.9	operator>>	255
2.54.1.10	operator>>	256
2.55	Poisson	256
2.55.1	Function Documentation	259
2.55.1.1	operator!=	259
2.55.1.2	operator!=	259
2.55.1.3	operator!=	259
2.55.1.4	operator!=	259
2.55.1.5	operator!=	259
2.55.1.6	operator!=	260
2.55.1.7	operator!=	260
2.55.1.8	operator<<	260

---

---

2.55.1.9	operator<<	260
2.55.1.10	operator<<	261
2.55.1.11	operator==	261
2.55.1.12	operator==	262
2.55.1.13	operator==	262
2.55.1.14	operator==	262
2.55.1.15	operator==	262
2.55.1.16	operator==	263
2.55.1.17	operator>>	263
2.55.1.18	operator>>	263
2.55.1.19	operator>>	264
2.56	Random Number Utilities	264
<b>3</b>	<b>Directory Documentation</b>	<b>267</b>
3.1	include/backward/ Directory Reference	268
3.2	include/x86_64-unknown-linux-gnu/bits/ Directory Reference	269
3.3	include/bits/ Directory Reference	271
3.4	include/debug/ Directory Reference	274
3.5	include/decimal/ Directory Reference	276
3.6	include/ext/pb_ds/detail/ Directory Reference	277
3.7	/mnt/share/src/gcc.svn-4_5-branch/libstdc++-v3/doc/ Directory Reference	278
3.8	/mnt/share/src/gcc.svn-4_5-branch/libstdc++-v3/doc/doxygen/ Directory Reference	279
3.9	include/ext/ Directory Reference	280
3.10	/mnt/share/src/gcc.svn-4_5-branch/ Directory Reference	282
3.11	include/profile/impl/ Directory Reference	283
3.12	include/ Directory Reference	285
3.13	/mnt/share/src/gcc.svn-4_5-branch/libstdc++-v3/ Directory Reference	288
3.14	/mnt/share/src/gcc.svn-4_5-branch/libstdc++-v3/libsupc++/ Directory Reference	289
3.15	include/parallel/ Directory Reference	290

3.16	<code>include/ext/pb_ds/</code> Directory Reference . . . . .	293
3.17	<code>include/profile/</code> Directory Reference . . . . .	295
3.18	<code>/mnt/share/src/</code> Directory Reference . . . . .	296
3.19	<code>include/tr1/</code> Directory Reference . . . . .	297
3.20	<code>include/tr1_impl/</code> Directory Reference . . . . .	298
3.21	<code>include/x86_64-unknown-linux-gnu/</code> Directory Reference . . . . .	299
<b>4</b>	<b>Namespace Documentation</b>	<b>301</b>
4.1	<code>__gnu_cxx</code> Namespace Reference . . . . .	301
4.1.1	Detailed Description . . . . .	323
4.1.2	Function Documentation . . . . .	323
4.1.2.1	<code>__static_pointer_cast</code> . . . . .	323
4.1.2.2	<code>__static_pointer_cast</code> . . . . .	323
4.1.2.3	<code>_Bit_scan_forward</code> . . . . .	324
4.1.2.4	<code>operator!=</code> . . . . .	324
4.1.2.5	<code>operator!=</code> . . . . .	324
4.1.2.6	<code>operator!=</code> . . . . .	325
4.1.2.7	<code>operator+</code> . . . . .	325
4.1.2.8	<code>operator+</code> . . . . .	326
4.1.2.9	<code>operator+</code> . . . . .	326
4.1.2.10	<code>operator+</code> . . . . .	327
4.1.2.11	<code>operator+</code> . . . . .	327
4.1.2.12	<code>operator&lt;</code> . . . . .	328
4.1.2.13	<code>operator&lt;</code> . . . . .	328
4.1.2.14	<code>operator&lt;</code> . . . . .	328
4.1.2.15	<code>operator&lt;=</code> . . . . .	329
4.1.2.16	<code>operator&lt;=</code> . . . . .	329
4.1.2.17	<code>operator&lt;=</code> . . . . .	330
4.1.2.18	<code>operator==</code> . . . . .	330
4.1.2.19	<code>operator==</code> . . . . .	330
4.1.2.20	<code>operator==</code> . . . . .	331

---

4.1.2.21	operator==	331
4.1.2.22	operator>	332
4.1.2.23	operator>	332
4.1.2.24	operator>	333
4.1.2.25	operator>=	333
4.1.2.26	operator>=	334
4.1.2.27	operator>=	334
4.1.2.28	swap	335
4.2	__gnu_cxx::__detail Namespace Reference	335
4.2.1	Detailed Description	336
4.2.2	Function Documentation	336
4.2.2.1	__bit_allocate	336
4.2.2.2	__bit_free	336
4.2.2.3	__num_bitmaps	336
4.2.2.4	__num_blocks	337
4.3	__gnu_cxx::typelist Namespace Reference	337
4.3.1	Detailed Description	337
4.3.2	Function Documentation	337
4.3.2.1	apply_generator	337
4.4	__gnu_debug Namespace Reference	338
4.4.1	Detailed Description	344
4.4.2	Function Documentation	344
4.4.2.1	__check_dereferenceable	344
4.4.2.2	__check_dereferenceable	344
4.4.2.3	__check_dereferenceable	345
4.4.2.4	__check_singular	345
4.4.2.5	__check_singular	345
4.4.2.6	__check_singular_aux	345
4.4.2.7	__check_string	345
4.4.2.8	__check_string	346
4.4.2.9	__valid_range	346

---

4.4.2.10	<code>__valid_range</code>	346
4.4.2.11	<code>__valid_range_aux</code>	346
4.4.2.12	<code>__valid_range_aux</code>	346
4.4.2.13	<code>__valid_range_aux2</code>	347
4.4.2.14	<code>__valid_range_aux2</code>	347
4.5	<code>__gnu_internal</code> Namespace Reference	347
4.5.1	Detailed Description	347
4.6	<code>__gnu_parallel</code> Namespace Reference	347
4.6.1	Detailed Description	364
4.6.2	Typedef Documentation	364
4.6.2.1	<code>_BinIndex</code>	364
4.6.2.2	<code>_CASable</code>	364
4.6.2.3	<code>_SequenceIndex</code>	364
4.6.2.4	<code>_ThreadIndex</code>	364
4.6.3	Enumeration Type Documentation	365
4.6.3.1	<code>_AlgorithmStrategy</code>	365
4.6.3.2	<code>_FindAlgorithm</code>	365
4.6.3.3	<code>_MultiwayMergeAlgorithm</code>	365
4.6.3.4	<code>_Parallelism</code>	365
4.6.3.5	<code>_PartialSumAlgorithm</code>	365
4.6.3.6	<code>_SortAlgorithm</code>	366
4.6.3.7	<code>_SplittingAlgorithm</code>	366
4.6.4	Function Documentation	366
4.6.4.1	<code>__calc_borders</code>	366
4.6.4.2	<code>__compare_and_swap</code>	366
4.6.4.3	<code>__compare_and_swap_32</code>	367
4.6.4.4	<code>__compare_and_swap_64</code>	367
4.6.4.5	<code>__decode2</code>	367
4.6.4.6	<code>__determine_samples</code>	368
4.6.4.7	<code>__encode2</code>	368
4.6.4.8	<code>__fetch_and_add</code>	369

---

---

4.6.4.9	<code>__fetch_and_add_32</code>	369
4.6.4.10	<code>__fetch_and_add_64</code>	370
4.6.4.11	<code>__find_template</code>	370
4.6.4.12	<code>__find_template</code>	371
4.6.4.13	<code>__find_template</code>	371
4.6.4.14	<code>__find_template</code>	372
4.6.4.15	<code>__for_each_template_random_access</code>	373
4.6.4.16	<code>__for_each_template_random_access_ed</code>	373
4.6.4.17	<code>__for_each_template_random_access_omp_loop</code>	374
4.6.4.18	<code>__for_each_template_random_access_omp_loop_-static</code>	375
4.6.4.19	<code>__for_each_template_random_access_workstealing</code>	376
4.6.4.20	<code>__is_sorted</code>	376
4.6.4.21	<code>__median_of_three_iterators</code>	377
4.6.4.22	<code>__merge_advance</code>	377
4.6.4.23	<code>__merge_advance_movc</code>	378
4.6.4.24	<code>__merge_advance_usual</code>	379
4.6.4.25	<code>__parallel_merge_advance</code>	379
4.6.4.26	<code>__parallel_merge_advance</code>	380
4.6.4.27	<code>__parallel_nth_element</code>	381
4.6.4.28	<code>__parallel_partial_sort</code>	381
4.6.4.29	<code>__parallel_partial_sum</code>	381
4.6.4.30	<code>__parallel_partial_sum_basecase</code>	382
4.6.4.31	<code>__parallel_partial_sum_linear</code>	383
4.6.4.32	<code>__parallel_partition</code>	383
4.6.4.33	<code>__parallel_random_shuffle</code>	384
4.6.4.34	<code>__parallel_random_shuffle_drs</code>	384
4.6.4.35	<code>__parallel_random_shuffle_drs_pu</code>	385
4.6.4.36	<code>__parallel_sort</code>	385
4.6.4.37	<code>__parallel_sort</code>	386
4.6.4.38	<code>__parallel_sort</code>	387

---

4.6.4.39	<code>__parallel_sort</code>	388
4.6.4.40	<code>__parallel_sort</code>	388
4.6.4.41	<code>__parallel_sort</code>	389
4.6.4.42	<code>__parallel_sort</code>	390
4.6.4.43	<code>__parallel_sort_qs</code>	390
4.6.4.44	<code>__parallel_sort_qs_conquer</code>	391
4.6.4.45	<code>__parallel_sort_qs_divide</code>	391
4.6.4.46	<code>__parallel_sort_qsb</code>	392
4.6.4.47	<code>__parallel_unique_copy</code>	392
4.6.4.48	<code>__parallel_unique_copy</code>	392
4.6.4.49	<code>__qsb_conquer</code>	393
4.6.4.50	<code>__qsb_divide</code>	394
4.6.4.51	<code>__qsb_local_sort_with_helping</code>	394
4.6.4.52	<code>__random_number_pow2</code>	395
4.6.4.53	<code>__rd_log2</code>	395
4.6.4.54	<code>__round_up_to_pow2</code>	395
4.6.4.55	<code>__search_template</code>	396
4.6.4.56	<code>__sequential_multiway_merge</code>	396
4.6.4.57	<code>__sequential_random_shuffle</code>	397
4.6.4.58	<code>__shrink</code>	397
4.6.4.59	<code>__shrink_and_double</code>	398
4.6.4.60	<code>__yield</code>	398
4.6.4.61	<code>equally_split</code>	398
4.6.4.62	<code>equally_split_point</code>	399
4.6.4.63	<code>list_partition</code>	399
4.6.4.64	<code>max</code>	400
4.6.4.65	<code>min</code>	400
4.6.4.66	<code>multiseq_partition</code>	401
4.6.4.67	<code>multiseq_selection</code>	401
4.6.4.68	<code>multiway_merge</code>	402
4.6.4.69	<code>multiway_merge_3_variant</code>	404

---

4.6.4.70	multiway_merge_4_variant . . . . .	405
4.6.4.71	multiway_merge_exact_splitting . . . . .	405
4.6.4.72	multiway_merge_loser_tree . . . . .	406
4.6.4.73	multiway_merge_loser_tree_sentinel . . . . .	407
4.6.4.74	multiway_merge_loser_tree_unguarded . . . . .	407
4.6.4.75	multiway_merge_sampling_splitting . . . . .	408
4.6.4.76	multiway_merge_sentinels . . . . .	408
4.6.4.77	parallel_multiway_merge . . . . .	410
4.6.4.78	parallel_sort_mwms . . . . .	411
4.6.4.79	parallel_sort_mwms_pu . . . . .	412
4.6.5	Variable Documentation . . . . .	412
4.6.5.1	_CASable_bits . . . . .	412
4.6.5.2	_CASable_mask . . . . .	412
4.7	__gnu_pbds Namespace Reference . . . . .	412
4.7.1	Detailed Description . . . . .	415
4.8	__gnu_profile Namespace Reference . . . . .	415
4.8.1	Detailed Description . . . . .	420
4.8.2	Typedef Documentation . . . . .	420
4.8.2.1	__env_t . . . . .	420
4.8.3	Function Documentation . . . . .	420
4.8.3.1	__get__global_lock . . . . .	420
4.8.3.2	__profcxx_init . . . . .	421
4.8.3.3	__report . . . . .	421
4.9	__gnu_sequential Namespace Reference . . . . .	421
4.9.1	Detailed Description . . . . .	421
4.10	abi Namespace Reference . . . . .	421
4.10.1	Detailed Description . . . . .	421
4.11	std Namespace Reference . . . . .	422
4.11.1	Detailed Description . . . . .	553
4.11.2	Typedef Documentation . . . . .	553
4.11.2.1	new_handler . . . . .	553

---

4.11.2.2	streamoff . . . . .	553
4.11.2.3	streampos . . . . .	554
4.11.2.4	streamsize . . . . .	554
4.11.2.5	u16streampos . . . . .	554
4.11.2.6	u32streampos . . . . .	554
4.11.2.7	wstreampos . . . . .	554
4.11.3	Enumeration Type Documentation . . . . .	554
4.11.3.1	"@53 . . . . .	554
4.11.3.2	float_denorm_style . . . . .	555
4.11.3.3	float_round_style . . . . .	555
4.11.4	Function Documentation . . . . .	555
4.11.4.1	__final_insertion_sort . . . . .	555
4.11.4.2	__final_insertion_sort . . . . .	556
4.11.4.3	__find . . . . .	556
4.11.4.4	__find . . . . .	556
4.11.4.5	__find_if . . . . .	556
4.11.4.6	__find_if . . . . .	556
4.11.4.7	__find_if_not . . . . .	557
4.11.4.8	__find_if_not . . . . .	557
4.11.4.9	__gcd . . . . .	557
4.11.4.10	__heap_select . . . . .	557
4.11.4.11	__heap_select . . . . .	557
4.11.4.12	__inplace_stable_partition . . . . .	558
4.11.4.13	__inplace_stable_sort . . . . .	558
4.11.4.14	__inplace_stable_sort . . . . .	558
4.11.4.15	__insertion_sort . . . . .	558
4.11.4.16	__insertion_sort . . . . .	559
4.11.4.17	__introsort_loop . . . . .	559
4.11.4.18	__introsort_loop . . . . .	559
4.11.4.19	__lg . . . . .	559
4.11.4.20	__merge_adaptive . . . . .	560

---

4.11.4.21	<code>__merge_adaptive</code>	560
4.11.4.22	<code>__merge_backward</code>	560
4.11.4.23	<code>__merge_backward</code>	561
4.11.4.24	<code>__merge_without_buffer</code>	561
4.11.4.25	<code>__merge_without_buffer</code>	561
4.11.4.26	<code>__move_median_first</code>	561
4.11.4.27	<code>__move_median_first</code>	562
4.11.4.28	<code>__partition</code>	562
4.11.4.29	<code>__partition</code>	562
4.11.4.30	<code>__reverse</code>	562
4.11.4.31	<code>__reverse</code>	563
4.11.4.32	<code>__rotate</code>	563
4.11.4.33	<code>__rotate</code>	563
4.11.4.34	<code>__rotate</code>	563
4.11.4.35	<code>__rotate_adaptive</code>	564
4.11.4.36	<code>__search_n</code>	564
4.11.4.37	<code>__search_n</code>	564
4.11.4.38	<code>__search_n</code>	564
4.11.4.39	<code>__search_n</code>	565
4.11.4.40	<code>__stable_partition_adaptive</code>	565
4.11.4.41	<code>__unguarded_insertion_sort</code>	565
4.11.4.42	<code>__unguarded_insertion_sort</code>	565
4.11.4.43	<code>__unguarded_linear_insert</code>	566
4.11.4.44	<code>__unguarded_linear_insert</code>	566
4.11.4.45	<code>__unguarded_partition</code>	566
4.11.4.46	<code>__unguarded_partition</code>	566
4.11.4.47	<code>__unguarded_partition_pivot</code>	567
4.11.4.48	<code>__unguarded_partition_pivot</code>	567
4.11.4.49	<code>__unique_copy</code>	567
4.11.4.50	<code>__unique_copy</code>	567
4.11.4.51	<code>__unique_copy</code>	568

---

---

4.11.4.52	<code>__unique_copy</code>	568
4.11.4.53	<code>__unique_copy</code>	568
4.11.4.54	<code>__unique_copy</code>	568
4.11.4.55	<code>_Construct</code>	569
4.11.4.56	<code>_Destroy</code>	569
4.11.4.57	<code>_Destroy</code>	569
4.11.4.58	<code>accumulate</code>	569
4.11.4.59	<code>accumulate</code>	570
4.11.4.60	<code>adjacent_difference</code>	570
4.11.4.61	<code>adjacent_difference</code>	571
4.11.4.62	<code>advance</code>	571
4.11.4.63	<code>bind</code>	572
4.11.4.64	<code>boolalpha</code>	572
4.11.4.65	<code>cref</code>	572
4.11.4.66	<code>cref</code>	572
4.11.4.67	<code>dec</code>	572
4.11.4.68	<code>distance</code>	573
4.11.4.69	<code>endl</code>	573
4.11.4.70	<code>ends</code>	574
4.11.4.71	<code>fixed</code>	574
4.11.4.72	<code>flush</code>	574
4.11.4.73	<code>forward</code>	574
4.11.4.74	<code>forward</code>	574
4.11.4.75	<code>get_money</code>	575
4.11.4.76	<code>get_temporary_buffer</code>	575
4.11.4.77	<code>getline</code>	576
4.11.4.78	<code>getline</code>	576
4.11.4.79	<code>getline</code>	577
4.11.4.80	<code>getline</code>	577
4.11.4.81	<code>has_facet</code>	578
4.11.4.82	<code>hex</code>	578

---

4.11.4.83 inner_product . . . . .	579
4.11.4.84 inner_product . . . . .	579
4.11.4.85 internal . . . . .	580
4.11.4.86 iota . . . . .	580
4.11.4.87 isalnum . . . . .	580
4.11.4.88 isalpha . . . . .	580
4.11.4.89 iscntrl . . . . .	581
4.11.4.90 isdigit . . . . .	581
4.11.4.91 isgraph . . . . .	581
4.11.4.92 islower . . . . .	581
4.11.4.93 isprint . . . . .	581
4.11.4.94 ispunct . . . . .	581
4.11.4.95 isspace . . . . .	581
4.11.4.96 isupper . . . . .	581
4.11.4.97 isxdigit . . . . .	582
4.11.4.98 left . . . . .	582
4.11.4.99 noboolalpha . . . . .	582
4.11.4.100noshowbase . . . . .	582
4.11.4.101noshowpoint . . . . .	582
4.11.4.102noshowpos . . . . .	582
4.11.4.103noskipws . . . . .	583
4.11.4.104noinitbuf . . . . .	583
4.11.4.105nouppercase . . . . .	583
4.11.4.106oct . . . . .	583
4.11.4.107operator!= . . . . .	583
4.11.4.108operator!= . . . . .	583
4.11.4.109operator!= . . . . .	584
4.11.4.110operator!= . . . . .	584
4.11.4.111operator!= . . . . .	585
4.11.4.112operator!= . . . . .	585
4.11.4.113operator!= . . . . .	585

---

4.11.4.114operator!=	585
4.11.4.115operator!=	585
4.11.4.116operator!=	586
4.11.4.117operator!=	586
4.11.4.118operator!=	586
4.11.4.119operator!=	586
4.11.4.120operator!=	586
4.11.4.121operator!=	587
4.11.4.122operator!=	587
4.11.4.123operator!=	587
4.11.4.124operator&	587
4.11.4.125operator+	588
4.11.4.126operator+	588
4.11.4.127operator+	588
4.11.4.128operator+	589
4.11.4.129operator+	589
4.11.4.130operator<	590
4.11.4.131operator<	590
4.11.4.132operator<	591
4.11.4.133operator<	591
4.11.4.134operator<	591
4.11.4.135operator<	592
4.11.4.136operator<	592
4.11.4.137operator<	593
4.11.4.138operator<	593
4.11.4.139operator<	593
4.11.4.140operator<	594
4.11.4.141operator<	594
4.11.4.142operator<	595
4.11.4.143operator<	595
4.11.4.144operator<<	596

---

4.11.4.145operator<<	596
4.11.4.146operator<<	597
4.11.4.147operator<<	597
4.11.4.148operator<<	598
4.11.4.149operator<<	598
4.11.4.150operator<<	599
4.11.4.151operator<<	599
4.11.4.152operator<<	600
4.11.4.153operator<<	600
4.11.4.154operator<<	601
4.11.4.155operator<<	601
4.11.4.156operator<<	602
4.11.4.157operator<<	602
4.11.4.158operator<=	603
4.11.4.159operator<=	603
4.11.4.160operator<=	603
4.11.4.161operator<=	604
4.11.4.162operator<=	604
4.11.4.163operator<=	604
4.11.4.164operator<=	605
4.11.4.165operator<=	605
4.11.4.166operator<=	605
4.11.4.167operator<=	605
4.11.4.168operator<=	605
4.11.4.169operator<=	606
4.11.4.170operator<=	606
4.11.4.171operator<=	606
4.11.4.172operator==	606
4.11.4.173operator==	607
4.11.4.174operator==	607
4.11.4.175operator==	607

---

4.11.4.176operator==	608
4.11.4.177operator==	608
4.11.4.178operator==	609
4.11.4.179operator==	609
4.11.4.180operator==	609
4.11.4.181operator==	610
4.11.4.182operator==	610
4.11.4.183operator==	611
4.11.4.184operator==	611
4.11.4.185operator==	611
4.11.4.186operator==	612
4.11.4.187operator==	612
4.11.4.188operator==	612
4.11.4.189operator==	613
4.11.4.190operator>	613
4.11.4.191operator>	613
4.11.4.192operator>	613
4.11.4.193operator>	614
4.11.4.194operator>	614
4.11.4.195operator>	614
4.11.4.196operator>	614
4.11.4.197operator>	615
4.11.4.198operator>	615
4.11.4.199operator>	615
4.11.4.200operator>	615
4.11.4.201operator>	616
4.11.4.202operator>	616
4.11.4.203operator>	616
4.11.4.204operator>=	616
4.11.4.205operator>=	617
4.11.4.206operator>=	617

---

---

4.11.4.207operator>=	617
4.11.4.208operator>=	617
4.11.4.209operator>=	618
4.11.4.210operator>=	618
4.11.4.211operator>=	618
4.11.4.212operator>=	618
4.11.4.213operator>=	618
4.11.4.214operator>=	619
4.11.4.215operator>=	619
4.11.4.216operator>=	619
4.11.4.217operator>=	620
4.11.4.218operator>>	620
4.11.4.219operator>>	620
4.11.4.220operator>>	621
4.11.4.221operator>>	621
4.11.4.222operator>>	622
4.11.4.223operator>>	623
4.11.4.224operator>>	623
4.11.4.225operator>>	624
4.11.4.226operator>>	624
4.11.4.227operator>>	625
4.11.4.228operator>>	626
4.11.4.229operator^	626
4.11.4.230operator	627
4.11.4.231partial_sum	627
4.11.4.232partial_sum	628
4.11.4.233put_money	628
4.11.4.234ref	628
4.11.4.235ref	629
4.11.4.236replace_copy	629
4.11.4.237resetiosflags	629

---

---

4.11.4.238	<code>return_temporary_buffer</code>	630
4.11.4.239	<code>right</code>	630
4.11.4.240	<code>scientific</code>	630
4.11.4.241	<code>set_new_handler</code>	630
4.11.4.242	<code>setbase</code>	631
4.11.4.243	<code>setfill</code>	631
4.11.4.244	<code>setiosflags</code>	631
4.11.4.245	<code>setprecision</code>	631
4.11.4.246	<code>setw</code>	632
4.11.4.247	<code>showbase</code>	632
4.11.4.248	<code>showpoint</code>	632
4.11.4.249	<code>showpos</code>	632
4.11.4.250	<code>skipws</code>	632
4.11.4.251	<code>swap</code>	633
4.11.4.252	<code>swap</code>	633
4.11.4.253	<code>swap</code>	633
4.11.4.254	<code>swap</code>	633
4.11.4.255	<code>swap</code>	634
4.11.4.256	<code>swap</code>	634
4.11.4.257	<code>swap</code>	634
4.11.4.258	<code>swap</code>	635
4.11.4.259	<code>swap</code>	635
4.11.4.260	<code>swap</code>	635
4.11.4.261	<code>swap</code>	635
4.11.4.262	<code>tolower</code>	636
4.11.4.263	<code>toupper</code>	636
4.11.4.264	<code>uninitialized_copy</code>	636
4.11.4.265	<code>uninitialized_copy_n</code>	636
4.11.4.266	<code>uninitialized_fill</code>	637
4.11.4.267	<code>uninitialized_fill_n</code>	637
4.11.4.268	<code>unitbuf</code>	638

---

4.11.4.26	uppercase	638
4.11.4.27	use_facet	638
4.11.4.27	lws	638
4.11.5	Variable Documentation	639
4.11.5.1	__invoke	639
4.11.5.2	cerr	639
4.11.5.3	cin	639
4.11.5.4	clog	639
4.11.5.5	cout	640
4.11.5.6	wcerr	640
4.11.5.7	wcin	640
4.11.5.8	wclog	640
4.11.5.9	wcout	640
4.12	std::__debug Namespace Reference	640
4.12.1	Detailed Description	646
4.13	std::__detail Namespace Reference	646
4.13.1	Detailed Description	647
4.14	std::__parallel Namespace Reference	647
4.14.1	Detailed Description	672
4.15	std::__profile Namespace Reference	672
4.15.1	Detailed Description	679
4.16	std::__chrono Namespace Reference	679
4.16.1	Detailed Description	682
4.16.2	Typedef Documentation	682
4.16.2.1	hours	682
4.16.2.2	microseconds	682
4.16.2.3	milliseconds	682
4.16.2.4	minutes	682
4.16.2.5	nanoseconds	683
4.16.2.6	seconds	683
4.16.3	Function Documentation	683

---

---

4.16.3.1	<code>duration_cast</code> . . . . .	683
4.16.3.2	<code>time_point_cast</code> . . . . .	683
4.17	<code>std::decimal</code> Namespace Reference . . . . .	683
4.17.1	Detailed Description . . . . .	694
4.17.2	Function Documentation . . . . .	694
4.17.2.1	<code>decimal32_to_long_long</code> . . . . .	694
4.18	<code>std::placeholders</code> Namespace Reference . . . . .	694
4.18.1	Detailed Description . . . . .	695
4.19	<code>std::rel_ops</code> Namespace Reference . . . . .	695
4.19.1	Detailed Description . . . . .	695
4.19.2	Function Documentation . . . . .	695
4.19.2.1	<code>operator!=</code> . . . . .	695
4.19.2.2	<code>operator&lt;=</code> . . . . .	696
4.19.2.3	<code>operator&gt;</code> . . . . .	696
4.19.2.4	<code>operator&gt;=</code> . . . . .	696
4.20	<code>std::this_thread</code> Namespace Reference . . . . .	697
4.20.1	Detailed Description . . . . .	697
4.20.2	Function Documentation . . . . .	697
4.20.2.1	<code>get_id</code> . . . . .	697
4.20.2.2	<code>sleep_for</code> . . . . .	697
4.20.2.3	<code>sleep_until</code> . . . . .	698
4.20.2.4	<code>yield</code> . . . . .	698
4.21	<code>std::tr1</code> Namespace Reference . . . . .	698
4.21.1	Detailed Description . . . . .	706
4.22	<code>std::tr1::__detail</code> Namespace Reference . . . . .	706
4.22.1	Detailed Description . . . . .	706
<b>5</b>	<b>Class Documentation</b>	<b>707</b>
5.1	<code>__atomic0::atomic_address</code> Struct Reference . . . . .	707
5.1.1	Detailed Description . . . . .	708
5.2	<code>__atomic0::atomic_bool</code> Struct Reference . . . . .	708

---

---

5.2.1	Detailed Description . . . . .	708
5.3	<code>__atomic0::atomic_flag</code> Struct Reference . . . . .	709
5.3.1	Detailed Description . . . . .	709
5.4	<code>__atomic2::atomic_address</code> Struct Reference . . . . .	709
5.4.1	Detailed Description . . . . .	710
5.5	<code>__atomic2::atomic_bool</code> Struct Reference . . . . .	710
5.5.1	Detailed Description . . . . .	711
5.6	<code>__atomic2::atomic_flag</code> Struct Reference . . . . .	711
5.6.1	Detailed Description . . . . .	711
5.7	<code>__cxxabiv1::__forced_unwind</code> Class Reference . . . . .	711
5.7.1	Detailed Description . . . . .	711
5.8	<code>__gnu_cxx::__common_pool_policy&lt; _PoolTp, _Thread &gt;</code> Struct Template Reference . . . . .	712
5.8.1	Detailed Description . . . . .	712
5.9	<code>__gnu_cxx::__detail::__mini_vector&lt; _Tp &gt;</code> Class Template Reference . . . . .	712
5.9.1	Detailed Description . . . . .	713
5.10	<code>__gnu_cxx::__detail::__Bitmap_counter&lt; _Tp &gt;</code> Class Template Ref- erence . . . . .	713
5.10.1	Detailed Description . . . . .	714
5.11	<code>__gnu_cxx::__detail::__Ffit_finder&lt; _Tp &gt;</code> Class Template Reference . . . . .	715
5.11.1	Detailed Description . . . . .	716
5.11.2	Member Typedef Documentation . . . . .	716
5.11.2.1	<code>argument_type</code> . . . . .	716
5.11.2.2	<code>result_type</code> . . . . .	716
5.12	<code>__gnu_cxx::__mt_alloc&lt; _Tp, _Poolp &gt;</code> Class Template Reference . . . . .	716
5.12.1	Detailed Description . . . . .	718
5.13	<code>__gnu_cxx::__mt_alloc_base&lt; _Tp &gt;</code> Class Template Reference . . . . .	718
5.13.1	Detailed Description . . . . .	719
5.14	<code>__gnu_cxx::__per_type_pool_policy&lt; _Tp, _PoolTp, _Thread &gt;</code> Struct Template Reference . . . . .	719
5.14.1	Detailed Description . . . . .	720
5.15	<code>__gnu_cxx::__pool&lt; false &gt;</code> Class Template Reference . . . . .	720

---

---

5.15.1 Detailed Description . . . . .	721
5.16 <code>__gnu_cxx::__pool&lt; true &gt;</code> Class Template Reference . . . . .	721
5.16.1 Detailed Description . . . . .	723
5.17 <code>__gnu_cxx::__pool_alloc&lt; _Tp &gt;</code> Class Template Reference . . . . .	723
5.17.1 Detailed Description . . . . .	725
5.18 <code>__gnu_cxx::__pool_alloc_base</code> Class Reference . . . . .	725
5.18.1 Detailed Description . . . . .	726
5.19 <code>__gnu_cxx::__pool_base</code> Struct Reference . . . . .	726
5.19.1 Detailed Description . . . . .	727
5.20 <code>__gnu_cxx::__rc_string_base&lt; _CharT, _Traits, _Alloc &gt;</code> Class Template Reference . . . . .	728
5.20.1 Detailed Description . . . . .	730
5.21 <code>__gnu_cxx::__scoped_lock</code> Class Reference . . . . .	731
5.21.1 Detailed Description . . . . .	731
5.22 <code>__gnu_cxx::__versa_string&lt; _CharT, _Traits, _Alloc, _Base &gt;</code> Class Template Reference . . . . .	731
5.22.1 Detailed Description . . . . .	736
5.22.2 Constructor & Destructor Documentation . . . . .	736
5.22.2.1 <code>__versa_string</code> . . . . .	736
5.22.2.2 <code>__versa_string</code> . . . . .	736
5.22.2.3 <code>__versa_string</code> . . . . .	736
5.22.2.4 <code>__versa_string</code> . . . . .	737
5.22.2.5 <code>__versa_string</code> . . . . .	737
5.22.2.6 <code>__versa_string</code> . . . . .	737
5.22.2.7 <code>__versa_string</code> . . . . .	738
5.22.2.8 <code>__versa_string</code> . . . . .	738
5.22.2.9 <code>__versa_string</code> . . . . .	739
5.22.2.10 <code>__versa_string</code> . . . . .	739
5.22.2.11 <code>__versa_string</code> . . . . .	739
5.22.2.12 <code>~__versa_string</code> . . . . .	740
5.22.3 Member Function Documentation . . . . .	740
5.22.3.1 <code>append</code> . . . . .	740

---

---

5.22.3.2	append	740
5.22.3.3	append	741
5.22.3.4	append	741
5.22.3.5	append	742
5.22.3.6	append	742
5.22.3.7	append	743
5.22.3.8	assign	743
5.22.3.9	assign	743
5.22.3.10	assign	744
5.22.3.11	assign	745
5.22.3.12	assign	745
5.22.3.13	assign	745
5.22.3.14	assign	746
5.22.3.15	assign	746
5.22.3.16	at	747
5.22.3.17	at	747
5.22.3.18	back	748
5.22.3.19	back	748
5.22.3.20	begin	748
5.22.3.21	begin	748
5.22.3.22	c_str	749
5.22.3.23	capacity	749
5.22.3.24	cbegin	749
5.22.3.25	cend	749
5.22.3.26	clear	750
5.22.3.27	compare	750
5.22.3.28	compare	750
5.22.3.29	compare	751
5.22.3.30	compare	752
5.22.3.31	compare	753
5.22.3.32	compare	753

---

5.22.3.33 copy . . . . .	754
5.22.3.34 crbegin . . . . .	755
5.22.3.35 crend . . . . .	755
5.22.3.36 data . . . . .	755
5.22.3.37 empty . . . . .	755
5.22.3.38 end . . . . .	756
5.22.3.39 end . . . . .	756
5.22.3.40 erase . . . . .	756
5.22.3.41 erase . . . . .	757
5.22.3.42 erase . . . . .	757
5.22.3.43 find . . . . .	757
5.22.3.44 find . . . . .	758
5.22.3.45 find . . . . .	759
5.22.3.46 find . . . . .	759
5.22.3.47 find_first_not_of . . . . .	760
5.22.3.48 find_first_not_of . . . . .	760
5.22.3.49 find_first_not_of . . . . .	761
5.22.3.50 find_first_not_of . . . . .	761
5.22.3.51 find_first_of . . . . .	762
5.22.3.52 find_first_of . . . . .	762
5.22.3.53 find_first_of . . . . .	763
5.22.3.54 find_first_of . . . . .	763
5.22.3.55 find_last_not_of . . . . .	764
5.22.3.56 find_last_not_of . . . . .	764
5.22.3.57 find_last_not_of . . . . .	765
5.22.3.58 find_last_not_of . . . . .	766
5.22.3.59 find_last_of . . . . .	766
5.22.3.60 find_last_of . . . . .	767
5.22.3.61 find_last_of . . . . .	767
5.22.3.62 find_last_of . . . . .	768
5.22.3.63 front . . . . .	768

---

---

5.22.3.64	front	768
5.22.3.65	get_allocator	769
5.22.3.66	insert	769
5.22.3.67	insert	769
5.22.3.68	insert	770
5.22.3.69	insert	771
5.22.3.70	insert	771
5.22.3.71	insert	772
5.22.3.72	insert	772
5.22.3.73	insert	773
5.22.3.74	insert	773
5.22.3.75	length	774
5.22.3.76	max_size	774
5.22.3.77	operator+=	775
5.22.3.78	operator+=	775
5.22.3.79	operator+=	775
5.22.3.80	operator+=	776
5.22.3.81	operator=	776
5.22.3.82	operator=	776
5.22.3.83	operator=	777
5.22.3.84	operator=	777
5.22.3.85	operator=	777
5.22.3.86	operator[]	778
5.22.3.87	operator[]	778
5.22.3.88	push_back	779
5.22.3.89	rbegin	779
5.22.3.90	rbegin	779
5.22.3.91	rend	779
5.22.3.92	rend	780
5.22.3.93	replace	780
5.22.3.94	replace	781

---

5.22.3.95	replace	781
5.22.3.96	replace	782
5.22.3.97	replace	783
5.22.3.98	replace	783
5.22.3.99	replace	784
5.22.3.100	replace	785
5.22.3.101	replace	785
5.22.3.102	replace	786
5.22.3.103	replace	787
5.22.3.104	reserve	787
5.22.3.105	resize	788
5.22.3.106	resize	788
5.22.3.107	find	789
5.22.3.108	find	789
5.22.3.109	find	790
5.22.3.110	find	790
5.22.3.111	lshrink_to_fit	791
5.22.3.112	size	791
5.22.3.113	substr	791
5.22.3.114	swap	792
5.22.4	Member Data Documentation	792
5.22.4.1	npos	792
5.23	<code>__gnu_cxx::_Caster&lt;_ToType&gt;</code> Struct Template Reference	793
5.23.1	Detailed Description	793
5.24	<code>__gnu_cxx::_Char_types&lt;_CharT&gt;</code> Struct Template Reference	793
5.24.1	Detailed Description	794
5.25	<code>__gnu_cxx::_ExtPtr_allocator&lt;_Tp&gt;</code> Class Template Reference	794
5.25.1	Detailed Description	795
5.26	<code>__gnu_cxx::_Invalid_type</code> Struct Reference	796
5.26.1	Detailed Description	796

---

---

5.27	<a href="#">__gnu_cxx::Pointer_adapter&lt; _Storage_policy &gt; Class Template Reference</a>	796
5.27.1	<a href="#">Detailed Description</a>	798
5.28	<a href="#">__gnu_cxx::Relative_pointer_impl&lt; _Tp &gt; Class Template Reference</a>	799
5.28.1	<a href="#">Detailed Description</a>	799
5.29	<a href="#">__gnu_cxx::Relative_pointer_impl&lt; const _Tp &gt; Class Template Reference</a>	800
5.29.1	<a href="#">Detailed Description</a>	800
5.30	<a href="#">__gnu_cxx::Std_pointer_impl&lt; _Tp &gt; Class Template Reference</a>	800
5.30.1	<a href="#">Detailed Description</a>	801
5.31	<a href="#">__gnu_cxx::Unqualified_type&lt; _Tp &gt; Struct Template Reference</a>	801
5.31.1	<a href="#">Detailed Description</a>	801
5.32	<a href="#">__gnu_cxx::annotate_base Struct Reference</a>	802
5.32.1	<a href="#">Detailed Description</a>	802
5.33	<a href="#">__gnu_cxx::array_allocator&lt; _Tp, _Array &gt; Class Template Reference</a>	803
5.33.1	<a href="#">Detailed Description</a>	804
5.34	<a href="#">__gnu_cxx::array_allocator_base&lt; _Tp &gt; Class Template Reference</a>	804
5.34.1	<a href="#">Detailed Description</a>	805
5.35	<a href="#">__gnu_cxx::binary_compose&lt; _Operation1, _Operation2, _Operation3 &gt; Class Template Reference</a>	805
5.35.1	<a href="#">Detailed Description</a>	806
5.35.2	<a href="#">Member Typedef Documentation</a>	807
5.35.2.1	<a href="#">argument_type</a>	807
5.35.2.2	<a href="#">result_type</a>	807
5.36	<a href="#">__gnu_cxx::bitmap_allocator&lt; _Tp &gt; Class Template Reference</a>	807
5.36.1	<a href="#">Detailed Description</a>	809
5.36.2	<a href="#">Member Function Documentation</a>	809
5.36.2.1	<a href="#">_M_allocate_single_object</a>	809
5.36.2.2	<a href="#">_M_deallocate_single_object</a>	809
5.37	<a href="#">__gnu_cxx::char_traits&lt; _CharT &gt; Struct Template Reference</a>	810
5.37.1	<a href="#">Detailed Description</a>	811
5.38	<a href="#">__gnu_cxx::character&lt; V, I, S &gt; Struct Template Reference</a>	811

---

5.38.1 Detailed Description . . . . .	812
5.39 <code>__gnu_cxx::condition_base</code> Struct Reference . . . . .	812
5.39.1 Detailed Description . . . . .	813
5.40 <code>__gnu_cxx::constant_binary_fun&lt; _Result, _Arg1, _Arg2 &gt;</code> Struct Template Reference . . . . .	813
5.40.1 Detailed Description . . . . .	814
5.41 <code>__gnu_cxx::constant_unary_fun&lt; _Result, _Argument &gt;</code> Struct Tem- plate Reference . . . . .	814
5.41.1 Detailed Description . . . . .	815
5.42 <code>__gnu_cxx::constant_void_fun&lt; _Result &gt;</code> Struct Template Reference	815
5.42.1 Detailed Description . . . . .	815
5.43 <code>__gnu_cxx::debug_allocator&lt; _Alloc &gt;</code> Class Template Reference . .	816
5.43.1 Detailed Description . . . . .	816
5.44 <code>__gnu_cxx::enc_filebuf&lt; _CharT &gt;</code> Class Template Reference . . . .	817
5.44.1 Detailed Description . . . . .	820
5.44.2 Member Typedef Documentation . . . . .	820
5.44.2.1 <code>__streambuf_type</code> . . . . .	820
5.44.2.2 <code>char_type</code> . . . . .	821
5.44.2.3 <code>int_type</code> . . . . .	821
5.44.2.4 <code>off_type</code> . . . . .	821
5.44.2.5 <code>pos_type</code> . . . . .	821
5.44.2.6 <code>traits_type</code> . . . . .	822
5.44.3 Member Function Documentation . . . . .	822
5.44.3.1 <code>_M_create_pback</code> . . . . .	822
5.44.3.2 <code>_M_destroy_pback</code> . . . . .	822
5.44.3.3 <code>_M_set_buffer</code> . . . . .	822
5.44.3.4 <code>close</code> . . . . .	823
5.44.3.5 <code>eback</code> . . . . .	823
5.44.3.6 <code>egptr</code> . . . . .	823
5.44.3.7 <code>epptr</code> . . . . .	824
5.44.3.8 <code>gbump</code> . . . . .	824
5.44.3.9 <code>getloc</code> . . . . .	824

---

5.44.3.10 gptr . . . . .	825
5.44.3.11 imbue . . . . .	825
5.44.3.12 in_avail . . . . .	826
5.44.3.13 is_open . . . . .	826
5.44.3.14 open . . . . .	826
5.44.3.15 open . . . . .	827
5.44.3.16 overflow . . . . .	827
5.44.3.17 pbackfail . . . . .	828
5.44.3.18 pbase . . . . .	828
5.44.3.19 pbump . . . . .	828
5.44.3.20 pptr . . . . .	829
5.44.3.21 pubimbue . . . . .	829
5.44.3.22 pubseekoff . . . . .	829
5.44.3.23 pubseekpos . . . . .	830
5.44.3.24 pubsetbuf . . . . .	830
5.44.3.25 pubsync . . . . .	830
5.44.3.26 sbumpc . . . . .	831
5.44.3.27 seekoff . . . . .	831
5.44.3.28 seekpos . . . . .	831
5.44.3.29 setbuf . . . . .	832
5.44.3.30 setbuf . . . . .	832
5.44.3.31 setg . . . . .	833
5.44.3.32 setp . . . . .	833
5.44.3.33 sgetc . . . . .	833
5.44.3.34 sgetn . . . . .	834
5.44.3.35 showmanyc . . . . .	834
5.44.3.36 snextc . . . . .	834
5.44.3.37 sputbackc . . . . .	835
5.44.3.38 sputc . . . . .	835
5.44.3.39 sputn . . . . .	836
5.44.3.40 stossc . . . . .	836

---

5.44.3.41	<code>sungetc</code> . . . . .	836
5.44.3.42	<code>sync</code> . . . . .	837
5.44.3.43	<code>uflow</code> . . . . .	837
5.44.3.44	<code>underflow</code> . . . . .	837
5.44.3.45	<code>xsgetn</code> . . . . .	838
5.44.3.46	<code>xspn</code> . . . . .	838
5.44.4	Member Data Documentation . . . . .	839
5.44.4.1	<code>_M_buf</code> . . . . .	839
5.44.4.2	<code>_M_buf_locale</code> . . . . .	839
5.44.4.3	<code>_M_buf_size</code> . . . . .	839
5.44.4.4	<code>_M_ext_buf</code> . . . . .	839
5.44.4.5	<code>_M_ext_buf_size</code> . . . . .	840
5.44.4.6	<code>_M_ext_next</code> . . . . .	840
5.44.4.7	<code>_M_in_beg</code> . . . . .	840
5.44.4.8	<code>_M_in_cur</code> . . . . .	840
5.44.4.9	<code>_M_in_end</code> . . . . .	841
5.44.4.10	<code>_M_mode</code> . . . . .	841
5.44.4.11	<code>_M_out_beg</code> . . . . .	841
5.44.4.12	<code>_M_out_cur</code> . . . . .	841
5.44.4.13	<code>_M_out_end</code> . . . . .	842
5.44.4.14	<code>_M_pback</code> . . . . .	842
5.44.4.15	<code>_M_pback_cur_save</code> . . . . .	842
5.44.4.16	<code>_M_pback_end_save</code> . . . . .	843
5.44.4.17	<code>_M_pback_init</code> . . . . .	843
5.44.4.18	<code>_M_reading</code> . . . . .	843
5.45	<code>__gnu_cxx::encoding_char_traits&lt;_CharT&gt;</code> Struct Template Reference	843
5.45.1	Detailed Description . . . . .	845
5.46	<code>__gnu_cxx::encoding_state</code> Class Reference . . . . .	845
5.46.1	Detailed Description . . . . .	846
5.47	<code>__gnu_cxx::forced_error</code> Struct Reference . . . . .	846
5.47.1	Detailed Description . . . . .	847

---

---

5.47.2	Member Function Documentation . . . . .	847
5.47.2.1	what . . . . .	847
5.48	<code>__gnu_cxx::free_list</code> Class Reference . . . . .	848
5.48.1	Detailed Description . . . . .	848
5.48.2	Member Function Documentation . . . . .	849
5.48.2.1	<code>_M_clear</code> . . . . .	849
5.48.2.2	<code>_M_get</code> . . . . .	849
5.48.2.3	<code>_M_insert</code> . . . . .	849
5.49	<code>__gnu_cxx::hash_map&lt;_Key, _Tp, _HashFn, _EqualKey, _Alloc &gt;</code> Class Template Reference . . . . .	849
5.49.1	Detailed Description . . . . .	851
5.50	<code>__gnu_cxx::hash_multimap&lt;_Key, _Tp, _HashFn, _EqualKey, _-</code> <code>Alloc &gt;</code> Class Template Reference . . . . .	852
5.50.1	Detailed Description . . . . .	853
5.51	<code>__gnu_cxx::hash_multiset&lt;_Value, _HashFcn, _EqualKey, _Alloc &gt;</code> Class Template Reference . . . . .	854
5.51.1	Detailed Description . . . . .	855
5.52	<code>__gnu_cxx::hash_set&lt;_Value, _HashFcn, _EqualKey, _Alloc &gt;</code> Class Template Reference . . . . .	856
5.52.1	Detailed Description . . . . .	857
5.53	<code>__gnu_cxx::limit_condition</code> Struct Reference . . . . .	858
5.53.1	Detailed Description . . . . .	859
5.54	<code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference . . . . .	859
5.54.1	Detailed Description . . . . .	859
5.55	<code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference . . . . .	859
5.55.1	Detailed Description . . . . .	859
5.56	<code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference . . . . .	860
5.56.1	Detailed Description . . . . .	860
5.57	<code>__gnu_cxx::malloc_allocator&lt;_Tp &gt;</code> Class Template Reference . . . . .	860
5.57.1	Detailed Description . . . . .	861
5.58	<code>__gnu_cxx::new_allocator&lt;_Tp &gt;</code> Class Template Reference . . . . .	861
5.58.1	Detailed Description . . . . .	863

---

5.59	<a href="#">__gnu_cxx::project1st&lt; _Arg1, _Arg2 &gt; Struct Template Reference</a>	863
5.59.1	<a href="#">Detailed Description</a>	864
5.59.2	<a href="#">Member Typedef Documentation</a>	864
5.59.2.1	<a href="#">first_argument_type</a>	864
5.59.2.2	<a href="#">result_type</a>	864
5.59.2.3	<a href="#">second_argument_type</a>	864
5.60	<a href="#">__gnu_cxx::project2nd&lt; _Arg1, _Arg2 &gt; Struct Template Reference</a>	864
5.60.1	<a href="#">Detailed Description</a>	865
5.60.2	<a href="#">Member Typedef Documentation</a>	865
5.60.2.1	<a href="#">first_argument_type</a>	865
5.60.2.2	<a href="#">result_type</a>	865
5.60.2.3	<a href="#">second_argument_type</a>	865
5.61	<a href="#">__gnu_cxx::random_condition Struct Reference</a>	866
5.61.1	<a href="#">Detailed Description</a>	867
5.62	<a href="#">__gnu_cxx::random_condition::always_adjustor Struct Reference</a>	867
5.62.1	<a href="#">Detailed Description</a>	867
5.63	<a href="#">__gnu_cxx::random_condition::group_adjustor Struct Reference</a>	867
5.63.1	<a href="#">Detailed Description</a>	867
5.64	<a href="#">__gnu_cxx::random_condition::never_adjustor Struct Reference</a>	868
5.64.1	<a href="#">Detailed Description</a>	868
5.65	<a href="#">__gnu_cxx::rb_tree&lt; _Key, _Value, _KeyOfValue, _Compare, _Alloc &gt; Struct Template Reference</a>	868
5.65.1	<a href="#">Detailed Description</a>	871
5.66	<a href="#">__gnu_cxx::recursive_init_error Class Reference</a>	871
5.66.1	<a href="#">Detailed Description</a>	872
5.66.2	<a href="#">Member Function Documentation</a>	872
5.66.2.1	<a href="#">what</a>	872
5.67	<a href="#">__gnu_cxx::rope&lt; _CharT, _Alloc &gt; Class Template Reference</a>	873
5.67.1	<a href="#">Detailed Description</a>	879
5.68	<a href="#">__gnu_cxx::select1st&lt; _Pair &gt; Struct Template Reference</a>	879
5.68.1	<a href="#">Detailed Description</a>	880

---

---

5.68.2	Member Typedef Documentation	880
5.68.2.1	argument_type	880
5.68.2.2	result_type	880
5.69	<code>__gnu_cxx::select2nd&lt;_Pair&gt;</code> Struct Template Reference	880
5.69.1	Detailed Description	881
5.69.2	Member Typedef Documentation	881
5.69.2.1	argument_type	881
5.69.2.2	result_type	881
5.70	<code>__gnu_cxx::slist&lt;_Tp, _Alloc&gt;</code> Class Template Reference	881
5.70.1	Detailed Description	884
5.71	<code>__gnu_cxx::stdio_filebuf&lt;_CharT, _Traits&gt;</code> Class Template Reference	884
5.71.1	Detailed Description	889
5.71.2	Member Typedef Documentation	889
5.71.2.1	<code>__streambuf_type</code>	889
5.71.2.2	<code>char_type</code>	889
5.71.2.3	<code>int_type</code>	889
5.71.2.4	<code>off_type</code>	890
5.71.2.5	<code>pos_type</code>	890
5.71.2.6	<code>traits_type</code>	890
5.71.3	Constructor & Destructor Documentation	890
5.71.3.1	<code>stdio_filebuf</code>	890
5.71.3.2	<code>stdio_filebuf</code>	891
5.71.3.3	<code>stdio_filebuf</code>	891
5.71.3.4	<code>~stdio_filebuf</code>	892
5.71.4	Member Function Documentation	892
5.71.4.1	<code>_M_create_pback</code>	892
5.71.4.2	<code>_M_destroy_pback</code>	892
5.71.4.3	<code>_M_set_buffer</code>	892
5.71.4.4	<code>close</code>	893
5.71.4.5	<code>eback</code>	893
5.71.4.6	<code>egptr</code>	893

---

5.71.4.7	epptr	894
5.71.4.8	fd	894
5.71.4.9	file	895
5.71.4.10	gbump	895
5.71.4.11	getloc	895
5.71.4.12	gptr	895
5.71.4.13	imbue	896
5.71.4.14	in_avail	897
5.71.4.15	is_open	897
5.71.4.16	open	897
5.71.4.17	open	898
5.71.4.18	overflow	898
5.71.4.19	pbackfail	899
5.71.4.20	pbase	900
5.71.4.21	pbump	900
5.71.4.22	pptr	900
5.71.4.23	pubimbue	901
5.71.4.24	pubseekoff	901
5.71.4.25	pubseekpos	902
5.71.4.26	pubsetbuf	902
5.71.4.27	pubsync	902
5.71.4.28	sbumpc	903
5.71.4.29	seekoff	903
5.71.4.30	seekpos	903
5.71.4.31	setbuf	904
5.71.4.32	setbuf	904
5.71.4.33	setg	905
5.71.4.34	setp	905
5.71.4.35	sgetc	905
5.71.4.36	sgetn	906
5.71.4.37	showmanyc	906

---

5.71.4.38	snextc	907
5.71.4.39	sputbackc	907
5.71.4.40	sputc	908
5.71.4.41	sputn	908
5.71.4.42	stossc	908
5.71.4.43	sungetc	909
5.71.4.44	sync	909
5.71.4.45	uflow	910
5.71.4.46	underflow	910
5.71.4.47	xsgetn	911
5.71.4.48	xsputn	911
5.71.5	Member Data Documentation	912
5.71.5.1	_M_buf	912
5.71.5.2	_M_buf_locale	912
5.71.5.3	_M_buf_size	912
5.71.5.4	_M_ext_buf	913
5.71.5.5	_M_ext_buf_size	913
5.71.5.6	_M_ext_next	913
5.71.5.7	_M_in_beg	913
5.71.5.8	_M_in_cur	914
5.71.5.9	_M_in_end	914
5.71.5.10	_M_mode	914
5.71.5.11	_M_out_beg	915
5.71.5.12	_M_out_cur	915
5.71.5.13	_M_out_end	915
5.71.5.14	_M_pback	916
5.71.5.15	_M_pback_cur_save	916
5.71.5.16	_M_pback_end_save	916
5.71.5.17	_M_pback_init	917
5.71.5.18	_M_reading	917

---

5.72	<code>__gnu_cxx::stdio_sync_filebuf&lt;_CharT, _Traits &gt;</code> Class Template Reference	917
5.72.1	Detailed Description	921
5.72.2	Member Typedef Documentation	921
5.72.2.1	<code>__streambuf_type</code>	921
5.72.2.2	<code>char_type</code>	922
5.72.2.3	<code>int_type</code>	922
5.72.2.4	<code>off_type</code>	922
5.72.2.5	<code>pos_type</code>	922
5.72.2.6	<code>traits_type</code>	923
5.72.3	Member Function Documentation	923
5.72.3.1	<code>eback</code>	923
5.72.3.2	<code>egptr</code>	923
5.72.3.3	<code>eptr</code>	924
5.72.3.4	<code>file</code>	924
5.72.3.5	<code>gbump</code>	924
5.72.3.6	<code>getloc</code>	925
5.72.3.7	<code>gptr</code>	925
5.72.3.8	<code>imbue</code>	925
5.72.3.9	<code>in_avail</code>	926
5.72.3.10	<code>overflow</code>	926
5.72.3.11	<code>pbackfail</code>	927
5.72.3.12	<code>pbase</code>	928
5.72.3.13	<code>pbump</code>	928
5.72.3.14	<code>pptr</code>	928
5.72.3.15	<code>pubimbue</code>	929
5.72.3.16	<code>pubseekoff</code>	929
5.72.3.17	<code>pubseekpos</code>	930
5.72.3.18	<code>pubsetbuf</code>	930
5.72.3.19	<code>pubsync</code>	930
5.72.3.20	<code>sbumpc</code>	931

---

---

5.72.3.21	seekoff	931
5.72.3.22	seekpos	931
5.72.3.23	setbuf	932
5.72.3.24	setg	932
5.72.3.25	setp	932
5.72.3.26	sgetc	933
5.72.3.27	sgetn	933
5.72.3.28	showmanyc	934
5.72.3.29	snextc	934
5.72.3.30	sputbackc	935
5.72.3.31	sputc	935
5.72.3.32	sputn	936
5.72.3.33	stossc	936
5.72.3.34	sungetc	936
5.72.3.35	sync	937
5.72.3.36	uflow	937
5.72.3.37	underflow	937
5.72.3.38	xsgetn	938
5.72.3.39	xsputn	939
5.72.4	Member Data Documentation	939
5.72.4.1	_M_buf_locale	939
5.72.4.2	_M_in_beg	939
5.72.4.3	_M_in_cur	940
5.72.4.4	_M_in_end	940
5.72.4.5	_M_out_beg	940
5.72.4.6	_M_out_cur	941
5.72.4.7	_M_out_end	941
5.73	__gnu_cxx::subtractive_rng Class Reference	942
5.73.1	Detailed Description	942
5.73.2	Member Typedef Documentation	943
5.73.2.1	argument_type	943

---

---

5.73.2.2	result_type . . . . .	943
5.73.3	Constructor & Destructor Documentation . . . . .	943
5.73.3.1	subtractive_rng . . . . .	943
5.73.3.2	subtractive_rng . . . . .	943
5.73.4	Member Function Documentation . . . . .	943
5.73.4.1	operator() . . . . .	943
5.74	<code>__gnu_cxx::temporary_buffer&lt;_ForwardIterator, _Tp&gt;</code> Struct Template Reference . . . . .	944
5.74.1	Detailed Description . . . . .	945
5.74.2	Constructor & Destructor Documentation . . . . .	945
5.74.2.1	temporary_buffer . . . . .	945
5.74.2.2	~temporary_buffer . . . . .	945
5.74.3	Member Function Documentation . . . . .	946
5.74.3.1	begin . . . . .	946
5.74.3.2	end . . . . .	946
5.74.3.3	requested_size . . . . .	946
5.74.3.4	size . . . . .	946
5.75	<code>__gnu_cxx::throw_allocator_base&lt;_Tp, _Cond&gt;</code> Class Template Reference . . . . .	947
5.75.1	Detailed Description . . . . .	948
5.76	<code>__gnu_cxx::throw_allocator_limit&lt;_Tp&gt;</code> Struct Template Reference . . . . .	948
5.76.1	Detailed Description . . . . .	949
5.77	<code>__gnu_cxx::throw_allocator_random&lt;_Tp&gt;</code> Struct Template Reference . . . . .	950
5.77.1	Detailed Description . . . . .	951
5.78	<code>__gnu_cxx::throw_value_base&lt;_Cond&gt;</code> Struct Template Reference . . . . .	951
5.78.1	Detailed Description . . . . .	952
5.79	<code>__gnu_cxx::throw_value_limit</code> Struct Reference . . . . .	952
5.79.1	Detailed Description . . . . .	954
5.80	<code>__gnu_cxx::throw_value_random</code> Struct Reference . . . . .	954
5.80.1	Detailed Description . . . . .	955
5.81	<code>__gnu_cxx::unary_compose&lt;_Operation1, _Operation2&gt;</code> Class Template Reference . . . . .	955

---

---

5.81.1	Detailed Description . . . . .	956
5.81.2	Member Typedef Documentation . . . . .	957
5.81.2.1	argument_type . . . . .	957
5.81.2.2	result_type . . . . .	957
5.82	__gnu_debug::__is_same<_Type1, _Type2 > Struct Template Reference . . . . .	957
5.82.1	Detailed Description . . . . .	957
5.83	__gnu_debug::_After_nth_from<_Iterator > Class Template Reference . . . . .	958
5.83.1	Detailed Description . . . . .	958
5.84	__gnu_debug::_Not_equal_to<_Type > Class Template Reference . . . . .	958
5.84.1	Detailed Description . . . . .	958
5.85	__gnu_debug::_Safe_iterator<_Iterator, _Sequence > Class Template Reference . . . . .	959
5.85.1	Detailed Description . . . . .	961
5.85.2	Constructor & Destructor Documentation . . . . .	961
5.85.2.1	_Safe_iterator . . . . .	961
5.85.2.2	_Safe_iterator . . . . .	962
5.85.2.3	_Safe_iterator . . . . .	962
5.85.2.4	_Safe_iterator . . . . .	962
5.85.3	Member Function Documentation . . . . .	963
5.85.3.1	_M_attach . . . . .	963
5.85.3.2	_M_attach . . . . .	963
5.85.3.3	_M_attach_single . . . . .	963
5.85.3.4	_M_attach_single . . . . .	963
5.85.3.5	_M_attached_to . . . . .	963
5.85.3.6	_M_can_compare . . . . .	964
5.85.3.7	_M_dereferenceable . . . . .	964
5.85.3.8	_M_detach . . . . .	964
5.85.3.9	_M_detach_single . . . . .	964
5.85.3.10	_M_get_distance . . . . .	964
5.85.3.11	_M_get_mutex . . . . .	965
5.85.3.12	_M_incrementable . . . . .	965

---

---

5.85.3.13	<code>_M_invalidate</code>	965
5.85.3.14	<code>_M_invalidate_single</code>	965
5.85.3.15	<code>_M_is_begin</code>	966
5.85.3.16	<code>_M_is_end</code>	966
5.85.3.17	<code>_M_singular</code>	966
5.85.3.18	<code>base</code>	966
5.85.3.19	<code>operator_iterator</code>	967
5.85.3.20	<code>operator*</code>	967
5.85.3.21	<code>operator++</code>	967
5.85.3.22	<code>operator++</code>	968
5.85.3.23	<code>operator--</code>	968
5.85.3.24	<code>operator--</code>	968
5.85.3.25	<code>operator-&gt;</code>	968
5.85.3.26	<code>operator=</code>	969
5.85.4	Member Data Documentation	969
5.85.4.1	<code>_M_next</code>	969
5.85.4.2	<code>_M_prior</code>	969
5.85.4.3	<code>_M_sequence</code>	970
5.85.4.4	<code>_M_version</code>	970
5.86	<code>__gnu_debug::_Safe_iterator_base</code> Class Reference	970
5.86.1	Detailed Description	972
5.86.2	Constructor & Destructor Documentation	972
5.86.2.1	<code>_Safe_iterator_base</code>	972
5.86.2.2	<code>_Safe_iterator_base</code>	972
5.86.2.3	<code>_Safe_iterator_base</code>	972
5.86.3	Member Function Documentation	973
5.86.3.1	<code>_M_attach</code>	973
5.86.3.2	<code>_M_attach_single</code>	973
5.86.3.3	<code>_M_attached_to</code>	973
5.86.3.4	<code>_M_can_compare</code>	973
5.86.3.5	<code>_M_detach</code>	973

---

---

5.86.3.6	<code>_M_detach_single</code>	973
5.86.3.7	<code>_M_get_mutex</code>	974
5.86.3.8	<code>_M_singular</code>	974
5.86.4	Member Data Documentation	974
5.86.4.1	<code>_M_next</code>	974
5.86.4.2	<code>_M_prior</code>	974
5.86.4.3	<code>_M_sequence</code>	974
5.86.4.4	<code>_M_version</code>	975
5.87	<code>__gnu_debug::_Safe_sequence&lt;_Sequence &gt;</code> Class Template Reference	975
5.87.1	Detailed Description	976
5.87.2	Member Function Documentation	977
5.87.2.1	<code>_M_detach_all</code>	977
5.87.2.2	<code>_M_detach_singular</code>	977
5.87.2.3	<code>_M_get_mutex</code>	977
5.87.2.4	<code>_M_invalidate_all</code>	977
5.87.2.5	<code>_M_invalidate_if</code>	977
5.87.2.6	<code>_M_revalidate_singular</code>	978
5.87.2.7	<code>_M_swap</code>	978
5.87.2.8	<code>_M_transfer_iter</code>	978
5.87.3	Member Data Documentation	978
5.87.3.1	<code>_M_const_iterators</code>	978
5.87.3.2	<code>_M_iterators</code>	978
5.87.3.3	<code>_M_version</code>	979
5.88	<code>__gnu_debug::_Safe_sequence_base</code> Class Reference	979
5.88.1	Detailed Description	980
5.88.2	Constructor & Destructor Documentation	980
5.88.2.1	<code>~_Safe_sequence_base</code>	980
5.88.3	Member Function Documentation	981
5.88.3.1	<code>_M_detach_all</code>	981
5.88.3.2	<code>_M_detach_singular</code>	981

---

5.88.3.3	<code>_M_get_mutex</code>	981
5.88.3.4	<code>_M_invalidate_all</code>	981
5.88.3.5	<code>_M_revalidate_singular</code>	981
5.88.3.6	<code>_M_swap</code>	981
5.88.4	Member Data Documentation	982
5.88.4.1	<code>_M_const_iterators</code>	982
5.88.4.2	<code>_M_iterators</code>	982
5.88.4.3	<code>_M_version</code>	982
5.89	<code>__gnu_debug::basic_string&lt; _CharT, _Traits, _Allocator &gt; Class</code>	
	Template Reference	982
5.89.1	Detailed Description	990
5.89.2	Member Function Documentation	991
5.89.2.1	<code>_M_detach_all</code>	991
5.89.2.2	<code>_M_detach_singular</code>	991
5.89.2.3	<code>_M_get_mutex</code>	991
5.89.2.4	<code>_M_invalidate_all</code>	991
5.89.2.5	<code>_M_invalidate_if</code>	991
5.89.2.6	<code>_M_revalidate_singular</code>	992
5.89.2.7	<code>_M_swap</code>	992
5.89.2.8	<code>_M_transfer_iter</code>	992
5.89.2.9	<code>append</code>	992
5.89.2.10	<code>append</code>	992
5.89.2.11	<code>append</code>	993
5.89.2.12	<code>append</code>	993
5.89.2.13	<code>append</code>	994
5.89.2.14	<code>append</code>	994
5.89.2.15	<code>append</code>	994
5.89.2.16	<code>assign</code>	995
5.89.2.17	<code>assign</code>	995
5.89.2.18	<code>assign</code>	995
5.89.2.19	<code>assign</code>	996

---

---

5.89.2.20 assign . . . . .	996
5.89.2.21 assign . . . . .	997
5.89.2.22 assign . . . . .	997
5.89.2.23 assign . . . . .	997
5.89.2.24 at . . . . .	998
5.89.2.25 at . . . . .	998
5.89.2.26 begin . . . . .	999
5.89.2.27 begin . . . . .	999
5.89.2.28 c_str . . . . .	999
5.89.2.29 capacity . . . . .	999
5.89.2.30 cbegin . . . . .	999
5.89.2.31 cend . . . . .	999
5.89.2.32 clear . . . . .	1000
5.89.2.33 compare . . . . .	1000
5.89.2.34 compare . . . . .	1000
5.89.2.35 compare . . . . .	1001
5.89.2.36 compare . . . . .	1001
5.89.2.37 compare . . . . .	1002
5.89.2.38 compare . . . . .	1002
5.89.2.39 copy . . . . .	1003
5.89.2.40 crbegin . . . . .	1003
5.89.2.41 crend . . . . .	1004
5.89.2.42 data . . . . .	1004
5.89.2.43 empty . . . . .	1004
5.89.2.44 end . . . . .	1004
5.89.2.45 end . . . . .	1004
5.89.2.46 erase . . . . .	1004
5.89.2.47 erase . . . . .	1005
5.89.2.48 erase . . . . .	1005
5.89.2.49 find . . . . .	1006
5.89.2.50 find . . . . .	1006

---

5.89.2.51 find . . . . .	1007
5.89.2.52 find . . . . .	1007
5.89.2.53 find_first_not_of . . . . .	1007
5.89.2.54 find_first_not_of . . . . .	1008
5.89.2.55 find_first_not_of . . . . .	1008
5.89.2.56 find_first_not_of . . . . .	1009
5.89.2.57 find_first_of . . . . .	1009
5.89.2.58 find_first_of . . . . .	1009
5.89.2.59 find_first_of . . . . .	1010
5.89.2.60 find_first_of . . . . .	1010
5.89.2.61 find_last_not_of . . . . .	1011
5.89.2.62 find_last_not_of . . . . .	1011
5.89.2.63 find_last_not_of . . . . .	1012
5.89.2.64 find_last_not_of . . . . .	1012
5.89.2.65 find_last_of . . . . .	1012
5.89.2.66 find_last_of . . . . .	1013
5.89.2.67 find_last_of . . . . .	1013
5.89.2.68 find_last_of . . . . .	1014
5.89.2.69 get_allocator . . . . .	1014
5.89.2.70 insert . . . . .	1014
5.89.2.71 insert . . . . .	1015
5.89.2.72 insert . . . . .	1015
5.89.2.73 insert . . . . .	1016
5.89.2.74 insert . . . . .	1016
5.89.2.75 insert . . . . .	1017
5.89.2.76 insert . . . . .	1017
5.89.2.77 insert . . . . .	1018
5.89.2.78 insert . . . . .	1018
5.89.2.79 length . . . . .	1019
5.89.2.80 max_size . . . . .	1019
5.89.2.81 operator+= . . . . .	1019

---

5.89.2.82 operator+=	1020
5.89.2.83 operator+=	1020
5.89.2.84 operator+=	1020
5.89.2.85 operator[]	1021
5.89.2.86 operator[]	1021
5.89.2.87 push_back	1021
5.89.2.88 rbegin	1022
5.89.2.89 rbegin	1022
5.89.2.90 rend	1022
5.89.2.91 rend	1022
5.89.2.92 replace	1022
5.89.2.93 replace	1023
5.89.2.94 replace	1023
5.89.2.95 replace	1024
5.89.2.96 replace	1025
5.89.2.97 replace	1025
5.89.2.98 replace	1026
5.89.2.99 replace	1026
5.89.2.100replace	1027
5.89.2.101replace	1028
5.89.2.102replace	1028
5.89.2.103reserve	1029
5.89.2.104resize	1029
5.89.2.105resize	1030
5.89.2.106rfind	1030
5.89.2.107rfind	1030
5.89.2.108rfind	1031
5.89.2.109rfind	1031
5.89.2.110shrink_to_fit	1032
5.89.2.111size	1032
5.89.2.112substr	1032

---

5.89.2.113swap	1032
5.89.3 Member Data Documentation	1033
5.89.3.1 <code>_M_const_iterators</code>	1033
5.89.3.2 <code>_M_iterators</code>	1033
5.89.3.3 <code>_M_version</code>	1033
5.89.3.4 <code>npos</code>	1033
5.90 <code>__gnu_parallel::__accumulate_binop_reduct&lt;_BinOp&gt;</code> Struct Template Reference	1034
5.90.1 Detailed Description	1034
5.91 <code>__gnu_parallel::__accumulate_selector&lt;_It&gt;</code> Struct Template Reference	1034
5.91.1 Detailed Description	1035
5.91.2 Member Function Documentation	1035
5.91.2.1 <code>operator()</code>	1035
5.91.3 Member Data Documentation	1036
5.91.3.1 <code>_M_finish_iterator</code>	1036
5.92 <code>__gnu_parallel::__adjacent_difference_selector&lt;_It&gt;</code> Struct Template Reference	1036
5.92.1 Detailed Description	1037
5.92.2 Member Data Documentation	1037
5.92.2.1 <code>_M_finish_iterator</code>	1037
5.93 <code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	1038
5.93.1 Detailed Description	1038
5.93.2 Member Function Documentation	1039
5.93.2.1 <code>_M_sequential_algorithm</code>	1039
5.93.2.2 <code>operator()</code>	1039
5.94 <code>__gnu_parallel::__binder1st&lt;_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType&gt;</code> Class Template Reference	1039
5.94.1 Detailed Description	1040
5.94.2 Member Typedef Documentation	1041
5.94.2.1 <code>argument_type</code>	1041
5.94.2.2 <code>result_type</code>	1041

---

---

5.95	<code>__gnu_parallel::__binder2nd&lt; _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType &gt;</code> Class Template Reference . . .	1041
5.95.1	Detailed Description . . . . .	1042
5.95.2	Member Typedef Documentation . . . . .	1043
5.95.2.1	<code>argument_type</code> . . . . .	1043
5.95.2.2	<code>result_type</code> . . . . .	1043
5.96	<code>__gnu_parallel::__count_if_selector&lt; _It, _Diff &gt;</code> Struct Template Reference . . . . .	1043
5.96.1	Detailed Description . . . . .	1044
5.96.2	Member Function Documentation . . . . .	1044
5.96.2.1	<code>operator()</code> . . . . .	1044
5.96.3	Member Data Documentation . . . . .	1044
5.96.3.1	<code>_M_finish_iterator</code> . . . . .	1044
5.97	<code>__gnu_parallel::__count_selector&lt; _It, _Diff &gt;</code> Struct Template Reference . . . . .	1045
5.97.1	Detailed Description . . . . .	1046
5.97.2	Member Function Documentation . . . . .	1046
5.97.2.1	<code>operator()</code> . . . . .	1046
5.97.3	Member Data Documentation . . . . .	1046
5.97.3.1	<code>_M_finish_iterator</code> . . . . .	1046
5.98	<code>__gnu_parallel::__fill_selector&lt; _It &gt;</code> Struct Template Reference . . .	1047
5.98.1	Detailed Description . . . . .	1047
5.98.2	Member Function Documentation . . . . .	1048
5.98.2.1	<code>operator()</code> . . . . .	1048
5.98.3	Member Data Documentation . . . . .	1048
5.98.3.1	<code>_M_finish_iterator</code> . . . . .	1048
5.99	<code>__gnu_parallel::__find_first_of_selector&lt; _FIterator &gt;</code> Struct Template Reference . . . . .	1048
5.99.1	Detailed Description . . . . .	1049
5.99.2	Member Function Documentation . . . . .	1050
5.99.2.1	<code>_M_sequential_algorithm</code> . . . . .	1050
5.99.2.2	<code>operator()</code> . . . . .	1050

---

---

5.100	<code>__gnu_parallel::__find_if_selector</code> Struct Reference . . . . .	1050
5.100.1	Detailed Description . . . . .	1051
5.100.2	Member Function Documentation . . . . .	1051
5.100.2.1	<code>_M_sequential_algorithm</code> . . . . .	1051
5.100.2.2	<code>operator()</code> . . . . .	1052
5.101	<code>__gnu_parallel::__for_each_selector&lt;_It&gt;</code> Struct Template Reference	1052
5.101.1	Detailed Description . . . . .	1053
5.101.2	Member Function Documentation . . . . .	1053
5.101.2.1	<code>operator()</code> . . . . .	1053
5.101.3	Member Data Documentation . . . . .	1054
5.101.3.1	<code>_M_finish_iterator</code> . . . . .	1054
5.102	<code>__gnu_parallel::__generate_selector&lt;_It&gt;</code> Struct Template Reference	1054
5.102.1	Detailed Description . . . . .	1055
5.102.2	Member Function Documentation . . . . .	1055
5.102.2.1	<code>operator()</code> . . . . .	1055
5.102.3	Member Data Documentation . . . . .	1055
5.102.3.1	<code>_M_finish_iterator</code> . . . . .	1055
5.103	<code>__gnu_parallel::__generic_find_selector</code> Struct Reference . . . . .	1056
5.103.1	Detailed Description . . . . .	1056
5.104	<code>__gnu_parallel::__generic_for_each_selector&lt;_It&gt;</code> Struct Template Reference . . . . .	1056
5.104.1	Detailed Description . . . . .	1057
5.104.2	Member Data Documentation . . . . .	1058
5.104.2.1	<code>_M_finish_iterator</code> . . . . .	1058
5.105	<code>__gnu_parallel::__identity_selector&lt;_It&gt;</code> Struct Template Reference	1058
5.105.1	Detailed Description . . . . .	1059
5.105.2	Member Function Documentation . . . . .	1059
5.105.2.1	<code>operator()</code> . . . . .	1059
5.105.3	Member Data Documentation . . . . .	1059
5.105.3.1	<code>_M_finish_iterator</code> . . . . .	1059
5.106	<code>__gnu_parallel::__inner_product_selector&lt;_It, _It2, _Tp&gt;</code> Struct Template Reference . . . . .	1060

---

5.106.1 Detailed Description . . . . .	1060
5.106.2 Constructor & Destructor Documentation . . . . .	1061
5.106.2.1 __inner_product_selector . . . . .	1061
5.106.3 Member Function Documentation . . . . .	1061
5.106.3.1 operator() . . . . .	1061
5.106.4 Member Data Documentation . . . . .	1061
5.106.4.1 __begin1_iterator . . . . .	1061
5.106.4.2 __begin2_iterator . . . . .	1062
5.106.4.3 _M_finish_iterator . . . . .	1062
5.107 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference . . . . .	1062
5.107.1 Detailed Description . . . . .	1063
5.108 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference . . . . .	1063
5.108.1 Detailed Description . . . . .	1063
5.109 __gnu_parallel::__mismatch_selector Struct Reference . . . . .	1064
5.109.1 Detailed Description . . . . .	1064
5.109.2 Member Function Documentation . . . . .	1065
5.109.2.1 _M_sequential_algorithm . . . . .	1065
5.109.2.2 operator() . . . . .	1065
5.110 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference . . . . .	1065
5.110.1 Detailed Description . . . . .	1066
5.111 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference . . . . .	1066
5.111.1 Detailed Description . . . . .	1066
5.112 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference . . . . .	1067
5.112.1 Detailed Description . . . . .	1067

5.113	<code>_gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt; true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt; Struct Template Reference</code>	1067
5.113.1	Detailed Description	1068
5.114	<code>_gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt; Struct Template Reference</code>	1068
5.114.1	Detailed Description	1069
5.115	<code>_gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt; Struct Template Reference</code>	1069
5.115.1	Detailed Description	1069
5.116	<code>_gnu_parallel::__replace_if_selector&lt; _It, _Op, _Tp &gt; Struct Template Reference</code>	1070
5.116.1	Detailed Description	1070
5.116.2	Constructor & Destructor Documentation	1071
5.116.2.1	<code>__replace_if_selector</code>	1071
5.116.3	Member Function Documentation	1071
5.116.3.1	<code>operator()</code>	1071
5.116.4	Member Data Documentation	1071
5.116.4.1	<code>__new_val</code>	1071
5.116.4.2	<code>_M_finish_iterator</code>	1072
5.117	<code>_gnu_parallel::__replace_selector&lt; _It, _Tp &gt; Struct Template Reference</code>	1072
5.117.1	Detailed Description	1073
5.117.2	Constructor & Destructor Documentation	1073
5.117.2.1	<code>__replace_selector</code>	1073
5.117.3	Member Function Documentation	1073
5.117.3.1	<code>operator()</code>	1073
5.117.4	Member Data Documentation	1073
5.117.4.1	<code>__new_val</code>	1073
5.117.4.2	<code>_M_finish_iterator</code>	1074
5.118	<code>_gnu_parallel::__transform1_selector&lt; _It &gt; Struct Template Reference</code>	1074

---

5.118.1 Detailed Description . . . . .	1075
5.118.2 Member Function Documentation . . . . .	1075
5.118.2.1 operator() . . . . .	1075
5.118.3 Member Data Documentation . . . . .	1075
5.118.3.1 _M_finish_iterator . . . . .	1075
5.119 __gnu_parallel::__transform2_selector< _It > Struct Template Reference . . . . .	1076
5.119.1 Detailed Description . . . . .	1076
5.119.2 Member Function Documentation . . . . .	1077
5.119.2.1 operator() . . . . .	1077
5.119.3 Member Data Documentation . . . . .	1077
5.119.3.1 _M_finish_iterator . . . . .	1077
5.120 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference . . . . .	1077
5.120.1 Detailed Description . . . . .	1078
5.120.2 Member Typedef Documentation . . . . .	1079
5.120.2.1 argument_type . . . . .	1079
5.120.2.2 result_type . . . . .	1079
5.121 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference . . . . .	1079
5.121.1 Detailed Description . . . . .	1080
5.121.2 Constructor & Destructor Documentation . . . . .	1080
5.121.2.1 _DRandomShufflingGlobalData . . . . .	1080
5.121.3 Member Data Documentation . . . . .	1080
5.121.3.1 _M_bin_proc . . . . .	1080
5.121.3.2 _M_dist . . . . .	1081
5.121.3.3 _M_num_bins . . . . .	1081
5.121.3.4 _M_num_bits . . . . .	1081
5.121.3.5 _M_source . . . . .	1081
5.121.3.6 _M_starts . . . . .	1082
5.121.3.7 _M_temporaries . . . . .	1082

---

5.122__gnu_parallel::_DRSSorterPU< _RAIter, RandomNumberGenerator > Struct Template Reference . . . . .	1082
5.122.1 Detailed Description . . . . .	1083
5.122.2 Member Data Documentation . . . . .	1083
5.122.2.1 __bins_end . . . . .	1083
5.122.2.2 _M_bins_begin . . . . .	1083
5.122.2.3 _M_num_threads . . . . .	1083
5.122.2.4 _M_sd . . . . .	1084
5.122.2.5 _M_seed . . . . .	1084
5.123__gnu_parallel::_DummyReduct Struct Reference . . . . .	1084
5.123.1 Detailed Description . . . . .	1084
5.124__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference . . . . .	1085
5.124.1 Detailed Description . . . . .	1086
5.124.2 Member Typedef Documentation . . . . .	1086
5.124.2.1 first_argument_type . . . . .	1086
5.124.2.2 result_type . . . . .	1086
5.124.2.3 second_argument_type . . . . .	1086
5.125__gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference . . . . .	1086
5.125.1 Detailed Description . . . . .	1087
5.125.2 Member Typedef Documentation . . . . .	1088
5.125.2.1 first_argument_type . . . . .	1088
5.125.2.2 result_type . . . . .	1088
5.125.2.3 second_argument_type . . . . .	1088
5.126__gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference . . . . .	1088
5.126.1 Detailed Description . . . . .	1089
5.126.2 Constructor & Destructor Documentation . . . . .	1089
5.126.2.1 _GuardedIterator . . . . .	1089
5.126.3 Member Function Documentation . . . . .	1089
5.126.3.1 operator _RAIter . . . . .	1089
5.126.3.2 operator* . . . . .	1090

---

5.126.3.3 operator++	1090
5.126.4 Friends And Related Function Documentation	1090
5.126.4.1 operator<	1090
5.126.4.2 operator<=	1091
5.127__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference	1091
5.127.1 Detailed Description	1093
5.127.2 Member Typedef Documentation	1093
5.127.2.1 first_type	1093
5.127.2.2 second_type	1093
5.127.3 Member Data Documentation	1093
5.127.3.1 first	1093
5.127.3.2 second	1094
5.128__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference	1094
5.128.1 Detailed Description	1095
5.129__gnu_parallel::_Job< _DifferenceTp > Struct Template Reference	1095
5.129.1 Detailed Description	1095
5.129.2 Member Data Documentation	1096
5.129.2.1 _M_first	1096
5.129.2.2 _M_last	1096
5.129.2.3 _M_load	1096
5.130__gnu_parallel::_Less< _T1, _T2 > Struct Template Reference	1096
5.130.1 Detailed Description	1097
5.130.2 Member Typedef Documentation	1098
5.130.2.1 first_argument_type	1098
5.130.2.2 result_type	1098
5.130.2.3 second_argument_type	1098
5.131__gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference	1098
5.131.1 Detailed Description	1099
5.131.2 Member Typedef Documentation	1100

---

---

5.131.2.1	first_argument_type . . . . .	1100
5.131.2.2	result_type . . . . .	1100
5.131.2.3	second_argument_type . . . . .	1100
5.132	<code>__gnu_parallel::_LexicographicReverse&lt; _T1, _T2, _Compare &gt;</code> Class Template Reference . . . . .	1100
5.132.1	Detailed Description . . . . .	1101
5.132.2	Member Typedef Documentation . . . . .	1102
5.132.2.1	first_argument_type . . . . .	1102
5.132.2.2	result_type . . . . .	1102
5.132.2.3	second_argument_type . . . . .	1102
5.133	<code>__gnu_parallel::_LoserTree&lt; __stable, _Tp, _Compare &gt;</code> Class Tem- plate Reference . . . . .	1102
5.133.1	Detailed Description . . . . .	1103
5.133.2	Member Function Documentation . . . . .	1104
5.133.2.1	<code>__delete_min_insert</code> . . . . .	1104
5.133.2.2	<code>__get_min_source</code> . . . . .	1104
5.133.2.3	<code>__insert_start</code> . . . . .	1104
5.133.3	Member Data Documentation . . . . .	1105
5.133.3.1	<code>_M_comp</code> . . . . .	1105
5.133.3.2	<code>_M_first_insert</code> . . . . .	1105
5.133.3.3	<code>_M_log_k</code> . . . . .	1105
5.133.3.4	<code>_M_losers</code> . . . . .	1105
5.134	<code>__gnu_parallel::_LoserTree&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1106
5.134.1	Detailed Description . . . . .	1107
5.134.2	Member Function Documentation . . . . .	1107
5.134.2.1	<code>__delete_min_insert</code> . . . . .	1107
5.134.2.2	<code>__get_min_source</code> . . . . .	1107
5.134.2.3	<code>__init_winner</code> . . . . .	1108
5.134.2.4	<code>__insert_start</code> . . . . .	1108
5.134.3	Member Data Documentation . . . . .	1109
5.134.3.1	<code>_M_comp</code> . . . . .	1109

---

---

5.134.3.2	<code>_M_first_insert</code>	1109
5.134.3.3	<code>_M_log_k</code>	1109
5.134.3.4	<code>_M_losers</code>	1109
5.135	<code>__gnu_parallel::_LoserTreeBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1110
5.135.1	Detailed Description	1111
5.135.2	Constructor & Destructor Documentation	1111
5.135.2.1	<code>_LoserTreeBase</code>	1111
5.135.2.2	<code>~_LoserTreeBase</code>	1111
5.135.3	Member Function Documentation	1112
5.135.3.1	<code>__get_min_source</code>	1112
5.135.3.2	<code>__insert_start</code>	1112
5.135.4	Member Data Documentation	1112
5.135.4.1	<code>_M_comp</code>	1112
5.135.4.2	<code>_M_first_insert</code>	1113
5.135.4.3	<code>_M_log_k</code>	1113
5.135.4.4	<code>_M_losers</code>	1113
5.136	<code>__gnu_parallel::_LoserTreeBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference	1113
5.136.1	Detailed Description	1114
5.136.2	Member Data Documentation	1114
5.136.2.1	<code>_M_key</code>	1114
5.136.2.2	<code>_M_source</code>	1114
5.136.2.3	<code>_M_sup</code>	1114
5.137	<code>__gnu_parallel::_LoserTreePointer&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference	1115
5.137.1	Detailed Description	1116
5.138	<code>__gnu_parallel::_LoserTreePointer&lt; false, _Tp, _Compare &gt;</code> Class Template Reference	1116
5.138.1	Detailed Description	1117
5.139	<code>__gnu_parallel::_LoserTreePointerBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1117
5.139.1	Detailed Description	1118

---

5.140	<code>__gnu_parallel::_LoserTreePointerBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference . . . . .	1118
5.140.1	Detailed Description . . . . .	1119
5.141	<code>__gnu_parallel::_LoserTreePointerUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1119
5.141.1	Detailed Description . . . . .	1120
5.142	<code>__gnu_parallel::_LoserTreePointerUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1120
5.142.1	Detailed Description . . . . .	1121
5.143	<code>__gnu_parallel::_LoserTreePointerUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1122
5.143.1	Detailed Description . . . . .	1123
5.144	<code>__gnu_parallel::_LoserTreeTraits&lt; _Tp &gt;</code> Struct Template Reference	1123
5.144.1	Detailed Description . . . . .	1123
5.144.2	Member Data Documentation . . . . .	1124
5.144.2.1	<code>_M_use_pointer</code> . . . . .	1124
5.145	<code>__gnu_parallel::_LoserTreeUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1124
5.145.1	Detailed Description . . . . .	1125
5.146	<code>__gnu_parallel::_LoserTreeUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1126
5.146.1	Detailed Description . . . . .	1127
5.147	<code>__gnu_parallel::_LoserTreeUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference . . . . .	1127
5.147.1	Detailed Description . . . . .	1128
5.148	<code>__gnu_parallel::_Multiplies&lt; _Tp1, _Tp2, _Result &gt;</code> Struct Template Reference . . . . .	1128
5.148.1	Detailed Description . . . . .	1129
5.148.2	Member Typedef Documentation . . . . .	1130
5.148.2.1	<code>first_argument_type</code> . . . . .	1130
5.148.2.2	<code>result_type</code> . . . . .	1130
5.148.2.3	<code>second_argument_type</code> . . . . .	1130
5.149	<code>__gnu_parallel::_Nothing</code> Struct Reference . . . . .	1130
5.149.1	Detailed Description . . . . .	1130

---

5.149.2 Member Function Documentation . . . . .	1131
5.149.2.1 operator() . . . . .	1131
5.150 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference	1131
5.150.1 Detailed Description . . . . .	1131
5.150.2 Member Data Documentation . . . . .	1132
5.150.2.1 _M_begin . . . . .	1132
5.150.2.2 _M_end . . . . .	1132
5.151 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference	1132
5.151.1 Detailed Description . . . . .	1133
5.151.2 Member Typedef Documentation . . . . .	1134
5.151.2.1 first_argument_type . . . . .	1134
5.151.2.2 result_type . . . . .	1134
5.151.2.3 second_argument_type . . . . .	1134
5.152 __gnu_parallel::_PMWMSortingData< _RAIter > Struct Template Reference	1134
5.152.1 Detailed Description . . . . .	1135
5.152.2 Member Data Documentation . . . . .	1135
5.152.2.1 _M_num_threads . . . . .	1135
5.152.2.2 _M_offsets . . . . .	1135
5.152.2.3 _M_pieces . . . . .	1135
5.152.2.4 _M_samples . . . . .	1136
5.152.2.5 _M_source . . . . .	1136
5.152.2.6 _M_starts . . . . .	1136
5.152.2.7 _M_temporary . . . . .	1136
5.153 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference	1137
5.153.1 Detailed Description . . . . .	1137
5.153.2 Constructor & Destructor Documentation . . . . .	1137
5.153.2.1 _PseudoSequence . . . . .	1137
5.153.3 Member Function Documentation . . . . .	1138
5.153.3.1 begin . . . . .	1138

5.153.3.2 end . . . . .	1138
5.154__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference . . . . .	1138
5.154.1 Detailed Description . . . . .	1139
5.155__gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Ref- erence . . . . .	1139
5.155.1 Detailed Description . . . . .	1140
5.155.2 Member Typedef Documentation . . . . .	1140
5.155.2.1 _Piece . . . . .	1140
5.155.3 Constructor & Destructor Documentation . . . . .	1140
5.155.3.1 _QSBThreadLocal . . . . .	1140
5.155.4 Member Data Documentation . . . . .	1141
5.155.4.1 _M_elements_leftover . . . . .	1141
5.155.4.2 _M_global . . . . .	1141
5.155.4.3 _M_initial . . . . .	1141
5.155.4.4 _M_leftover_parts . . . . .	1141
5.155.4.5 _M_num_threads . . . . .	1141
5.156__gnu_parallel::_RandomNumber Class Reference . . . . .	1142
5.156.1 Detailed Description . . . . .	1142
5.156.2 Constructor & Destructor Documentation . . . . .	1142
5.156.2.1 _RandomNumber . . . . .	1142
5.156.2.2 _RandomNumber . . . . .	1142
5.156.3 Member Function Documentation . . . . .	1143
5.156.3.1 __genrand_bits . . . . .	1143
5.156.3.2 operator() . . . . .	1143
5.156.3.3 operator() . . . . .	1143
5.157__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference . . . . .	1143
5.157.1 Detailed Description . . . . .	1144
5.157.2 Constructor & Destructor Documentation . . . . .	1145
5.157.2.1 _RestrictedBoundedConcurrentQueue . . . . .	1145
5.157.2.2 ~_RestrictedBoundedConcurrentQueue . . . . .	1145

---

5.157.3 Member Function Documentation . . . . .	1145
5.157.3.1 pop_back . . . . .	1145
5.157.3.2 pop_front . . . . .	1145
5.157.3.3 push_front . . . . .	1146
5.158 __gnu_parallel::_SamplingSorter< __stable, __RAIter, __- StrictWeakOrdering > Struct Template Reference . . . . .	1146
5.158.1 Detailed Description . . . . .	1146
5.159 __gnu_parallel::_SamplingSorter< false, __RAIter, __- StrictWeakOrdering > Struct Template Reference . . . . .	1147
5.159.1 Detailed Description . . . . .	1147
5.160 __gnu_parallel::_Settings Struct Reference . . . . .	1147
5.160.1 Detailed Description . . . . .	1149
5.160.2 Member Function Documentation . . . . .	1149
5.160.2.1 get . . . . .	1149
5.160.2.2 set . . . . .	1149
5.160.3 Member Data Documentation . . . . .	1149
5.160.3.1 accumulate_minimal_n . . . . .	1149
5.160.3.2 adjacent_difference_minimal_n . . . . .	1150
5.160.3.3 cache_line_size . . . . .	1150
5.160.3.4 count_minimal_n . . . . .	1150
5.160.3.5 fill_minimal_n . . . . .	1150
5.160.3.6 find_increasing_factor . . . . .	1150
5.160.3.7 find_initial_block_size . . . . .	1150
5.160.3.8 find_maximum_block_size . . . . .	1151
5.160.3.9 find_sequential_search_size . . . . .	1151
5.160.3.10 for_each_minimal_n . . . . .	1151
5.160.3.11 generate_minimal_n . . . . .	1151
5.160.3.12 L1_cache_size . . . . .	1151
5.160.3.13 L2_cache_size . . . . .	1151
5.160.3.14 max_element_minimal_n . . . . .	1152
5.160.3.15 merge_minimal_n . . . . .	1152
5.160.3.16 merge_oversampling . . . . .	1152

5.160.3.17	<code>min_element_minimal_n</code> . . . . .	1152
5.160.3.18	<code>multiway_merge_minimal_k</code> . . . . .	1152
5.160.3.19	<code>multiway_merge_minimal_n</code> . . . . .	1152
5.160.3.20	<code>multiway_merge_oversampling</code> . . . . .	1153
5.160.3.21	<code>nth_element_minimal_n</code> . . . . .	1153
5.160.3.22	<code>partial_sort_minimal_n</code> . . . . .	1153
5.160.3.23	<code>partial_sum_dilation</code> . . . . .	1153
5.160.3.24	<code>partial_sum_minimal_n</code> . . . . .	1153
5.160.3.25	<code>partition_chunk_share</code> . . . . .	1153
5.160.3.26	<code>partition_chunk_size</code> . . . . .	1154
5.160.3.27	<code>partition_minimal_n</code> . . . . .	1154
5.160.3.28	<code>qsb_steals</code> . . . . .	1154
5.160.3.29	<code>random_shuffle_minimal_n</code> . . . . .	1154
5.160.3.30	<code>replace_minimal_n</code> . . . . .	1154
5.160.3.31	<code>search_minimal_n</code> . . . . .	1154
5.160.3.32	<code>set_difference_minimal_n</code> . . . . .	1154
5.160.3.33	<code>set_intersection_minimal_n</code> . . . . .	1155
5.160.3.34	<code>set_symmetric_difference_minimal_n</code> . . . . .	1155
5.160.3.35	<code>set_union_minimal_n</code> . . . . .	1155
5.160.3.36	<code>sort_minimal_n</code> . . . . .	1155
5.160.3.37	<code>sort_mwms_oversampling</code> . . . . .	1155
5.160.3.38	<code>sort_qs_num_samples_preset</code> . . . . .	1155
5.160.3.39	<code>sort_qsb_base_case_maximal_n</code> . . . . .	1156
5.160.3.40	<code>TLB_size</code> . . . . .	1156
5.160.3.41	<code>transform_minimal_n</code> . . . . .	1156
5.160.3.42	<code>unique_copy_minimal_n</code> . . . . .	1156
5.161	<code>__gnu_parallel::SplitConsistently&lt; __exact, _RAIter, _Compare, _-</code> <code>SortingPlacesIterator &gt;</code> Struct Template Reference . . . . .	1156
5.161.1	Detailed Description . . . . .	1157
5.162	<code>__gnu_parallel::SplitConsistently&lt; false, _RAIter, _Compare, _-</code> <code>SortingPlacesIterator &gt;</code> Struct Template Reference . . . . .	1157
5.162.1	Detailed Description . . . . .	1157

---

5.163	<a href="#">__gnu_parallel::_SplitConsistently&lt; true, _RAIter, _Compare, _SortingPlacesIterator &gt; Struct Template Reference</a>	1158
5.163.1	Detailed Description	1158
5.164	<a href="#">__gnu_parallel::balanced_quicksort_tag Struct Reference</a>	1158
5.164.1	Detailed Description	1159
5.164.2	Member Function Documentation	1159
5.164.2.1	<a href="#">__get_num_threads</a>	1159
5.164.2.2	<a href="#">set_num_threads</a>	1160
5.165	<a href="#">__gnu_parallel::balanced_tag Struct Reference</a>	1160
5.165.1	Detailed Description	1160
5.165.2	Member Function Documentation	1161
5.165.2.1	<a href="#">__get_num_threads</a>	1161
5.165.2.2	<a href="#">set_num_threads</a>	1161
5.166	<a href="#">__gnu_parallel::constant_size_blocks_tag Struct Reference</a>	1161
5.166.1	Detailed Description	1162
5.167	<a href="#">__gnu_parallel::default_parallel_tag Struct Reference</a>	1162
5.167.1	Detailed Description	1163
5.167.2	Member Function Documentation	1163
5.167.2.1	<a href="#">__get_num_threads</a>	1163
5.167.2.2	<a href="#">set_num_threads</a>	1164
5.168	<a href="#">__gnu_parallel::equal_split_tag Struct Reference</a>	1164
5.168.1	Detailed Description	1164
5.169	<a href="#">__gnu_parallel::exact_tag Struct Reference</a>	1165
5.169.1	Detailed Description	1165
5.169.2	Member Function Documentation	1166
5.169.2.1	<a href="#">__get_num_threads</a>	1166
5.169.2.2	<a href="#">set_num_threads</a>	1166
5.170	<a href="#">__gnu_parallel::find_tag Struct Reference</a>	1166
5.170.1	Detailed Description	1167
5.171	<a href="#">__gnu_parallel::growing_blocks_tag Struct Reference</a>	1167
5.171.1	Detailed Description	1168

---

5.172__gnu_parallel::multiway_mergesort_exact_tag Struct Reference . . .	1168
5.172.1 Detailed Description . . . . .	1169
5.172.2 Member Function Documentation . . . . .	1169
5.172.2.1 __get_num_threads . . . . .	1169
5.172.2.2 set_num_threads . . . . .	1170
5.173__gnu_parallel::multiway_mergesort_sampling_tag Struct Reference .	1170
5.173.1 Detailed Description . . . . .	1171
5.173.2 Member Function Documentation . . . . .	1171
5.173.2.1 __get_num_threads . . . . .	1171
5.173.2.2 set_num_threads . . . . .	1171
5.174__gnu_parallel::multiway_mergesort_tag Struct Reference . . . . .	1171
5.174.1 Detailed Description . . . . .	1172
5.174.2 Member Function Documentation . . . . .	1172
5.174.2.1 __get_num_threads . . . . .	1172
5.174.2.2 set_num_threads . . . . .	1173
5.175__gnu_parallel::omp_loop_static_tag Struct Reference . . . . .	1173
5.175.1 Detailed Description . . . . .	1174
5.175.2 Member Function Documentation . . . . .	1174
5.175.2.1 __get_num_threads . . . . .	1174
5.175.2.2 set_num_threads . . . . .	1174
5.176__gnu_parallel::omp_loop_tag Struct Reference . . . . .	1174
5.176.1 Detailed Description . . . . .	1175
5.176.2 Member Function Documentation . . . . .	1175
5.176.2.1 __get_num_threads . . . . .	1175
5.176.2.2 set_num_threads . . . . .	1176
5.177__gnu_parallel::parallel_tag Struct Reference . . . . .	1176
5.177.1 Detailed Description . . . . .	1178
5.177.2 Constructor & Destructor Documentation . . . . .	1178
5.177.2.1 parallel_tag . . . . .	1178
5.177.2.2 parallel_tag . . . . .	1178
5.177.3 Member Function Documentation . . . . .	1178

---

---

5.177.3.1	<code>__get_num_threads</code>	1178
5.177.3.2	<code>set_num_threads</code>	1178
5.178	<code>__gnu_parallel::quicksort_tag</code> Struct Reference	1179
5.178.1	Detailed Description	1179
5.178.2	Member Function Documentation	1180
5.178.2.1	<code>__get_num_threads</code>	1180
5.178.2.2	<code>set_num_threads</code>	1180
5.179	<code>__gnu_parallel::sampling_tag</code> Struct Reference	1180
5.179.1	Detailed Description	1181
5.179.2	Member Function Documentation	1181
5.179.2.1	<code>__get_num_threads</code>	1181
5.179.2.2	<code>set_num_threads</code>	1182
5.180	<code>__gnu_parallel::sequential_tag</code> Struct Reference	1182
5.180.1	Detailed Description	1182
5.181	<code>__gnu_parallel::unbalanced_tag</code> Struct Reference	1182
5.181.1	Detailed Description	1183
5.181.2	Member Function Documentation	1183
5.181.2.1	<code>__get_num_threads</code>	1183
5.181.2.2	<code>set_num_threads</code>	1184
5.182	<code>__gnu_pbds::associative_container_tag</code> Struct Reference	1184
5.182.1	Detailed Description	1184
5.183	<code>__gnu_pbds::basic_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator &gt;</code> Class Template Reference	1185
5.183.1	Detailed Description	1186
5.184	<code>__gnu_pbds::basic_hash_tag</code> Struct Reference	1186
5.184.1	Detailed Description	1187
5.185	<code>__gnu_pbds::basic_tree&lt; Key, Mapped, Tag, Node_Update, Policy_TI, Allocator &gt;</code> Class Template Reference	1187
5.185.1	Detailed Description	1188
5.186	<code>__gnu_pbds::basic_tree_tag</code> Struct Reference	1188
5.186.1	Detailed Description	1189

---

5.187__gnu_pbds::binary_heap_tag Struct Reference . . . . .	1189
5.187.1 Detailed Description . . . . .	1189
5.188__gnu_pbds::binomial_heap_tag Struct Reference . . . . .	1190
5.188.1 Detailed Description . . . . .	1190
5.189__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference . . . . .	1191
5.189.1 Detailed Description . . . . .	1192
5.190__gnu_pbds::cc_hash_tag Struct Reference . . . . .	1193
5.190.1 Detailed Description . . . . .	1193
5.191__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Alloca- tor > Class Template Reference . . . . .	1194
5.191.1 Detailed Description . . . . .	1195
5.192__gnu_pbds::container_tag Struct Reference . . . . .	1195
5.192.1 Detailed Description . . . . .	1195
5.193__gnu_pbds::container_traits< Cntnr > Struct Template Reference . .	1196
5.193.1 Detailed Description . . . . .	1196
5.194__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false > Struct Template Reference . . . . .	1196
5.194.1 Detailed Description . . . . .	1197
5.195__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true > Struct Template Reference . . . . .	1197
5.195.1 Detailed Description . . . . .	1198
5.196__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, false > Struct Template Reference . . . . .	1198
5.196.1 Detailed Description . . . . .	1199
5.197__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, true > Struct Template Reference . . . . .	1199
5.197.1 Detailed Description . . . . .	1200
5.198__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference . . . . .	1200
5.198.1 Detailed Description . . . . .	1202
5.199__gnu_pbds::gp_hash_tag Struct Reference . . . . .	1202

---

5.199.1 Detailed Description . . . . .	1203
5.200__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Al- locator > Class Template Reference . . . . .	1203
5.200.1 Detailed Description . . . . .	1205
5.201__gnu_pbds::list_update_tag Struct Reference . . . . .	1205
5.201.1 Detailed Description . . . . .	1206
5.202__gnu_pbds::null_mapped_type Struct Reference . . . . .	1206
5.202.1 Detailed Description . . . . .	1206
5.203__gnu_pbds::ov_tree_tag Struct Reference . . . . .	1207
5.203.1 Detailed Description . . . . .	1207
5.204__gnu_pbds::pairing_heap_tag Struct Reference . . . . .	1208
5.204.1 Detailed Description . . . . .	1208
5.205__gnu_pbds::pat_trie_tag Struct Reference . . . . .	1208
5.205.1 Detailed Description . . . . .	1209
5.206__gnu_pbds::priority_queue_tag Struct Reference . . . . .	1210
5.206.1 Detailed Description . . . . .	1210
5.207__gnu_pbds::rb_tree_tag Struct Reference . . . . .	1210
5.207.1 Detailed Description . . . . .	1211
5.208__gnu_pbds::rc_binomial_heap_tag Struct Reference . . . . .	1212
5.208.1 Detailed Description . . . . .	1212
5.209__gnu_pbds::sequence_tag Struct Reference . . . . .	1212
5.209.1 Detailed Description . . . . .	1213
5.210__gnu_pbds::splay_tree_tag Struct Reference . . . . .	1213
5.210.1 Detailed Description . . . . .	1214
5.211__gnu_pbds::string_tag Struct Reference . . . . .	1214
5.211.1 Detailed Description . . . . .	1215
5.212__gnu_pbds::thin_heap_tag Struct Reference . . . . .	1215
5.212.1 Detailed Description . . . . .	1216
5.213__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allo- cator > Class Template Reference . . . . .	1216
5.213.1 Detailed Description . . . . .	1218
5.214__gnu_pbds::tree_tag Struct Reference . . . . .	1218

---

5.214.1 Detailed Description . . . . .	1219
5.215 __gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_-, Update, Allocator > Class Template Reference . . . . .	1219
5.215.1 Detailed Description . . . . .	1220
5.216 __gnu_pbds::trie_tag Struct Reference . . . . .	1221
5.216.1 Detailed Description . . . . .	1221
5.217 __gnu_profile::__container_size_info Class Reference . . . . .	1222
5.217.1 Detailed Description . . . . .	1223
5.218 __gnu_profile::__container_size_stack_info Class Reference . . . . .	1223
5.218.1 Detailed Description . . . . .	1224
5.219 __gnu_profile::__hashfunc_info Class Reference . . . . .	1224
5.219.1 Detailed Description . . . . .	1225
5.220 __gnu_profile::__hashfunc_stack_info Class Reference . . . . .	1226
5.220.1 Detailed Description . . . . .	1227
5.221 __gnu_profile::__list2vector_info Class Reference . . . . .	1227
5.221.1 Detailed Description . . . . .	1228
5.222 __gnu_profile::__map2umap_info Class Reference . . . . .	1228
5.222.1 Detailed Description . . . . .	1230
5.223 __gnu_profile::__map2umap_stack_info Class Reference . . . . .	1230
5.223.1 Detailed Description . . . . .	1231
5.224 __gnu_profile::__object_info_base Class Reference . . . . .	1231
5.224.1 Detailed Description . . . . .	1232
5.225 __gnu_profile::__reentrance_guard Struct Reference . . . . .	1232
5.225.1 Detailed Description . . . . .	1232
5.226 __gnu_profile::__stack_hash Class Reference . . . . .	1233
5.226.1 Detailed Description . . . . .	1233
5.227 __gnu_profile::__stack_info_base< __object_info > Class Template Reference . . . . .	1233
5.227.1 Detailed Description . . . . .	1233
5.228 __gnu_profile::__trace_base< __object_info, __stack_info > Class Template Reference . . . . .	1234
5.228.1 Detailed Description . . . . .	1234

---

---

5.229	<a href="#">__gnu_profile::__trace_container_size Class Reference</a>	1235
5.229.1	<a href="#">Detailed Description</a>	1235
5.230	<a href="#">__gnu_profile::__trace_hash_func Class Reference</a>	1235
5.230.1	<a href="#">Detailed Description</a>	1236
5.231	<a href="#">__gnu_profile::__trace_hashtable_size Class Reference</a>	1237
5.231.1	<a href="#">Detailed Description</a>	1237
5.232	<a href="#">__gnu_profile::__trace_map2umap Class Reference</a>	1237
5.232.1	<a href="#">Detailed Description</a>	1238
5.233	<a href="#">__gnu_profile::__trace_vector_size Class Reference</a>	1239
5.233.1	<a href="#">Detailed Description</a>	1239
5.234	<a href="#">__gnu_profile::__trace_vector_to_list Class Reference</a>	1239
5.234.1	<a href="#">Detailed Description</a>	1241
5.235	<a href="#">__gnu_profile::__vector2list_info Class Reference</a>	1241
5.235.1	<a href="#">Detailed Description</a>	1242
5.236	<a href="#">__gnu_profile::__vector2list_stack_info Class Reference</a>	1242
5.236.1	<a href="#">Detailed Description</a>	1244
5.237	<a href="#">__gnu_profile::__warning_data Struct Reference</a>	1244
5.237.1	<a href="#">Detailed Description</a>	1244
5.238	<a href="#">std::__basic_future&lt;_Res&gt; Class Template Reference</a>	1245
5.238.1	<a href="#">Detailed Description</a>	1246
5.238.2	<a href="#">Member Function Documentation</a>	1246
5.238.2.1	<a href="#">_M_get_result</a>	1246
5.239	<a href="#">std::__codecvt_abstract_base&lt;_InternT, _ExternT, _StateT&gt; Class Template Reference</a>	1246
5.239.1	<a href="#">Detailed Description</a>	1248
5.239.2	<a href="#">Member Function Documentation</a>	1248
5.239.2.1	<a href="#">do_out</a>	1248
5.239.2.2	<a href="#">in</a>	1249
5.239.2.3	<a href="#">out</a>	1250
5.239.2.4	<a href="#">unshift</a>	1251
5.240	<a href="#">std::__ctype_abstract_base&lt;_CharT&gt; Class Template Reference</a>	1251

---

---

5.240.1 Detailed Description . . . . .	1254
5.240.2 Member Typedef Documentation . . . . .	1254
5.240.2.1 char_type . . . . .	1254
5.240.3 Member Function Documentation . . . . .	1255
5.240.3.1 do_is . . . . .	1255
5.240.3.2 do_is . . . . .	1255
5.240.3.3 do_narrow . . . . .	1256
5.240.3.4 do_narrow . . . . .	1256
5.240.3.5 do_scan_is . . . . .	1257
5.240.3.6 do_scan_not . . . . .	1257
5.240.3.7 do_tolower . . . . .	1258
5.240.3.8 do_tolower . . . . .	1258
5.240.3.9 do_toupper . . . . .	1259
5.240.3.10do_toupper . . . . .	1259
5.240.3.11do_widen . . . . .	1260
5.240.3.12do_widen . . . . .	1260
5.240.3.13s . . . . .	1261
5.240.3.14s . . . . .	1261
5.240.3.15narrow . . . . .	1262
5.240.3.16narrow . . . . .	1262
5.240.3.17scan_is . . . . .	1263
5.240.3.18scan_not . . . . .	1263
5.240.3.19tolower . . . . .	1264
5.240.3.20tolower . . . . .	1264
5.240.3.21toupper . . . . .	1264
5.240.3.22toupper . . . . .	1265
5.240.3.23widen . . . . .	1265
5.240.3.24widen . . . . .	1266
5.241 std::__debug::bitset<_Nb> Class Template Reference . . . . .	1266
5.241.1 Detailed Description . . . . .	1269
5.241.2 Member Function Documentation . . . . .	1269

---

---

5.241.2.1	_M_detach_all . . . . .	1269
5.241.2.2	_M_detach_singular . . . . .	1269
5.241.2.3	_M_get_mutex . . . . .	1269
5.241.2.4	_M_invalidate_all . . . . .	1269
5.241.2.5	_M_revalidate_singular . . . . .	1270
5.241.2.6	_M_swap . . . . .	1270
5.241.3	Member Data Documentation . . . . .	1270
5.241.3.1	_M_const_iterators . . . . .	1270
5.241.3.2	_M_iterators . . . . .	1270
5.241.3.3	_M_version . . . . .	1270
5.242	std::__debug::deque< _Tp, _Allocator > Class Template Reference . . . . .	1271
5.242.1	Detailed Description . . . . .	1274
5.242.2	Member Function Documentation . . . . .	1274
5.242.2.1	_M_detach_all . . . . .	1274
5.242.2.2	_M_detach_singular . . . . .	1274
5.242.2.3	_M_get_mutex . . . . .	1274
5.242.2.4	_M_invalidate_all . . . . .	1275
5.242.2.5	_M_invalidate_if . . . . .	1275
5.242.2.6	_M_revalidate_singular . . . . .	1275
5.242.2.7	_M_swap . . . . .	1275
5.242.2.8	_M_transfer_iter . . . . .	1275
5.242.3	Member Data Documentation . . . . .	1275
5.242.3.1	_M_const_iterators . . . . .	1275
5.242.3.2	_M_iterators . . . . .	1276
5.242.3.3	_M_version . . . . .	1276
5.243	std::__debug::list< _Tp, _Allocator > Class Template Reference . . . . .	1276
5.243.1	Detailed Description . . . . .	1280
5.243.2	Member Function Documentation . . . . .	1280
5.243.2.1	_M_detach_all . . . . .	1280
5.243.2.2	_M_detach_singular . . . . .	1280
5.243.2.3	_M_get_mutex . . . . .	1280

---

5.243.2.4	<code>_M_invalidate_all</code>	1281
5.243.2.5	<code>_M_invalidate_if</code>	1281
5.243.2.6	<code>_M_revalidate_singular</code>	1281
5.243.2.7	<code>_M_swap</code>	1281
5.243.2.8	<code>_M_transfer_iter</code>	1281
5.243.3	Member Data Documentation	1281
5.243.3.1	<code>_M_const_iterators</code>	1281
5.243.3.2	<code>_M_iterators</code>	1282
5.243.3.3	<code>_M_version</code>	1282
5.244	<code>std::__debug::map&lt;_Key, _Tp, _Compare, _Allocator&gt;</code> Class Template Reference	1282
5.244.1	Detailed Description	1285
5.244.2	Member Function Documentation	1285
5.244.2.1	<code>_M_detach_all</code>	1285
5.244.2.2	<code>_M_detach_singular</code>	1286
5.244.2.3	<code>_M_get_mutex</code>	1286
5.244.2.4	<code>_M_invalidate_all</code>	1286
5.244.2.5	<code>_M_invalidate_if</code>	1286
5.244.2.6	<code>_M_revalidate_singular</code>	1286
5.244.2.7	<code>_M_swap</code>	1287
5.244.2.8	<code>_M_transfer_iter</code>	1287
5.244.3	Member Data Documentation	1287
5.244.3.1	<code>_M_const_iterators</code>	1287
5.244.3.2	<code>_M_iterators</code>	1287
5.244.3.3	<code>_M_version</code>	1287
5.245	<code>std::__debug::multimap&lt;_Key, _Tp, _Compare, _Allocator&gt;</code> Class Template Reference	1288
5.245.1	Detailed Description	1290
5.245.2	Member Function Documentation	1291
5.245.2.1	<code>_M_detach_all</code>	1291
5.245.2.2	<code>_M_detach_singular</code>	1291
5.245.2.3	<code>_M_get_mutex</code>	1291

---

5.245.2.4	<code>_M_invalidate_all</code>	1291
5.245.2.5	<code>_M_invalidate_if</code>	1291
5.245.2.6	<code>_M_revalidate_singular</code>	1292
5.245.2.7	<code>_M_swap</code>	1292
5.245.2.8	<code>_M_transfer_iter</code>	1292
5.245.3	Member Data Documentation	1292
5.245.3.1	<code>_M_const_iterators</code>	1292
5.245.3.2	<code>_M_iterators</code>	1292
5.245.3.3	<code>_M_version</code>	1293
5.246	<code>std::__debug::multiset&lt; _Key, _Compare, _Allocator &gt; Class Template Reference</code>	1293
5.246.1	Detailed Description	1296
5.246.2	Member Function Documentation	1296
5.246.2.1	<code>_M_detach_all</code>	1296
5.246.2.2	<code>_M_detach_singular</code>	1296
5.246.2.3	<code>_M_get_mutex</code>	1296
5.246.2.4	<code>_M_invalidate_all</code>	1296
5.246.2.5	<code>_M_invalidate_if</code>	1297
5.246.2.6	<code>_M_revalidate_singular</code>	1297
5.246.2.7	<code>_M_swap</code>	1297
5.246.2.8	<code>_M_transfer_iter</code>	1297
5.246.3	Member Data Documentation	1297
5.246.3.1	<code>_M_const_iterators</code>	1297
5.246.3.2	<code>_M_iterators</code>	1298
5.246.3.3	<code>_M_version</code>	1298
5.247	<code>std::__debug::set&lt; _Key, _Compare, _Allocator &gt; Class Template Reference</code>	1298
5.247.1	Detailed Description	1301
5.247.2	Member Function Documentation	1301
5.247.2.1	<code>_M_detach_all</code>	1301
5.247.2.2	<code>_M_detach_singular</code>	1302
5.247.2.3	<code>_M_get_mutex</code>	1302

---

5.247.2.4	<code>_M_invalidate_all</code>	1302
5.247.2.5	<code>_M_invalidate_if</code>	1302
5.247.2.6	<code>_M_revalidate_singular</code>	1302
5.247.2.7	<code>_M_swap</code>	1303
5.247.2.8	<code>_M_transfer_iter</code>	1303
5.247.3	Member Data Documentation	1303
5.247.3.1	<code>_M_const_iterators</code>	1303
5.247.3.2	<code>_M_iterators</code>	1303
5.247.3.3	<code>_M_version</code>	1303
5.248	<code>std::__debug::unordered_map&lt;_Key, _Tp, _Hash, _Pred, _Alloc &gt;</code> Class Template Reference	1304
5.248.1	Detailed Description	1306
5.248.2	Member Function Documentation	1306
5.248.2.1	<code>_M_detach_all</code>	1306
5.248.2.2	<code>_M_detach_singular</code>	1306
5.248.2.3	<code>_M_get_mutex</code>	1307
5.248.2.4	<code>_M_invalidate_all</code>	1307
5.248.2.5	<code>_M_invalidate_if</code>	1307
5.248.2.6	<code>_M_revalidate_singular</code>	1307
5.248.2.7	<code>_M_swap</code>	1307
5.248.2.8	<code>_M_transfer_iter</code>	1308
5.248.3	Member Data Documentation	1308
5.248.3.1	<code>_M_const_iterators</code>	1308
5.248.3.2	<code>_M_iterators</code>	1308
5.248.3.3	<code>_M_version</code>	1308
5.249	<code>std::__debug::unordered_multimap&lt;_Key, _Tp, _Hash, _Pred, _Alloc &gt;</code> > Class Template Reference	1309
5.249.1	Detailed Description	1311
5.249.2	Member Function Documentation	1311
5.249.2.1	<code>_M_detach_all</code>	1311
5.249.2.2	<code>_M_detach_singular</code>	1311
5.249.2.3	<code>_M_get_mutex</code>	1311

---

---

5.249.2.4	_M_invalidate_all	1311
5.249.2.5	_M_invalidate_if	1312
5.249.2.6	_M_revalidate_singular	1312
5.249.2.7	_M_swap	1312
5.249.2.8	_M_transfer_iter	1312
5.249.3	Member Data Documentation	1312
5.249.3.1	_M_const_iterators	1312
5.249.3.2	_M_iterators	1313
5.249.3.3	_M_version	1313
5.250	std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1313
5.250.1	Detailed Description	1315
5.250.2	Member Function Documentation	1315
5.250.2.1	_M_detach_all	1315
5.250.2.2	_M_detach_singular	1315
5.250.2.3	_M_get_mutex	1316
5.250.2.4	_M_invalidate_all	1316
5.250.2.5	_M_invalidate_if	1316
5.250.2.6	_M_revalidate_singular	1316
5.250.2.7	_M_swap	1316
5.250.2.8	_M_transfer_iter	1317
5.250.3	Member Data Documentation	1317
5.250.3.1	_M_const_iterators	1317
5.250.3.2	_M_iterators	1317
5.250.3.3	_M_version	1317
5.251	std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1318
5.251.1	Detailed Description	1320
5.251.2	Member Function Documentation	1320
5.251.2.1	_M_detach_all	1320
5.251.2.2	_M_detach_singular	1320
5.251.2.3	_M_get_mutex	1321

---

5.251.2.4	<a href="#">_M_invalidate_all</a>	1321
5.251.2.5	<a href="#">_M_invalidate_if</a>	1321
5.251.2.6	<a href="#">_M_revalidate_singular</a>	1321
5.251.2.7	<a href="#">_M_swap</a>	1321
5.251.2.8	<a href="#">_M_transfer_iter</a>	1322
5.251.3	<a href="#">Member Data Documentation</a>	1322
5.251.3.1	<a href="#">_M_const_iterators</a>	1322
5.251.3.2	<a href="#">_M_iterators</a>	1322
5.251.3.3	<a href="#">_M_version</a>	1322
5.252	<a href="#">std::__debug::vector&lt; _Tp, _Allocator &gt; Class Template Reference</a>	1323
5.252.1	<a href="#">Detailed Description</a>	1326
5.252.2	<a href="#">Constructor &amp; Destructor Documentation</a>	1326
5.252.2.1	<a href="#">vector</a>	1326
5.252.3	<a href="#">Member Function Documentation</a>	1326
5.252.3.1	<a href="#">_M_detach_all</a>	1326
5.252.3.2	<a href="#">_M_detach_singular</a>	1326
5.252.3.3	<a href="#">_M_get_mutex</a>	1326
5.252.3.4	<a href="#">_M_invalidate_all</a>	1327
5.252.3.5	<a href="#">_M_invalidate_if</a>	1327
5.252.3.6	<a href="#">_M_revalidate_singular</a>	1327
5.252.3.7	<a href="#">_M_swap</a>	1327
5.252.3.8	<a href="#">_M_transfer_iter</a>	1327
5.252.4	<a href="#">Member Data Documentation</a>	1327
5.252.4.1	<a href="#">_M_const_iterators</a>	1327
5.252.4.2	<a href="#">_M_iterators</a>	1328
5.252.4.3	<a href="#">_M_version</a>	1328
5.253	<a href="#">std::__declval_protector&lt; _Tp &gt; Struct Template Reference</a>	1328
5.253.1	<a href="#">Detailed Description</a>	1329
5.254	<a href="#">std::__exception_ptr::exception_ptr Class Reference</a>	1329
5.254.1	<a href="#">Detailed Description</a>	1330
5.255	<a href="#">std::__future_base Struct Reference</a>	1330

---

---

5.255.1 Detailed Description . . . . .	1331
5.256std::__future_base::__Ptr< _Res > Struct Template Reference . . . . .	1331
5.256.1 Detailed Description . . . . .	1331
5.257std::__future_base::__Result< _Res > Struct Template Reference . . . . .	1332
5.257.1 Detailed Description . . . . .	1333
5.258std::__future_base::__Result< _Res & > Struct Template Reference . . . . .	1333
5.258.1 Detailed Description . . . . .	1334
5.259std::__future_base::__Result< void > Struct Template Reference . . . . .	1334
5.259.1 Detailed Description . . . . .	1334
5.260std::__future_base::__Result_base Struct Reference . . . . .	1335
5.260.1 Detailed Description . . . . .	1335
5.261std::__future_base::__State Class Reference . . . . .	1336
5.261.1 Detailed Description . . . . .	1336
5.262std::__is_location_invariant< _Tp > Struct Template Reference . . . . .	1337
5.262.1 Detailed Description . . . . .	1337
5.263std::__numeric_limits_base Struct Reference . . . . .	1337
5.263.1 Detailed Description . . . . .	1339
5.263.2 Member Data Documentation . . . . .	1339
5.263.2.1 digits . . . . .	1339
5.263.2.2 digits10 . . . . .	1339
5.263.2.3 has_denorm . . . . .	1339
5.263.2.4 has_denorm_loss . . . . .	1339
5.263.2.5 has_infinity . . . . .	1340
5.263.2.6 has_quiet_NaN . . . . .	1340
5.263.2.7 has_signaling_NaN . . . . .	1340
5.263.2.8 is_bounded . . . . .	1340
5.263.2.9 is_exact . . . . .	1340
5.263.2.10is_iec559 . . . . .	1340
5.263.2.11is_integer . . . . .	1340
5.263.2.12is_modulo . . . . .	1341
5.263.2.13is_signed . . . . .	1341

5.263.2.14	<code>is_specialized</code>	1341
5.263.2.15	<code>max_digits10</code>	1341
5.263.2.16	<code>max_exponent</code>	1341
5.263.2.17	<code>max_exponent10</code>	1341
5.263.2.18	<code>min_exponent</code>	1342
5.263.2.19	<code>min_exponent10</code>	1342
5.263.2.20	<code>radix</code>	1342
5.263.2.21	<code>round_style</code>	1342
5.263.2.22	<code>triyeness_before</code>	1342
5.263.2.23	<code>traps</code>	1342
5.264	<code>std::__parallel::_CRandNumber&lt;_MustBeInt&gt;</code> Struct Template Reference	1343
5.264.1	Detailed Description	1343
5.265	<code>std::__profile::bitset&lt;_Nb&gt;</code> Class Template Reference	1343
5.265.1	Detailed Description	1345
5.266	<code>std::__profile::deque&lt;_Tp, _Allocator&gt;</code> Class Template Reference	1345
5.266.1	Detailed Description	1347
5.267	<code>std::__profile::list&lt;_Tp, _Allocator&gt;</code> Class Template Reference	1347
5.267.1	Detailed Description	1350
5.268	<code>std::__profile::map&lt;_Key, _Tp, _Compare, _Allocator&gt;</code> Class Template Reference	1350
5.268.1	Detailed Description	1352
5.269	<code>std::__profile::multimap&lt;_Key, _Tp, _Compare, _Allocator&gt;</code> Class Template Reference	1352
5.269.1	Detailed Description	1354
5.270	<code>std::__profile::multiset&lt;_Key, _Compare, _Allocator&gt;</code> Class Template Reference	1354
5.270.1	Detailed Description	1356
5.271	<code>std::__profile::set&lt;_Key, _Compare, _Allocator&gt;</code> Class Template Reference	1356
5.271.1	Detailed Description	1358
5.272	<code>std::__profile::unordered_map&lt;_Key, _Tp, _Hash, _Pred, _Alloc&gt;</code> Class Template Reference	1358

5.272.1 Detailed Description . . . . .	1360
5.273std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1360
5.273.1 Detailed Description . . . . .	1361
5.274std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1361
5.274.1 Detailed Description . . . . .	1363
5.275std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	1363
5.275.1 Detailed Description . . . . .	1364
5.276std::_Base_bitset< _Nw > Struct Template Reference . . . . .	1364
5.276.1 Detailed Description . . . . .	1366
5.276.2 Member Data Documentation . . . . .	1366
5.276.2.1 _M_w . . . . .	1366
5.277std::_Base_bitset< 0 > Struct Template Reference . . . . .	1366
5.277.1 Detailed Description . . . . .	1367
5.278std::_Base_bitset< 1 > Struct Template Reference . . . . .	1367
5.278.1 Detailed Description . . . . .	1368
5.279std::_Build_index_tuple< _Num > Struct Template Reference . . . . .	1369
5.279.1 Detailed Description . . . . .	1369
5.280std::_Deque_base< _Tp, _Alloc > Class Template Reference . . . . .	1369
5.280.1 Detailed Description . . . . .	1371
5.280.2 Member Function Documentation . . . . .	1371
5.280.2.1 _M_initialize_map . . . . .	1371
5.281std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference . . . . .	1371
5.281.1 Detailed Description . . . . .	1373
5.281.2 Member Function Documentation . . . . .	1373
5.281.2.1 _M_set_node . . . . .	1373
5.282std::_Derives_from_binary_function< _Tp > Struct Template Reference	1374
5.282.1 Detailed Description . . . . .	1374
5.283std::_Derives_from_unary_function< _Tp > Struct Template Reference	1374
5.283.1 Detailed Description . . . . .	1374

5.284	<code>std::_Function_base</code> Class Reference . . . . .	1375
5.284.1	Detailed Description . . . . .	1376
5.285	<code>std::_Function_to_function_pointer&lt;_Tp, _IsFunctionType&gt;</code> Struct Template Reference . . . . .	1376
5.285.1	Detailed Description . . . . .	1376
5.286	<code>std::_Fwd_list_base&lt;_Tp, _Alloc&gt;</code> Struct Template Reference . . . . .	1376
5.286.1	Detailed Description . . . . .	1378
5.287	<code>std::_Fwd_list_const_iterator&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1378
5.287.1	Detailed Description . . . . .	1379
5.288	<code>std::_Fwd_list_iterator&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1379
5.288.1	Detailed Description . . . . .	1380
5.289	<code>std::_Fwd_list_node&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1380
5.289.1	Detailed Description . . . . .	1381
5.290	<code>std::_Fwd_list_node_base</code> Struct Reference . . . . .	1382
5.290.1	Detailed Description . . . . .	1383
5.291	<code>std::_Has_result_type_helper&lt;_Tp&gt;</code> Class Template Reference . . . . .	1383
5.291.1	Detailed Description . . . . .	1383
5.292	<code>std::_Index_tuple&lt;_Indexes&gt;</code> Struct Template Reference . . . . .	1383
5.292.1	Detailed Description . . . . .	1384
5.293	<code>std::_List_base&lt;_Tp, _Alloc&gt;</code> Class Template Reference . . . . .	1384
5.293.1	Detailed Description . . . . .	1385
5.294	<code>std::_List_const_iterator&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1385
5.294.1	Detailed Description . . . . .	1386
5.295	<code>std::_List_iterator&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1387
5.295.1	Detailed Description . . . . .	1387
5.296	<code>std::_List_node&lt;_Tp&gt;</code> Struct Template Reference . . . . .	1388
5.296.1	Detailed Description . . . . .	1389
5.296.2	Member Data Documentation . . . . .	1389
5.296.2.1	<code>_M_data</code> . . . . .	1389
5.297	<code>std::_List_node_base</code> Struct Reference . . . . .	1389
5.297.1	Detailed Description . . . . .	1390

5.298	<code>std::_Maybe_get_result_type&lt; _Has_result_type, _Functor &gt; Struct</code>	
	Template Reference	1390
5.298.1	Detailed Description	1390
5.299	<code>std::_Maybe_unary_or_binary_function&lt; _Res, _ArgTypes &gt; Struct</code>	
	Template Reference	1391
5.299.1	Detailed Description	1391
5.300	<code>std::_Maybe_unary_or_binary_function&lt; _Res, _T1 &gt; Struct</code>	
	Template Reference	1391
5.300.1	Detailed Description	1392
5.300.2	Member Typedef Documentation	1392
5.300.2.1	<code>argument_type</code>	1392
5.300.2.2	<code>result_type</code>	1393
5.301	<code>std::_Maybe_unary_or_binary_function&lt; _Res, _T1, _T2 &gt; Struct</code>	
	Template Reference	1393
5.301.1	Detailed Description	1394
5.301.2	Member Typedef Documentation	1394
5.301.2.1	<code>first_argument_type</code>	1394
5.301.2.2	<code>result_type</code>	1394
5.301.2.3	<code>second_argument_type</code>	1394
5.302	<code>std::_Maybe_wrap_member_pointer&lt; _Tp &gt; Struct</code>	
	Template Reference	1395
5.302.1	Detailed Description	1395
5.303	<code>std::_Maybe_wrap_member_pointer&lt; _Tp _Class::* &gt; Struct</code>	
	Template Reference	1395
5.303.1	Detailed Description	1396
5.304	<code>std::_Mem_fn&lt; _Res(_Class::*)(_ArgTypes...) const &gt; Class</code>	
	Template Reference	1396
5.304.1	Detailed Description	1397
5.305	<code>std::_Mem_fn&lt; _Res(_Class::*)(_ArgTypes...) const volatile &gt; Class</code>	
	Template Reference	1397
5.305.1	Detailed Description	1398
5.306	<code>std::_Mem_fn&lt; _Res(_Class::*)(_ArgTypes...) volatile &gt; Class</code>	
	Template Reference	1398
5.306.1	Detailed Description	1399

5.307std::_Mem_fn< _Res(_Class::*)(_ArgTypes...)> Class Template Reference . . . . .	1399
5.307.1 Detailed Description . . . . .	1400
5.308std::_Mu< _Arg, false, false > Class Template Reference . . . . .	1401
5.308.1 Detailed Description . . . . .	1401
5.309std::_Mu< _Arg, false, true > Class Template Reference . . . . .	1401
5.309.1 Detailed Description . . . . .	1401
5.310std::_Mu< _Arg, true, false > Class Template Reference . . . . .	1402
5.310.1 Detailed Description . . . . .	1402
5.311std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference . . . . .	1402
5.311.1 Detailed Description . . . . .	1403
5.312std::_Placeholder< _Num > Struct Template Reference . . . . .	1403
5.312.1 Detailed Description . . . . .	1403
5.313std::_Reference_wrapper_base< _Tp > Struct Template Reference . . . . .	1404
5.313.1 Detailed Description . . . . .	1404
5.314std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference . . . . .	1405
5.314.1 Detailed Description . . . . .	1405
5.315std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference . . . . .	1405
5.315.1 Detailed Description . . . . .	1406
5.316std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference . . . . .	1406
5.316.1 Detailed Description . . . . .	1406
5.317std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference . . . . .	1407
5.317.1 Detailed Description . . . . .	1408
5.317.2 Constructor & Destructor Documentation . . . . .	1408
5.317.2.1 _Temporary_buffer . . . . .	1408
5.317.3 Member Function Documentation . . . . .	1408
5.317.3.1 begin . . . . .	1408
5.317.3.2 end . . . . .	1408
5.317.3.3 requested_size . . . . .	1409

5.317.3.4 size . . . . .	1409
5.318std::_Tuple_impl< _Idx > Struct Template Reference . . . . .	1409
5.318.1 Detailed Description . . . . .	1409
5.319std::_Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference	1410
5.319.1 Detailed Description . . . . .	1411
5.320std::_Vector_base< _Tp, _Alloc > Struct Template Reference . . . . .	1411
5.320.1 Detailed Description . . . . .	1412
5.321std::_Weak_result_type< _Functor > Struct Template Reference . . . . .	1412
5.321.1 Detailed Description . . . . .	1412
5.322std::_Weak_result_type_impl< _Functor > Struct Template Reference	1413
5.322.1 Detailed Description . . . . .	1413
5.323std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Tem- plate Reference . . . . .	1413
5.323.1 Detailed Description . . . . .	1413
5.324std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)> Struct Tem- plate Reference . . . . .	1414
5.324.1 Detailed Description . . . . .	1414
5.325std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference . . . . .	1414
5.325.1 Detailed Description . . . . .	1414
5.326std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const > Struct Template Reference . . . . .	1415
5.326.1 Detailed Description . . . . .	1415
5.327std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile > Struct Template Reference . . . . .	1415
5.327.1 Detailed Description . . . . .	1416
5.328std::_Weak_result_type_impl<        _Res(_Class::*)(_ArgTypes...) volatile > Struct Template Reference . . . . .	1416
5.328.1 Detailed Description . . . . .	1416
5.329std::_Weak_result_type_impl<        _Res(_Class::*)(_ArgTypes...)> Struct Template Reference . . . . .	1416
5.329.1 Detailed Description . . . . .	1417
5.330std::add_lvalue_reference< _Tp > Struct Template Reference . . . . .	1417

---

5.330.1 Detailed Description . . . . .	1417
5.331std::add_rvalue_reference< _Tp > Struct Template Reference . . . . .	1418
5.331.1 Detailed Description . . . . .	1418
5.332std::adopt_lock_t Struct Reference . . . . .	1418
5.332.1 Detailed Description . . . . .	1418
5.333std::aligned_storage< _Len, _Align > Struct Template Reference . . . . .	1418
5.333.1 Detailed Description . . . . .	1419
5.334std::allocator< _Tp > Class Template Reference . . . . .	1419
5.334.1 Detailed Description . . . . .	1420
5.335std::allocator< void > Class Template Reference . . . . .	1420
5.335.1 Detailed Description . . . . .	1421
5.336std::atomic< _Tp > Struct Template Reference . . . . .	1421
5.336.1 Detailed Description . . . . .	1422
5.337std::atomic< _Tp * > Struct Template Reference . . . . .	1422
5.337.1 Detailed Description . . . . .	1423
5.338std::atomic< bool > Struct Template Reference . . . . .	1423
5.338.1 Detailed Description . . . . .	1423
5.339std::atomic< char > Struct Template Reference . . . . .	1424
5.339.1 Detailed Description . . . . .	1424
5.340std::atomic< char16_t > Struct Template Reference . . . . .	1424
5.340.1 Detailed Description . . . . .	1425
5.341std::atomic< char32_t > Struct Template Reference . . . . .	1425
5.341.1 Detailed Description . . . . .	1426
5.342std::atomic< int > Struct Template Reference . . . . .	1426
5.342.1 Detailed Description . . . . .	1427
5.343std::atomic< long > Struct Template Reference . . . . .	1427
5.343.1 Detailed Description . . . . .	1427
5.344std::atomic< long long > Struct Template Reference . . . . .	1428
5.344.1 Detailed Description . . . . .	1428
5.345std::atomic< short > Struct Template Reference . . . . .	1428
5.345.1 Detailed Description . . . . .	1429

---

---

5.346std::atomic< signed char > Struct Template Reference . . . . .	1429
5.346.1 Detailed Description . . . . .	1430
5.347std::atomic< unsigned char > Struct Template Reference . . . . .	1430
5.347.1 Detailed Description . . . . .	1431
5.348std::atomic< unsigned int > Struct Template Reference . . . . .	1431
5.348.1 Detailed Description . . . . .	1432
5.349std::atomic< unsigned long > Struct Template Reference . . . . .	1432
5.349.1 Detailed Description . . . . .	1432
5.350std::atomic< unsigned long long > Struct Template Reference . . . . .	1433
5.350.1 Detailed Description . . . . .	1433
5.351std::atomic< unsigned short > Struct Template Reference . . . . .	1433
5.351.1 Detailed Description . . . . .	1434
5.352std::atomic< void * > Struct Template Reference . . . . .	1434
5.352.1 Detailed Description . . . . .	1435
5.353std::atomic< wchar_t > Struct Template Reference . . . . .	1435
5.353.1 Detailed Description . . . . .	1436
5.354std::auto_ptr< _Tp > Class Template Reference . . . . .	1436
5.354.1 Detailed Description . . . . .	1437
5.354.2 Member Typedef Documentation . . . . .	1437
5.354.2.1 element_type . . . . .	1437
5.354.3 Constructor & Destructor Documentation . . . . .	1437
5.354.3.1 auto_ptr . . . . .	1437
5.354.3.2 auto_ptr . . . . .	1438
5.354.3.3 auto_ptr . . . . .	1438
5.354.3.4 ~auto_ptr . . . . .	1438
5.354.3.5 auto_ptr . . . . .	1439
5.354.4 Member Function Documentation . . . . .	1439
5.354.4.1 get . . . . .	1439
5.354.4.2 operator* . . . . .	1439
5.354.4.3 operator-> . . . . .	1440
5.354.4.4 operator= . . . . .	1440

---

5.354.4.5 operator= . . . . .	1440
5.354.4.6 release . . . . .	1440
5.354.4.7 reset . . . . .	1441
5.355std::auto_ptr_ref< _Tp1 > Struct Template Reference . . . . .	1441
5.355.1 Detailed Description . . . . .	1442
5.356std::back_insert_iterator< _Container > Class Template Reference . . . . .	1442
5.356.1 Detailed Description . . . . .	1443
5.356.2 Member Typedef Documentation . . . . .	1443
5.356.2.1 container_type . . . . .	1443
5.356.2.2 difference_type . . . . .	1444
5.356.2.3 iterator_category . . . . .	1444
5.356.2.4 pointer . . . . .	1444
5.356.2.5 reference . . . . .	1444
5.356.2.6 value_type . . . . .	1444
5.356.3 Constructor & Destructor Documentation . . . . .	1444
5.356.3.1 back_insert_iterator . . . . .	1444
5.356.4 Member Function Documentation . . . . .	1445
5.356.4.1 operator* . . . . .	1445
5.356.4.2 operator++ . . . . .	1445
5.356.4.3 operator++ . . . . .	1445
5.356.4.4 operator= . . . . .	1445
5.357std::bad_alloc Class Reference . . . . .	1446
5.357.1 Detailed Description . . . . .	1446
5.357.2 Member Function Documentation . . . . .	1446
5.357.2.1 what . . . . .	1446
5.358std::bad_cast Class Reference . . . . .	1447
5.358.1 Detailed Description . . . . .	1447
5.358.2 Member Function Documentation . . . . .	1448
5.358.2.1 what . . . . .	1448
5.359std::bad_exception Class Reference . . . . .	1448
5.359.1 Detailed Description . . . . .	1448

---

---

5.359.2 Member Function Documentation . . . . .	1449
5.359.2.1 what . . . . .	1449
5.360std::bad_function_call Class Reference . . . . .	1449
5.360.1 Detailed Description . . . . .	1449
5.360.2 Member Function Documentation . . . . .	1450
5.360.2.1 what . . . . .	1450
5.361std::bad_typeid Class Reference . . . . .	1450
5.361.1 Detailed Description . . . . .	1451
5.361.2 Member Function Documentation . . . . .	1451
5.361.2.1 what . . . . .	1451
5.362std::basic_filebuf< _CharT, _Traits > Class Template Reference . . . . .	1451
5.362.1 Detailed Description . . . . .	1455
5.362.2 Member Typedef Documentation . . . . .	1455
5.362.2.1 __streambuf_type . . . . .	1455
5.362.2.2 char_type . . . . .	1455
5.362.2.3 int_type . . . . .	1456
5.362.2.4 off_type . . . . .	1456
5.362.2.5 pos_type . . . . .	1456
5.362.2.6 traits_type . . . . .	1457
5.362.3 Constructor & Destructor Documentation . . . . .	1457
5.362.3.1 basic_filebuf . . . . .	1457
5.362.3.2 ~basic_filebuf . . . . .	1457
5.362.4 Member Function Documentation . . . . .	1457
5.362.4.1 _M_create_pback . . . . .	1457
5.362.4.2 _M_destroy_pback . . . . .	1458
5.362.4.3 _M_set_buffer . . . . .	1458
5.362.4.4 close . . . . .	1458
5.362.4.5 eback . . . . .	1459
5.362.4.6 egptr . . . . .	1459
5.362.4.7 epptr . . . . .	1459
5.362.4.8 gbump . . . . .	1460

---

5.362.4.9	getloc	1460
5.362.4.10	gptr	1460
5.362.4.11	limbue	1461
5.362.4.12	n_avail	1461
5.362.4.13	s_open	1462
5.362.4.14	open	1462
5.362.4.15	open	1463
5.362.4.16	overflow	1463
5.362.4.17	backfail	1464
5.362.4.18	base	1464
5.362.4.19	bump	1465
5.362.4.20	ptr	1465
5.362.4.21	pubimbue	1466
5.362.4.22	pubseekoff	1466
5.362.4.23	pubseekpos	1466
5.362.4.24	pubsetbuf	1467
5.362.4.25	pubsync	1467
5.362.4.26	sbumpc	1467
5.362.4.27	seekoff	1468
5.362.4.28	seekpos	1468
5.362.4.29	setbuf	1468
5.362.4.30	setbuf	1469
5.362.4.31	lsetg	1469
5.362.4.32	setp	1470
5.362.4.33	getc	1470
5.362.4.34	getn	1471
5.362.4.35	showmanyc	1471
5.362.4.36	nextc	1472
5.362.4.37	sputbackc	1472
5.362.4.38	putc	1472
5.362.4.39	sputn	1473

---

5.362.4.40	stossc . . . . .	1473
5.362.4.41	sungetc . . . . .	1474
5.362.4.42	sync . . . . .	1474
5.362.4.43	uflow . . . . .	1474
5.362.4.44	underflow . . . . .	1475
5.362.4.45	xsgetn . . . . .	1476
5.362.4.46	xspn . . . . .	1476
5.362.5	Member Data Documentation . . . . .	1477
5.362.5.1	_M_buf . . . . .	1477
5.362.5.2	_M_buf_locale . . . . .	1477
5.362.5.3	_M_buf_size . . . . .	1477
5.362.5.4	_M_ext_buf . . . . .	1478
5.362.5.5	_M_ext_buf_size . . . . .	1478
5.362.5.6	_M_ext_next . . . . .	1478
5.362.5.7	_M_in_beg . . . . .	1478
5.362.5.8	_M_in_cur . . . . .	1479
5.362.5.9	_M_in_end . . . . .	1479
5.362.5.10	_M_mode . . . . .	1479
5.362.5.11	_M_out_beg . . . . .	1480
5.362.5.12	_M_out_cur . . . . .	1480
5.362.5.13	_M_out_end . . . . .	1480
5.362.5.14	_M_pback . . . . .	1481
5.362.5.15	_M_pback_cur_save . . . . .	1481
5.362.5.16	_M_pback_end_save . . . . .	1481
5.362.5.17	_M_pback_init . . . . .	1481
5.362.5.18	_M_reading . . . . .	1482
5.363	std::basic_fstream<_CharT, _Traits > Class Template Reference . . . . .	1482
5.363.1	Detailed Description . . . . .	1490
5.363.2	Member Typedef Documentation . . . . .	1491
5.363.2.1	__ctype_type . . . . .	1491
5.363.2.2	__ctype_type . . . . .	1491

---

---

5.363.2.3	<code>__num_get_type</code>	1491
5.363.2.4	<code>__num_put_type</code>	1491
5.363.2.5	<code>__num_put_type</code>	1492
5.363.2.6	<code>char_type</code>	1492
5.363.2.7	<code>char_type</code>	1492
5.363.2.8	<code>event_callback</code>	1492
5.363.2.9	<code>fmtflags</code>	1493
5.363.2.10	<code>int_type</code>	1494
5.363.2.11	<code>int_type</code>	1494
5.363.2.12	<code>iostate</code>	1494
5.363.2.13	<code>off_type</code>	1495
5.363.2.14	<code>off_type</code>	1495
5.363.2.15	<code>openmode</code>	1495
5.363.2.16	<code>pos_type</code>	1496
5.363.2.17	<code>pos_type</code>	1496
5.363.2.18	<code>seekdir</code>	1496
5.363.2.19	<code>traits_type</code>	1496
5.363.2.20	<code>traits_type</code>	1497
5.363.3	Member Enumeration Documentation	1497
5.363.3.1	<code>event</code>	1497
5.363.4	Constructor & Destructor Documentation	1497
5.363.4.1	<code>basic_fstream</code>	1497
5.363.4.2	<code>basic_fstream</code>	1498
5.363.4.3	<code>basic_fstream</code>	1498
5.363.4.4	<code>~basic_fstream</code>	1498
5.363.5	Member Function Documentation	1499
5.363.5.1	<code>_M_getloc</code>	1499
5.363.5.2	<code>_M_write</code>	1499
5.363.5.3	<code>bad</code>	1499
5.363.5.4	<code>clear</code>	1500
5.363.5.5	<code>close</code>	1500

---

5.363.5.6 copyfmt . . . . .	1500
5.363.5.7 eof . . . . .	1501
5.363.5.8 exceptions . . . . .	1501
5.363.5.9 exceptions . . . . .	1502
5.363.5.10fail . . . . .	1502
5.363.5.11fill . . . . .	1503
5.363.5.12fill . . . . .	1503
5.363.5.13flags . . . . .	1503
5.363.5.14flags . . . . .	1504
5.363.5.15flush . . . . .	1504
5.363.5.16gcount . . . . .	1504
5.363.5.17get . . . . .	1505
5.363.5.18get . . . . .	1505
5.363.5.19get . . . . .	1506
5.363.5.20get . . . . .	1507
5.363.5.21get . . . . .	1507
5.363.5.22get . . . . .	1508
5.363.5.23getline . . . . .	1508
5.363.5.24getline . . . . .	1509
5.363.5.25getloc . . . . .	1509
5.363.5.26good . . . . .	1510
5.363.5.27ignore . . . . .	1510
5.363.5.28ignore . . . . .	1511
5.363.5.29ignore . . . . .	1511
5.363.5.30mbue . . . . .	1512
5.363.5.31limit . . . . .	1512
5.363.5.32is_open . . . . .	1513
5.363.5.33iword . . . . .	1513
5.363.5.34narrow . . . . .	1513
5.363.5.35open . . . . .	1514
5.363.5.36open . . . . .	1514

---

5.363.5.37operator void *	1515
5.363.5.38operator!	1515
5.363.5.39operator<<	1515
5.363.5.40operator<<	1515
5.363.5.41operator<<	1516
5.363.5.42operator<<	1516
5.363.5.43operator<<	1517
5.363.5.44operator<<	1517
5.363.5.45operator<<	1517
5.363.5.46operator<<	1517
5.363.5.47operator<<	1518
5.363.5.48operator<<	1518
5.363.5.49operator<<	1519
5.363.5.50operator<<	1519
5.363.5.51operator<<	1519
5.363.5.52operator<<	1520
5.363.5.53operator<<	1520
5.363.5.54operator<<	1521
5.363.5.55operator<<	1521
5.363.5.56operator>>	1522
5.363.5.57operator>>	1522
5.363.5.58operator>>	1522
5.363.5.59operator>>	1523
5.363.5.60operator>>	1523
5.363.5.61operator>>	1523
5.363.5.62operator>>	1524
5.363.5.63operator>>	1524
5.363.5.64operator>>	1524
5.363.5.65operator>>	1525
5.363.5.66operator>>	1525
5.363.5.67operator>>	1526

---

5.363.5.68operator>>	1526
5.363.5.69operator>>	1527
5.363.5.70operator>>	1527
5.363.5.71operator>>	1527
5.363.5.72operator>>	1528
5.363.5.73peek	1528
5.363.5.74precision	1529
5.363.5.75precision	1529
5.363.5.76put	1529
5.363.5.77putback	1530
5.363.5.78pword	1530
5.363.5.79rdbuf	1531
5.363.5.80rdbuf	1531
5.363.5.81rdstate	1532
5.363.5.82read	1532
5.363.5.83readsome	1533
5.363.5.84register_callback	1533
5.363.5.85seekg	1534
5.363.5.86seekg	1534
5.363.5.87seekp	1535
5.363.5.88seekp	1535
5.363.5.89setf	1536
5.363.5.90setf	1536
5.363.5.91setstate	1537
5.363.5.92sync	1537
5.363.5.93sync_with_stdio	1538
5.363.5.94tellg	1538
5.363.5.95tellp	1539
5.363.5.96tie	1539
5.363.5.97tie	1540
5.363.5.98unget	1540

---

5.363.5.9	unsetf . . . . .	1540
5.363.5.10	iden . . . . .	1541
5.363.5.10	width . . . . .	1541
5.363.5.10	width . . . . .	1542
5.363.5.10	write . . . . .	1542
5.363.5.10	alloc . . . . .	1543
5.363.6	Member Data Documentation . . . . .	1543
5.363.6.1	_M_gcount . . . . .	1543
5.363.6.2	adjustfield . . . . .	1543
5.363.6.3	app . . . . .	1543
5.363.6.4	ate . . . . .	1544
5.363.6.5	badbit . . . . .	1544
5.363.6.6	basefield . . . . .	1544
5.363.6.7	beg . . . . .	1544
5.363.6.8	binary . . . . .	1544
5.363.6.9	boolalpha . . . . .	1545
5.363.6.10	cur . . . . .	1545
5.363.6.11	dec . . . . .	1545
5.363.6.12	end . . . . .	1545
5.363.6.13	eofbit . . . . .	1545
5.363.6.14	failbit . . . . .	1546
5.363.6.15	fixed . . . . .	1546
5.363.6.16	floatfield . . . . .	1546
5.363.6.17	goodbit . . . . .	1546
5.363.6.18	hex . . . . .	1547
5.363.6.19	in . . . . .	1547
5.363.6.20	internal . . . . .	1547
5.363.6.21	left . . . . .	1547
5.363.6.22	oct . . . . .	1548
5.363.6.23	out . . . . .	1548
5.363.6.24	right . . . . .	1548

---

---

5.363.6.25	scientific	1548
5.363.6.26	showbase	1548
5.363.6.27	showpoint	1548
5.363.6.28	showpos	1548
5.363.6.29	skipws	1549
5.363.6.30	trunc	1549
5.363.6.31	unitbuf	1549
5.363.6.32	uppercase	1549
5.364	std::basic_ifstream<_CharT, _Traits> Class Template Reference	1549
5.364.1	Detailed Description	1556
5.364.2	Member Typedef Documentation	1556
5.364.2.1	__ctype_type	1556
5.364.2.2	__num_get_type	1556
5.364.2.3	__num_put_type	1557
5.364.2.4	char_type	1557
5.364.2.5	event_callback	1557
5.364.2.6	fmtflags	1557
5.364.2.7	int_type	1558
5.364.2.8	iostate	1559
5.364.2.9	off_type	1559
5.364.2.10	openmode	1559
5.364.2.11	lpos_type	1560
5.364.2.12	seekdir	1560
5.364.2.13	traits_type	1560
5.364.3	Member Enumeration Documentation	1560
5.364.3.1	event	1560
5.364.4	Constructor & Destructor Documentation	1561
5.364.4.1	basic_ifstream	1561
5.364.4.2	basic_ifstream	1561
5.364.4.3	basic_ifstream	1561
5.364.4.4	~basic_ifstream	1562

---

5.364.5 Member Function Documentation . . . . .	1562
5.364.5.1 _M_getloc . . . . .	1562
5.364.5.2 bad . . . . .	1562
5.364.5.3 clear . . . . .	1563
5.364.5.4 close . . . . .	1563
5.364.5.5 copyfmt . . . . .	1563
5.364.5.6 eof . . . . .	1564
5.364.5.7 exceptions . . . . .	1564
5.364.5.8 exceptions . . . . .	1564
5.364.5.9 fail . . . . .	1565
5.364.5.10fill . . . . .	1565
5.364.5.11fill . . . . .	1566
5.364.5.12flags . . . . .	1566
5.364.5.13flags . . . . .	1567
5.364.5.14gcount . . . . .	1567
5.364.5.15get . . . . .	1567
5.364.5.16get . . . . .	1568
5.364.5.17get . . . . .	1568
5.364.5.18get . . . . .	1569
5.364.5.19get . . . . .	1570
5.364.5.20get . . . . .	1570
5.364.5.21getline . . . . .	1571
5.364.5.22getline . . . . .	1571
5.364.5.23getloc . . . . .	1572
5.364.5.24good . . . . .	1572
5.364.5.25ignore . . . . .	1573
5.364.5.26ignore . . . . .	1573
5.364.5.27ignore . . . . .	1574
5.364.5.28mbue . . . . .	1574
5.364.5.29mit . . . . .	1575
5.364.5.30s_open . . . . .	1575

---

5.364.5.31word	1575
5.364.5.32narrow	1576
5.364.5.33open	1576
5.364.5.34open	1577
5.364.5.35operator void *	1577
5.364.5.36operator!	1577
5.364.5.37operator>>	1577
5.364.5.38operator>>	1578
5.364.5.39operator>>	1578
5.364.5.40operator>>	1578
5.364.5.41operator>>	1579
5.364.5.42operator>>	1579
5.364.5.43operator>>	1579
5.364.5.44operator>>	1580
5.364.5.45operator>>	1580
5.364.5.46operator>>	1581
5.364.5.47operator>>	1581
5.364.5.48operator>>	1582
5.364.5.49operator>>	1582
5.364.5.50operator>>	1583
5.364.5.51operator>>	1583
5.364.5.52operator>>	1583
5.364.5.53operator>>	1584
5.364.5.54peek	1584
5.364.5.55precision	1584
5.364.5.56precision	1585
5.364.5.57putback	1585
5.364.5.58pword	1586
5.364.5.59rdbuf	1586
5.364.5.60rdbuf	1587
5.364.5.61rdstate	1587

---

5.364.5.62	read	1587
5.364.5.63	readsome	1588
5.364.5.64	register_callback	1589
5.364.5.65	seekg	1589
5.364.5.66	seekg	1590
5.364.5.67	setf	1590
5.364.5.68	setf	1591
5.364.5.69	setstate	1591
5.364.5.70	sync	1592
5.364.5.71	sync_with_stdio	1592
5.364.5.72	tellg	1593
5.364.5.73	tie	1593
5.364.5.74	tie	1594
5.364.5.75	unsetf	1594
5.364.5.76	unsetf	1594
5.364.5.77	widen	1595
5.364.5.78	width	1595
5.364.5.79	width	1596
5.364.5.80	xalloc	1596
5.364.6	Member Data Documentation	1596
5.364.6.1	_M_gcount	1596
5.364.6.2	adjustfield	1597
5.364.6.3	app	1597
5.364.6.4	ate	1597
5.364.6.5	badbit	1597
5.364.6.6	basefield	1597
5.364.6.7	beg	1598
5.364.6.8	binary	1598
5.364.6.9	boolalpha	1598
5.364.6.10	cur	1598
5.364.6.11	ldec	1598

---

5.364.6.12end	1598
5.364.6.13eofbit	1599
5.364.6.14failbit	1599
5.364.6.15fixed	1599
5.364.6.16floatfield	1599
5.364.6.17goodbit	1600
5.364.6.18hex	1600
5.364.6.19n	1600
5.364.6.20internal	1600
5.364.6.21left	1601
5.364.6.22oct	1601
5.364.6.23out	1601
5.364.6.24right	1601
5.364.6.25scientific	1601
5.364.6.26showbase	1601
5.364.6.27showpoint	1602
5.364.6.28showpos	1602
5.364.6.29skipws	1602
5.364.6.30trunc	1602
5.364.6.31unitbuf	1602
5.364.6.32uppercase	1602
5.365std::basic_ios<_CharT, _Traits> Class Template Reference	1603
5.365.1 Detailed Description	1606
5.365.2 Member Typedef Documentation	1607
5.365.2.1 __ctype_type	1607
5.365.2.2 __num_get_type	1607
5.365.2.3 __num_put_type	1607
5.365.2.4 char_type	1607
5.365.2.5 event_callback	1608
5.365.2.6 fmtflags	1608
5.365.2.7 int_type	1609

---

5.365.2.8	iostate	1609
5.365.2.9	off_type	1610
5.365.2.10	openmode	1610
5.365.2.11	lpos_type	1611
5.365.2.12	seekdir	1611
5.365.2.13	traits_type	1611
5.365.3	Member Enumeration Documentation	1612
5.365.3.1	event	1612
5.365.4	Constructor & Destructor Documentation	1612
5.365.4.1	basic_ios	1612
5.365.4.2	~basic_ios	1612
5.365.4.3	basic_ios	1612
5.365.5	Member Function Documentation	1613
5.365.5.1	_M_getloc	1613
5.365.5.2	bad	1613
5.365.5.3	clear	1613
5.365.5.4	copyfmt	1614
5.365.5.5	eof	1614
5.365.5.6	exceptions	1614
5.365.5.7	exceptions	1615
5.365.5.8	fail	1615
5.365.5.9	fill	1616
5.365.5.10	fill	1616
5.365.5.11	lflags	1617
5.365.5.12	rflags	1617
5.365.5.13	getloc	1617
5.365.5.14	good	1618
5.365.5.15	imbue	1618
5.365.5.16	init	1618
5.365.5.17	word	1619
5.365.5.18	narrow	1619

---

5.365.5.19operator void *	1620
5.365.5.20operator!	1620
5.365.5.21precision	1620
5.365.5.22precision	1620
5.365.5.23pword	1621
5.365.5.24rdbuf	1621
5.365.5.25rdbuf	1622
5.365.5.26rdstate	1622
5.365.5.27register_callback	1623
5.365.5.28setf	1623
5.365.5.29setf	1623
5.365.5.30setstate	1624
5.365.5.31sync_with_stdio	1624
5.365.5.32tie	1625
5.365.5.33tie	1625
5.365.5.34unsetf	1625
5.365.5.35widen	1626
5.365.5.36width	1626
5.365.5.37width	1627
5.365.5.38&alloc	1627
5.365.6 Member Data Documentation	1627
5.365.6.1 adjustfield	1627
5.365.6.2 app	1627
5.365.6.3 ate	1628
5.365.6.4 badbit	1628
5.365.6.5 basefield	1628
5.365.6.6 beg	1628
5.365.6.7 binary	1628
5.365.6.8 boolalpha	1629
5.365.6.9 cur	1629
5.365.6.10dec	1629

5.365.6.1	<code>lend</code>	1629
5.365.6.12	<code>eofbit</code>	1629
5.365.6.13	<code>failbit</code>	1630
5.365.6.14	<code>fixed</code>	1630
5.365.6.15	<code>floatfield</code>	1630
5.365.6.16	<code>goodbit</code>	1630
5.365.6.17	<code>hex</code>	1631
5.365.6.18	<code>n</code>	1631
5.365.6.19	<code>internal</code>	1631
5.365.6.20	<code>left</code>	1631
5.365.6.21	<code>loct</code>	1632
5.365.6.22	<code>out</code>	1632
5.365.6.23	<code>right</code>	1632
5.365.6.24	<code>scientific</code>	1632
5.365.6.25	<code>showbase</code>	1632
5.365.6.26	<code>showpoint</code>	1632
5.365.6.27	<code>showpos</code>	1632
5.365.6.28	<code>skipws</code>	1633
5.365.6.29	<code>trunc</code>	1633
5.365.6.30	<code>unitbuf</code>	1633
5.365.6.3	<code>uppercase</code>	1633
5.366	<code>std::basic_istream&lt;_CharT, _Traits&gt;</code> Class Template Reference	1633
5.366.1	Detailed Description	1641
5.366.2	Member Typedef Documentation	1641
5.366.2.1	<code>__ctype_type</code>	1641
5.366.2.2	<code>__ctype_type</code>	1641
5.366.2.3	<code>__num_get_type</code>	1641
5.366.2.4	<code>__num_put_type</code>	1642
5.366.2.5	<code>__num_put_type</code>	1642
5.366.2.6	<code>char_type</code>	1642
5.366.2.7	<code>char_type</code>	1642

---

5.366.2.8 event_callback . . . . .	1643
5.366.2.9 fmtflags . . . . .	1643
5.366.2.10int_type . . . . .	1644
5.366.2.11int_type . . . . .	1644
5.366.2.12istate . . . . .	1644
5.366.2.13off_type . . . . .	1645
5.366.2.14off_type . . . . .	1645
5.366.2.15openmode . . . . .	1645
5.366.2.16pos_type . . . . .	1646
5.366.2.17pos_type . . . . .	1646
5.366.2.18seekdir . . . . .	1646
5.366.2.19traits_type . . . . .	1647
5.366.2.20traits_type . . . . .	1647
5.366.3 Member Enumeration Documentation . . . . .	1647
5.366.3.1 event . . . . .	1647
5.366.4 Constructor & Destructor Documentation . . . . .	1648
5.366.4.1 basic_istream . . . . .	1648
5.366.4.2 ~basic_istream . . . . .	1648
5.366.5 Member Function Documentation . . . . .	1648
5.366.5.1 _M_getloc . . . . .	1648
5.366.5.2 _M_write . . . . .	1649
5.366.5.3 bad . . . . .	1649
5.366.5.4 clear . . . . .	1649
5.366.5.5 copyfmt . . . . .	1650
5.366.5.6 eof . . . . .	1650
5.366.5.7 exceptions . . . . .	1651
5.366.5.8 exceptions . . . . .	1651
5.366.5.9 fail . . . . .	1652
5.366.5.10fill . . . . .	1652
5.366.5.11fill . . . . .	1652
5.366.5.12flags . . . . .	1653

---

5.366.5.13	flags	1653
5.366.5.14	flush	1653
5.366.5.15	gcount	1654
5.366.5.16	get	1654
5.366.5.17	get	1655
5.366.5.18	get	1655
5.366.5.19	get	1656
5.366.5.20	get	1656
5.366.5.21	get	1657
5.366.5.22	getline	1658
5.366.5.23	getline	1658
5.366.5.24	getloc	1659
5.366.5.25	good	1659
5.366.5.26	ignore	1660
5.366.5.27	ignore	1660
5.366.5.28	ignore	1661
5.366.5.29	imbue	1661
5.366.5.30	init	1662
5.366.5.31	word	1662
5.366.5.32	narrow	1662
5.366.5.33	operator void *	1663
5.366.5.34	operator!	1663
5.366.5.35	operator<<	1663
5.366.5.36	operator<<	1664
5.366.5.37	operator<<	1664
5.366.5.38	operator<<	1665
5.366.5.39	operator<<	1665
5.366.5.40	operator<<	1665
5.366.5.41	operator<<	1665
5.366.5.42	operator<<	1666
5.366.5.43	operator<<	1666

---

5.366.5.44operator<<	1667
5.366.5.45operator<<	1667
5.366.5.46operator<<	1667
5.366.5.47operator<<	1668
5.366.5.48operator<<	1668
5.366.5.49operator<<	1669
5.366.5.50operator<<	1669
5.366.5.51operator<<	1670
5.366.5.52operator>>	1670
5.366.5.53operator>>	1670
5.366.5.54operator>>	1671
5.366.5.55operator>>	1671
5.366.5.56operator>>	1672
5.366.5.57operator>>	1672
5.366.5.58operator>>	1672
5.366.5.59operator>>	1673
5.366.5.60operator>>	1673
5.366.5.61operator>>	1674
5.366.5.62operator>>	1674
5.366.5.63operator>>	1674
5.366.5.64operator>>	1675
5.366.5.65operator>>	1675
5.366.5.66operator>>	1675
5.366.5.67operator>>	1676
5.366.5.68operator>>	1676
5.366.5.69peek	1677
5.366.5.70precision	1677
5.366.5.71precision	1677
5.366.5.72put	1678
5.366.5.73putback	1678
5.366.5.74pword	1679

---

5.366.5.75	rdbuf	1679
5.366.5.76	rdbuf	1680
5.366.5.77	rdstate	1681
5.366.5.78	read	1681
5.366.5.79	readsome	1682
5.366.5.80	register_callback	1682
5.366.5.81	seekg	1683
5.366.5.82	seekg	1683
5.366.5.83	seekp	1684
5.366.5.84	seekp	1684
5.366.5.85	setf	1685
5.366.5.86	setf	1685
5.366.5.87	setstate	1686
5.366.5.88	sync	1686
5.366.5.89	sync_with_stdio	1687
5.366.5.90	tellg	1687
5.366.5.91	ltellp	1688
5.366.5.92	tie	1688
5.366.5.93	tie	1688
5.366.5.94	unget	1689
5.366.5.95	unsetf	1689
5.366.5.96	widen	1690
5.366.5.97	width	1690
5.366.5.98	width	1690
5.366.5.99	write	1691
5.366.5.100	alloc	1691
5.366.6	Member Data Documentation	1692
5.366.6.1	_M_gcount	1692
5.366.6.2	adjustfield	1692
5.366.6.3	app	1692
5.366.6.4	ate	1692

---

5.366.6.5 badbit . . . . .	1692
5.366.6.6 basefield . . . . .	1693
5.366.6.7 beg . . . . .	1693
5.366.6.8 binary . . . . .	1693
5.366.6.9 boolalpha . . . . .	1693
5.366.6.10cur . . . . .	1694
5.366.6.11dec . . . . .	1694
5.366.6.12end . . . . .	1694
5.366.6.13eofbit . . . . .	1694
5.366.6.14failbit . . . . .	1694
5.366.6.15fixed . . . . .	1695
5.366.6.16floatfield . . . . .	1695
5.366.6.17goodbit . . . . .	1695
5.366.6.18hex . . . . .	1695
5.366.6.19n . . . . .	1696
5.366.6.20internal . . . . .	1696
5.366.6.21left . . . . .	1696
5.366.6.22oct . . . . .	1696
5.366.6.23out . . . . .	1696
5.366.6.24right . . . . .	1697
5.366.6.25scientific . . . . .	1697
5.366.6.26showbase . . . . .	1697
5.366.6.27showpoint . . . . .	1697
5.366.6.28showpos . . . . .	1697
5.366.6.29skipws . . . . .	1697
5.366.6.30trunc . . . . .	1697
5.366.6.31unitbuf . . . . .	1698
5.366.6.32uppercase . . . . .	1698
5.367std::basic_istream<_CharT, _Traits > Class Template Reference . .	1698
5.367.1 Detailed Description . . . . .	1704
5.367.2 Member Typedef Documentation . . . . .	1704

---

---

5.367.2.1	<code>__ctype_type</code>	1704
5.367.2.2	<code>__num_get_type</code>	1705
5.367.2.3	<code>__num_put_type</code>	1705
5.367.2.4	<code>char_type</code>	1705
5.367.2.5	<code>event_callback</code>	1705
5.367.2.6	<code>fmtflags</code>	1706
5.367.2.7	<code>int_type</code>	1707
5.367.2.8	<code>iostate</code>	1707
5.367.2.9	<code>off_type</code>	1707
5.367.2.10	<code>openmode</code>	1707
5.367.2.11	<code>lpos_type</code>	1708
5.367.2.12	<code>seekdir</code>	1708
5.367.2.13	<code>traits_type</code>	1709
5.367.3	Member Enumeration Documentation	1709
5.367.3.1	<code>event</code>	1709
5.367.4	Constructor & Destructor Documentation	1709
5.367.4.1	<code>basic_istream</code>	1709
5.367.4.2	<code>~basic_istream</code>	1709
5.367.5	Member Function Documentation	1710
5.367.5.1	<code>_M_getloc</code>	1710
5.367.5.2	<code>bad</code>	1710
5.367.5.3	<code>clear</code>	1710
5.367.5.4	<code>copyfmt</code>	1711
5.367.5.5	<code>eof</code>	1711
5.367.5.6	<code>exceptions</code>	1712
5.367.5.7	<code>exceptions</code>	1712
5.367.5.8	<code>fail</code>	1713
5.367.5.9	<code>fill</code>	1713
5.367.5.10	<code>fill</code>	1713
5.367.5.11	<code>lflags</code>	1714
5.367.5.12	<code>rflags</code>	1714

---

---

5.367.5.13	gcount	1714
5.367.5.14	get	1715
5.367.5.15	get	1715
5.367.5.16	get	1716
5.367.5.17	get	1716
5.367.5.18	get	1717
5.367.5.19	get	1718
5.367.5.20	getline	1718
5.367.5.21	getline	1719
5.367.5.22	getloc	1719
5.367.5.23	good	1720
5.367.5.24	ignore	1720
5.367.5.25	ignore	1721
5.367.5.26	ignore	1721
5.367.5.27	imbue	1721
5.367.5.28	init	1722
5.367.5.29	inword	1722
5.367.5.30	narrow	1723
5.367.5.31	operator void *	1723
5.367.5.32	operator!	1723
5.367.5.33	operator>>	1724
5.367.5.34	operator>>	1724
5.367.5.35	operator>>	1725
5.367.5.36	operator>>	1725
5.367.5.37	operator>>	1725
5.367.5.38	operator>>	1726
5.367.5.39	operator>>	1726
5.367.5.40	operator>>	1726
5.367.5.41	operator>>	1727
5.367.5.42	operator>>	1727
5.367.5.43	operator>>	1727

---

5.367.5.44operator>>	1728
5.367.5.45operator>>	1728
5.367.5.46operator>>	1728
5.367.5.47operator>>	1729
5.367.5.48operator>>	1729
5.367.5.49operator>>	1730
5.367.5.50peek	1730
5.367.5.51precision	1730
5.367.5.52precision	1731
5.367.5.53putback	1731
5.367.5.54pword	1732
5.367.5.55rdbuf	1732
5.367.5.56rdbuf	1733
5.367.5.57rdstate	1734
5.367.5.58read	1734
5.367.5.59readsome	1735
5.367.5.60register_callback	1735
5.367.5.61seekg	1736
5.367.5.62seekg	1736
5.367.5.63setf	1737
5.367.5.64setf	1737
5.367.5.65setstate	1738
5.367.5.66sync	1738
5.367.5.67sync_with_stdio	1739
5.367.5.68tellg	1739
5.367.5.69tie	1740
5.367.5.70tie	1740
5.367.5.71lunget	1740
5.367.5.72unsetf	1741
5.367.5.73widen	1741
5.367.5.74width	1742

---

5.367.5.75width . . . . .	1742
5.367.5.76xalloc . . . . .	1742
5.367.6 Member Data Documentation . . . . .	1743
5.367.6.1 _M_gcount . . . . .	1743
5.367.6.2 adjustfield . . . . .	1743
5.367.6.3 app . . . . .	1743
5.367.6.4 ate . . . . .	1743
5.367.6.5 badbit . . . . .	1744
5.367.6.6 basefield . . . . .	1744
5.367.6.7 beg . . . . .	1744
5.367.6.8 binary . . . . .	1744
5.367.6.9 boolalpha . . . . .	1745
5.367.6.10cur . . . . .	1745
5.367.6.11dec . . . . .	1745
5.367.6.12end . . . . .	1745
5.367.6.13eofbit . . . . .	1745
5.367.6.14failbit . . . . .	1746
5.367.6.15fixed . . . . .	1746
5.367.6.16floatfield . . . . .	1746
5.367.6.17goodbit . . . . .	1746
5.367.6.18hex . . . . .	1747
5.367.6.19n . . . . .	1747
5.367.6.20internal . . . . .	1747
5.367.6.21left . . . . .	1747
5.367.6.22oct . . . . .	1747
5.367.6.23out . . . . .	1747
5.367.6.24right . . . . .	1748
5.367.6.25scientific . . . . .	1748
5.367.6.26showbase . . . . .	1748
5.367.6.27showpoint . . . . .	1748
5.367.6.28showpos . . . . .	1748

---

5.367.6.29skipws . . . . .	1748
5.367.6.30trunc . . . . .	1749
5.367.6.31unitbuf . . . . .	1749
5.367.6.32uppercase . . . . .	1749
5.368std::basic_istream< _CharT, _Traits >::sentry Class Reference . . . . .	1749
5.368.1 Detailed Description . . . . .	1750
5.368.2 Member Typedef Documentation . . . . .	1750
5.368.2.1 traits_type . . . . .	1750
5.368.3 Constructor & Destructor Documentation . . . . .	1750
5.368.3.1 sentry . . . . .	1750
5.368.4 Member Function Documentation . . . . .	1751
5.368.4.1 operator bool . . . . .	1751
5.369std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	1751
5.369.1 Detailed Description . . . . .	1758
5.369.2 Member Typedef Documentation . . . . .	1758
5.369.2.1 __ctype_type . . . . .	1758
5.369.2.2 __num_get_type . . . . .	1758
5.369.2.3 __num_put_type . . . . .	1758
5.369.2.4 char_type . . . . .	1759
5.369.2.5 event_callback . . . . .	1759
5.369.2.6 fmtflags . . . . .	1759
5.369.2.7 int_type . . . . .	1760
5.369.2.8 iostate . . . . .	1760
5.369.2.9 off_type . . . . .	1761
5.369.2.10openmode . . . . .	1761
5.369.2.11lpos_type . . . . .	1762
5.369.2.12seekdir . . . . .	1762
5.369.2.13traits_type . . . . .	1762
5.369.3 Member Enumeration Documentation . . . . .	1762
5.369.3.1 event . . . . .	1762

---

---

5.369.4 Constructor & Destructor Documentation . . . . .	1763
5.369.4.1 basic_istream . . . . .	1763
5.369.4.2 basic_istream . . . . .	1763
5.369.4.3 ~basic_istream . . . . .	1764
5.369.5 Member Function Documentation . . . . .	1764
5.369.5.1 _M_getloc . . . . .	1764
5.369.5.2 bad . . . . .	1764
5.369.5.3 clear . . . . .	1765
5.369.5.4 copyfmt . . . . .	1765
5.369.5.5 eof . . . . .	1765
5.369.5.6 exceptions . . . . .	1766
5.369.5.7 exceptions . . . . .	1766
5.369.5.8 fail . . . . .	1767
5.369.5.9 fill . . . . .	1767
5.369.5.10fill . . . . .	1768
5.369.5.11flags . . . . .	1768
5.369.5.12flags . . . . .	1768
5.369.5.13gcount . . . . .	1769
5.369.5.14get . . . . .	1769
5.369.5.15get . . . . .	1769
5.369.5.16get . . . . .	1770
5.369.5.17get . . . . .	1770
5.369.5.18get . . . . .	1771
5.369.5.19get . . . . .	1772
5.369.5.20getline . . . . .	1772
5.369.5.21getline . . . . .	1773
5.369.5.22getloc . . . . .	1774
5.369.5.23good . . . . .	1774
5.369.5.24ignore . . . . .	1774
5.369.5.25ignore . . . . .	1775
5.369.5.26ignore . . . . .	1776

---

5.369.5.27mbue . . . . .	1776
5.369.5.28nit . . . . .	1777
5.369.5.29word . . . . .	1777
5.369.5.30narrow . . . . .	1777
5.369.5.31operator void * . . . . .	1778
5.369.5.32operator! . . . . .	1778
5.369.5.33operator>> . . . . .	1778
5.369.5.34operator>> . . . . .	1779
5.369.5.35operator>> . . . . .	1779
5.369.5.36operator>> . . . . .	1779
5.369.5.37operator>> . . . . .	1780
5.369.5.38operator>> . . . . .	1780
5.369.5.39operator>> . . . . .	1781
5.369.5.40operator>> . . . . .	1781
5.369.5.41operator>> . . . . .	1782
5.369.5.42operator>> . . . . .	1782
5.369.5.43operator>> . . . . .	1782
5.369.5.44operator>> . . . . .	1783
5.369.5.45operator>> . . . . .	1783
5.369.5.46operator>> . . . . .	1783
5.369.5.47operator>> . . . . .	1784
5.369.5.48operator>> . . . . .	1784
5.369.5.49operator>> . . . . .	1785
5.369.5.50peek . . . . .	1785
5.369.5.51precision . . . . .	1785
5.369.5.52precision . . . . .	1786
5.369.5.53putback . . . . .	1786
5.369.5.54pword . . . . .	1787
5.369.5.55rdbuf . . . . .	1787
5.369.5.56rdbuf . . . . .	1787
5.369.5.57rdstate . . . . .	1788

---

---

5.369.5.58	read	1788
5.369.5.59	readsome	1789
5.369.5.60	register_callback	1790
5.369.5.61	seekg	1790
5.369.5.62	seekg	1791
5.369.5.63	setf	1791
5.369.5.64	setf	1792
5.369.5.65	setstate	1792
5.369.5.66	str	1793
5.369.5.67	str	1793
5.369.5.68	sync	1793
5.369.5.69	sync_with_stdio	1794
5.369.5.70	tellg	1794
5.369.5.71	tie	1795
5.369.5.72	tie	1795
5.369.5.73	unget	1795
5.369.5.74	unsetf	1796
5.369.5.75	widen	1796
5.369.5.76	width	1797
5.369.5.77	width	1797
5.369.5.78	xalloc	1797
5.369.6	Member Data Documentation	1798
5.369.6.1	_M_gcount	1798
5.369.6.2	adjustfield	1798
5.369.6.3	app	1798
5.369.6.4	ate	1798
5.369.6.5	badbit	1799
5.369.6.6	basefield	1799
5.369.6.7	beg	1799
5.369.6.8	binary	1799
5.369.6.9	boolalpha	1800

---

5.369.6.10	<code>cur</code>	1800
5.369.6.11	<code>dec</code>	1800
5.369.6.12	<code>end</code>	1800
5.369.6.13	<code>eofbit</code>	1800
5.369.6.14	<code>failbit</code>	1801
5.369.6.15	<code>fixed</code>	1801
5.369.6.16	<code>floatfield</code>	1801
5.369.6.17	<code>goodbit</code>	1801
5.369.6.18	<code>hex</code>	1802
5.369.6.19	<code>n</code>	1802
5.369.6.20	<code>internal</code>	1802
5.369.6.21	<code>left</code>	1802
5.369.6.22	<code>oct</code>	1802
5.369.6.23	<code>out</code>	1803
5.369.6.24	<code>right</code>	1803
5.369.6.25	<code>scientific</code>	1803
5.369.6.26	<code>showbase</code>	1803
5.369.6.27	<code>showpoint</code>	1803
5.369.6.28	<code>showpos</code>	1803
5.369.6.29	<code>skipws</code>	1804
5.369.6.30	<code>trunc</code>	1804
5.369.6.31	<code>unitbuf</code>	1804
5.369.6.32	<code>uppercase</code>	1804
5.370	<code>std::basic_ofstream&lt;_CharT, _Traits&gt;</code> Class Template Reference	1804
5.370.1	Detailed Description	1810
5.370.2	Member Typedef Documentation	1811
5.370.2.1	<code>__ctype_type</code>	1811
5.370.2.2	<code>__num_get_type</code>	1811
5.370.2.3	<code>__num_put_type</code>	1811
5.370.2.4	<code>char_type</code>	1811
5.370.2.5	<code>event_callback</code>	1812

---

---

5.370.2.6	fmtflags	1812
5.370.2.7	int_type	1813
5.370.2.8	iostate	1813
5.370.2.9	off_type	1813
5.370.2.10	openmode	1814
5.370.2.11	pos_type	1814
5.370.2.12	seekdir	1814
5.370.2.13	traits_type	1815
5.370.3	Member Enumeration Documentation	1815
5.370.3.1	event	1815
5.370.4	Constructor & Destructor Documentation	1815
5.370.4.1	basic_ofstream	1815
5.370.4.2	basic_ofstream	1815
5.370.4.3	basic_ofstream	1816
5.370.4.4	~basic_ofstream	1816
5.370.5	Member Function Documentation	1816
5.370.5.1	_M_getloc	1816
5.370.5.2	_M_write	1817
5.370.5.3	bad	1817
5.370.5.4	clear	1818
5.370.5.5	close	1818
5.370.5.6	copyfmt	1818
5.370.5.7	eof	1819
5.370.5.8	exceptions	1819
5.370.5.9	exceptions	1819
5.370.5.10	fail	1820
5.370.5.11	fill	1820
5.370.5.12	fill	1821
5.370.5.13	flags	1821
5.370.5.14	flags	1821
5.370.5.15	flush	1822

---

5.370.5.16	getloc . . . . .	1822
5.370.5.17	good . . . . .	1823
5.370.5.18	mbue . . . . .	1823
5.370.5.19	nit . . . . .	1823
5.370.5.20	is_open . . . . .	1824
5.370.5.21	word . . . . .	1824
5.370.5.22	narrow . . . . .	1824
5.370.5.23	open . . . . .	1825
5.370.5.24	open . . . . .	1825
5.370.5.25	operator void * . . . . .	1826
5.370.5.26	operator! . . . . .	1826
5.370.5.27	operator<< . . . . .	1826
5.370.5.28	operator<< . . . . .	1826
5.370.5.29	operator<< . . . . .	1827
5.370.5.30	operator<< . . . . .	1827
5.370.5.31	operator<< . . . . .	1828
5.370.5.32	operator<< . . . . .	1828
5.370.5.33	operator<< . . . . .	1828
5.370.5.34	operator<< . . . . .	1829
5.370.5.35	operator<< . . . . .	1829
5.370.5.36	operator<< . . . . .	1830
5.370.5.37	operator<< . . . . .	1830
5.370.5.38	operator<< . . . . .	1830
5.370.5.39	operator<< . . . . .	1831
5.370.5.40	operator<< . . . . .	1831
5.370.5.41	operator<< . . . . .	1832
5.370.5.42	operator<< . . . . .	1832
5.370.5.43	operator<< . . . . .	1832
5.370.5.44	precision . . . . .	1833
5.370.5.45	precision . . . . .	1833
5.370.5.46	put . . . . .	1833

---

5.370.5.47	pwd	1834
5.370.5.48	rdbuf	1834
5.370.5.49	rdbuf	1835
5.370.5.50	rstate	1835
5.370.5.51	register_callback	1835
5.370.5.52	seekp	1836
5.370.5.53	seekp	1836
5.370.5.54	setf	1837
5.370.5.55	setf	1837
5.370.5.56	setstate	1838
5.370.5.57	sync_with_stdio	1838
5.370.5.58	tellp	1839
5.370.5.59	tie	1839
5.370.5.60	tie	1839
5.370.5.61	unsetf	1840
5.370.5.62	widen	1840
5.370.5.63	width	1841
5.370.5.64	width	1841
5.370.5.65	write	1841
5.370.5.66	xalloc	1842
5.370.6	Member Data Documentation	1842
5.370.6.1	adjustfield	1842
5.370.6.2	app	1842
5.370.6.3	ate	1843
5.370.6.4	badbit	1843
5.370.6.5	basefield	1843
5.370.6.6	beg	1843
5.370.6.7	binary	1843
5.370.6.8	boolalpha	1844
5.370.6.9	cur	1844
5.370.6.10	dec	1844

---

5.370.6.1	lend	1844
5.370.6.12	eofbit	1844
5.370.6.13	failbit	1845
5.370.6.14	fixed	1845
5.370.6.15	floatfield	1845
5.370.6.16	goodbit	1845
5.370.6.17	hex	1846
5.370.6.18	n	1846
5.370.6.19	internal	1846
5.370.6.20	left	1846
5.370.6.21	loct	1847
5.370.6.22	out	1847
5.370.6.23	right	1847
5.370.6.24	scientific	1847
5.370.6.25	showbase	1847
5.370.6.26	showpoint	1847
5.370.6.27	showpos	1847
5.370.6.28	skipws	1848
5.370.6.29	trunc	1848
5.370.6.30	unitbuf	1848
5.370.6.3	uppercase	1848
5.371	std::basic_ostream<_CharT, _Traits> Class Template Reference	1848
5.371.1	Detailed Description	1854
5.371.2	Member Typedef Documentation	1854
5.371.2.1	__ctype_type	1854
5.371.2.2	__num_get_type	1854
5.371.2.3	__num_put_type	1854
5.371.2.4	char_type	1855
5.371.2.5	event_callback	1855
5.371.2.6	fmtflags	1855
5.371.2.7	int_type	1856

---

---

5.371.2.8	iostate	1857
5.371.2.9	off_type	1857
5.371.2.10	openmode	1857
5.371.2.11	lpos_type	1858
5.371.2.12	seekdir	1858
5.371.2.13	traits_type	1858
5.371.3	Member Enumeration Documentation	1859
5.371.3.1	event	1859
5.371.4	Constructor & Destructor Documentation	1859
5.371.4.1	basic_ostream	1859
5.371.4.2	~basic_ostream	1859
5.371.5	Member Function Documentation	1859
5.371.5.1	_M_getloc	1859
5.371.5.2	_M_write	1860
5.371.5.3	bad	1860
5.371.5.4	clear	1861
5.371.5.5	copyfmt	1861
5.371.5.6	eof	1861
5.371.5.7	exceptions	1862
5.371.5.8	exceptions	1862
5.371.5.9	fail	1863
5.371.5.10	fill	1863
5.371.5.11	fill	1864
5.371.5.12	flags	1864
5.371.5.13	flags	1864
5.371.5.14	flush	1865
5.371.5.15	getloc	1865
5.371.5.16	good	1865
5.371.5.17	imbue	1866
5.371.5.18	nit	1866
5.371.5.19	word	1866

---

5.371.5.20	narrow	1867
5.371.5.21	operator void *	1867
5.371.5.22	operator!	1868
5.371.5.23	operator<<	1868
5.371.5.24	operator<<	1868
5.371.5.25	operator<<	1869
5.371.5.26	operator<<	1869
5.371.5.27	operator<<	1870
5.371.5.28	operator<<	1870
5.371.5.29	operator<<	1870
5.371.5.30	operator<<	1871
5.371.5.31	operator<<	1871
5.371.5.32	operator<<	1872
5.371.5.33	operator<<	1872
5.371.5.34	operator<<	1872
5.371.5.35	operator<<	1873
5.371.5.36	operator<<	1873
5.371.5.37	operator<<	1873
5.371.5.38	operator<<	1874
5.371.5.39	operator<<	1874
5.371.5.40	precision	1875
5.371.5.41	precision	1875
5.371.5.42	put	1875
5.371.5.43	pwd	1876
5.371.5.44	rdbuf	1876
5.371.5.45	rdbuf	1877
5.371.5.46	rdstate	1877
5.371.5.47	register_callback	1878
5.371.5.48	seekp	1878
5.371.5.49	seekp	1879
5.371.5.50	setf	1879

---

5.371.5.51setf . . . . .	1879
5.371.5.52setstate . . . . .	1880
5.371.5.53sync_with_stdio . . . . .	1880
5.371.5.54tellp . . . . .	1881
5.371.5.55tie . . . . .	1881
5.371.5.56tie . . . . .	1882
5.371.5.57unsetf . . . . .	1882
5.371.5.58widen . . . . .	1882
5.371.5.59width . . . . .	1883
5.371.5.60width . . . . .	1883
5.371.5.61write . . . . .	1883
5.371.5.62xalloc . . . . .	1884
5.371.6 Member Data Documentation . . . . .	1884
5.371.6.1 adjustfield . . . . .	1884
5.371.6.2 app . . . . .	1885
5.371.6.3 ate . . . . .	1885
5.371.6.4 badbit . . . . .	1885
5.371.6.5 basefield . . . . .	1885
5.371.6.6 beg . . . . .	1885
5.371.6.7 binary . . . . .	1886
5.371.6.8 boolalpha . . . . .	1886
5.371.6.9 cur . . . . .	1886
5.371.6.10dec . . . . .	1886
5.371.6.11lend . . . . .	1886
5.371.6.12eofbit . . . . .	1887
5.371.6.13failbit . . . . .	1887
5.371.6.14fixed . . . . .	1887
5.371.6.15floatfield . . . . .	1887
5.371.6.16goodbit . . . . .	1888
5.371.6.17hex . . . . .	1888
5.371.6.18n . . . . .	1888

---

5.371.6.19	internal . . . . .	1888
5.371.6.20	left . . . . .	1889
5.371.6.21	loct . . . . .	1889
5.371.6.22	out . . . . .	1889
5.371.6.23	right . . . . .	1889
5.371.6.24	scientific . . . . .	1889
5.371.6.25	showbase . . . . .	1889
5.371.6.26	showpoint . . . . .	1890
5.371.6.27	showpos . . . . .	1890
5.371.6.28	skipws . . . . .	1890
5.371.6.29	trunc . . . . .	1890
5.371.6.30	unitbuf . . . . .	1890
5.371.6.31	uppercase . . . . .	1890
5.372	std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . . . .	1891
5.372.1	Detailed Description . . . . .	1891
5.372.2	Constructor & Destructor Documentation . . . . .	1891
5.372.2.1	sentry . . . . .	1891
5.372.2.2	~sentry . . . . .	1892
5.372.3	Member Function Documentation . . . . .	1892
5.372.3.1	operator bool . . . . .	1892
5.373	std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	1892
5.373.1	Detailed Description . . . . .	1898
5.373.2	Member Typedef Documentation . . . . .	1899
5.373.2.1	__ctype_type . . . . .	1899
5.373.2.2	__num_get_type . . . . .	1899
5.373.2.3	__num_put_type . . . . .	1899
5.373.2.4	char_type . . . . .	1899
5.373.2.5	event_callback . . . . .	1900
5.373.2.6	fmtflags . . . . .	1900
5.373.2.7	int_type . . . . .	1901

---

---

5.373.2.8	iostate	1901
5.373.2.9	off_type	1901
5.373.2.10	openmode	1902
5.373.2.11	lpos_type	1902
5.373.2.12	seekdir	1902
5.373.2.13	traits_type	1903
5.373.3	Member Enumeration Documentation	1903
5.373.3.1	event	1903
5.373.4	Constructor & Destructor Documentation	1903
5.373.4.1	basic_ostringstream	1903
5.373.4.2	basic_ostringstream	1904
5.373.4.3	~basic_ostringstream	1904
5.373.5	Member Function Documentation	1904
5.373.5.1	_M_getloc	1904
5.373.5.2	_M_write	1905
5.373.5.3	bad	1905
5.373.5.4	clear	1905
5.373.5.5	copyfmt	1906
5.373.5.6	eof	1906
5.373.5.7	exceptions	1907
5.373.5.8	exceptions	1907
5.373.5.9	fail	1908
5.373.5.10	fill	1908
5.373.5.11	fill	1908
5.373.5.12	flags	1909
5.373.5.13	flags	1909
5.373.5.14	flush	1910
5.373.5.15	getloc	1910
5.373.5.16	good	1910
5.373.5.17	imbue	1911
5.373.5.18	nit	1911

---

5.373.5.19word . . . . .	1911
5.373.5.20narrow . . . . .	1912
5.373.5.21operator void * . . . . .	1912
5.373.5.22operator! . . . . .	1913
5.373.5.23operator<< . . . . .	1913
5.373.5.24operator<< . . . . .	1913
5.373.5.25operator<< . . . . .	1914
5.373.5.26operator<< . . . . .	1914
5.373.5.27operator<< . . . . .	1914
5.373.5.28operator<< . . . . .	1915
5.373.5.29operator<< . . . . .	1915
5.373.5.30operator<< . . . . .	1915
5.373.5.31operator<< . . . . .	1916
5.373.5.32operator<< . . . . .	1916
5.373.5.33operator<< . . . . .	1917
5.373.5.34operator<< . . . . .	1917
5.373.5.35operator<< . . . . .	1917
5.373.5.36operator<< . . . . .	1918
5.373.5.37operator<< . . . . .	1918
5.373.5.38operator<< . . . . .	1919
5.373.5.39operator<< . . . . .	1919
5.373.5.40precision . . . . .	1919
5.373.5.41precision . . . . .	1920
5.373.5.42put . . . . .	1920
5.373.5.43pword . . . . .	1921
5.373.5.44rdbuf . . . . .	1921
5.373.5.45rdbuf . . . . .	1922
5.373.5.46rdstate . . . . .	1922
5.373.5.47register_callback . . . . .	1922
5.373.5.48seekp . . . . .	1923
5.373.5.49seekp . . . . .	1923

---

5.373.5.50	setf	1924
5.373.5.51	lsetf	1924
5.373.5.52	setstate	1924
5.373.5.53	str	1925
5.373.5.54	str	1925
5.373.5.55	sync_with_stdio	1926
5.373.5.56	tellp	1926
5.373.5.57	tie	1926
5.373.5.58	tie	1927
5.373.5.59	unsetf	1927
5.373.5.60	widen	1927
5.373.5.61	lwidth	1928
5.373.5.62	width	1928
5.373.5.63	write	1929
5.373.5.64	xalloc	1929
5.373.6	Member Data Documentation	1930
5.373.6.1	adjustfield	1930
5.373.6.2	app	1930
5.373.6.3	ate	1930
5.373.6.4	badbit	1930
5.373.6.5	basefield	1931
5.373.6.6	beg	1931
5.373.6.7	binary	1931
5.373.6.8	boolalpha	1931
5.373.6.9	cur	1931
5.373.6.10	dec	1931
5.373.6.11	lend	1932
5.373.6.12	eofbit	1932
5.373.6.13	failbit	1932
5.373.6.14	fixed	1932
5.373.6.15	floatfield	1933

5.373.6.16	goodbit . . . . .	1933
5.373.6.17	hex . . . . .	1933
5.373.6.18	n . . . . .	1933
5.373.6.19	internal . . . . .	1934
5.373.6.20	left . . . . .	1934
5.373.6.21	loct . . . . .	1934
5.373.6.22	out . . . . .	1934
5.373.6.23	right . . . . .	1934
5.373.6.24	scientific . . . . .	1934
5.373.6.25	showbase . . . . .	1935
5.373.6.26	showpoint . . . . .	1935
5.373.6.27	showpos . . . . .	1935
5.373.6.28	skipws . . . . .	1935
5.373.6.29	trunc . . . . .	1935
5.373.6.30	unitbuf . . . . .	1935
5.373.6.3	uppercase . . . . .	1935
5.374	std::basic_streambuf<_CharT, _Traits> Class Template Reference . . . . .	1936
5.374.1	Detailed Description . . . . .	1938
5.374.2	Member Typedef Documentation . . . . .	1940
5.374.2.1	__streambuf_type . . . . .	1940
5.374.2.2	char_type . . . . .	1940
5.374.2.3	int_type . . . . .	1940
5.374.2.4	off_type . . . . .	1941
5.374.2.5	pos_type . . . . .	1941
5.374.2.6	traits_type . . . . .	1941
5.374.3	Constructor & Destructor Documentation . . . . .	1942
5.374.3.1	~basic_streambuf . . . . .	1942
5.374.3.2	basic_streambuf . . . . .	1942
5.374.4	Member Function Documentation . . . . .	1942
5.374.4.1	eback . . . . .	1942
5.374.4.2	egptr . . . . .	1943

---

5.374.4.3	eptr	1943
5.374.4.4	gbump	1943
5.374.4.5	getloc	1944
5.374.4.6	gptr	1944
5.374.4.7	imbue	1944
5.374.4.8	in_avail	1945
5.374.4.9	overflow	1945
5.374.4.10	backfail	1946
5.374.4.11	lbase	1947
5.374.4.12	pbump	1947
5.374.4.13	pptr	1947
5.374.4.14	pubimbue	1948
5.374.4.15	pubseekoff	1948
5.374.4.16	pubseekpos	1949
5.374.4.17	pubsetbuf	1949
5.374.4.18	pubsync	1949
5.374.4.19	sbumpc	1950
5.374.4.20	seekoff	1950
5.374.4.21	seekpos	1950
5.374.4.22	setbuf	1951
5.374.4.23	setg	1951
5.374.4.24	setp	1951
5.374.4.25	sgetc	1952
5.374.4.26	sgetn	1952
5.374.4.27	showmanyc	1953
5.374.4.28	nextc	1953
5.374.4.29	sputbackc	1954
5.374.4.30	sputc	1954
5.374.4.31	lspun	1955
5.374.4.32	stoss	1955
5.374.4.33	sungetc	1955

---

5.374.4.34	<code>sync</code>	1956
5.374.4.35	<code>uflow</code>	1956
5.374.4.36	<code>underflow</code>	1956
5.374.4.37	<code>xsggetn</code>	1957
5.374.4.38	<code>xspn</code>	1958
5.374.5	Member Data Documentation	1958
5.374.5.1	<code>_M_buf_locale</code>	1958
5.374.5.2	<code>_M_in_beg</code>	1959
5.374.5.3	<code>_M_in_cur</code>	1959
5.374.5.4	<code>_M_in_end</code>	1959
5.374.5.5	<code>_M_out_beg</code>	1960
5.374.5.6	<code>_M_out_cur</code>	1960
5.374.5.7	<code>_M_out_end</code>	1960
5.375	<code>std::basic_string&lt;_CharT, _Traits, _Alloc&gt;</code> Class Template Reference	1961
5.375.1	Detailed Description	1965
5.375.2	Constructor & Destructor Documentation	1966
5.375.2.1	<code>basic_string</code>	1966
5.375.2.2	<code>basic_string</code>	1967
5.375.2.3	<code>basic_string</code>	1967
5.375.2.4	<code>basic_string</code>	1967
5.375.2.5	<code>basic_string</code>	1967
5.375.2.6	<code>basic_string</code>	1968
5.375.2.7	<code>basic_string</code>	1968
5.375.2.8	<code>basic_string</code>	1968
5.375.2.9	<code>basic_string</code>	1969
5.375.2.10	<code>basic_string</code>	1969
5.375.2.11	<code>basic_string</code>	1969
5.375.2.12	<code>~basic_string</code>	1970
5.375.3	Member Function Documentation	1970
5.375.3.1	<code>append</code>	1970
5.375.3.2	<code>append</code>	1971

---

5.375.3.3 append	1971
5.375.3.4 append	1971
5.375.3.5 append	1972
5.375.3.6 append	1972
5.375.3.7 append	1973
5.375.3.8 assign	1973
5.375.3.9 assign	1974
5.375.3.10 assign	1974
5.375.3.11 assign	1975
5.375.3.12 assign	1975
5.375.3.13 assign	1975
5.375.3.14 assign	1976
5.375.3.15 assign	1976
5.375.3.16 at	1977
5.375.3.17 at	1977
5.375.3.18 begin	1978
5.375.3.19 begin	1978
5.375.3.20 c_str	1978
5.375.3.21 capacity	1978
5.375.3.22 cbegin	1979
5.375.3.23 cend	1979
5.375.3.24 clear	1979
5.375.3.25 compare	1979
5.375.3.26 compare	1980
5.375.3.27 compare	1981
5.375.3.28 compare	1981
5.375.3.29 compare	1982
5.375.3.30 compare	1982
5.375.3.31 copy	1983
5.375.3.32 cbegin	1984
5.375.3.33 cend	1984

---

5.375.3.34	data	1984
5.375.3.35	empty	1984
5.375.3.36	end	1985
5.375.3.37	end	1985
5.375.3.38	erase	1985
5.375.3.39	erase	1986
5.375.3.40	erase	1986
5.375.3.41	find	1986
5.375.3.42	find	1987
5.375.3.43	find	1987
5.375.3.44	find	1988
5.375.3.45	find_first_not_of	1988
5.375.3.46	find_first_not_of	1989
5.375.3.47	find_first_not_of	1989
5.375.3.48	find_first_not_of	1990
5.375.3.49	find_first_of	1990
5.375.3.50	find_first_of	1991
5.375.3.51	find_first_of	1991
5.375.3.52	find_first_of	1992
5.375.3.53	find_last_not_of	1992
5.375.3.54	find_last_not_of	1993
5.375.3.55	find_last_not_of	1993
5.375.3.56	find_last_not_of	1994
5.375.3.57	find_last_of	1994
5.375.3.58	find_last_of	1995
5.375.3.59	find_last_of	1995
5.375.3.60	find_last_of	1996
5.375.3.61	get_allocator	1996
5.375.3.62	insert	1996
5.375.3.63	insert	1997
5.375.3.64	insert	1997

---

5.375.3.65	insert	1998
5.375.3.66	insert	1998
5.375.3.67	insert	1999
5.375.3.68	insert	1999
5.375.3.69	insert	2000
5.375.3.70	insert	2001
5.375.3.71	length	2001
5.375.3.72	max_size	2001
5.375.3.73	operator+=	2002
5.375.3.74	operator+=	2002
5.375.3.75	operator+=	2002
5.375.3.76	operator+=	2003
5.375.3.77	operator=	2003
5.375.3.78	operator=	2003
5.375.3.79	operator=	2004
5.375.3.80	operator=	2004
5.375.3.81	operator=	2004
5.375.3.82	operator[]	2004
5.375.3.83	operator[]	2005
5.375.3.84	push_back	2005
5.375.3.85	begin	2006
5.375.3.86	begin	2006
5.375.3.87	end	2006
5.375.3.88	end	2006
5.375.3.89	replace	2006
5.375.3.90	replace	2007
5.375.3.91	replace	2008
5.375.3.92	replace	2008
5.375.3.93	replace	2009
5.375.3.94	replace	2010
5.375.3.95	replace	2010

---

5.375.3.96	replace	2011
5.375.3.97	replace	2012
5.375.3.98	replace	2012
5.375.3.99	replace	2013
5.375.3.100	reserve	2013
5.375.3.101	size	2014
5.375.3.102	size	2014
5.375.3.103	find	2015
5.375.3.104	find	2015
5.375.3.105	find	2016
5.375.3.106	find	2016
5.375.3.107	shrink_to_fit	2017
5.375.3.108	size	2017
5.375.3.109	substr	2017
5.375.3.110	swap	2018
5.375.4	Member Data Documentation	2018
5.375.4.1	npos	2018
5.376	std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference	2019
5.376.1	Detailed Description	2022
5.376.2	Member Typedef Documentation	2022
5.376.2.1	__streambuf_type	2022
5.376.2.2	char_type	2023
5.376.2.3	int_type	2023
5.376.2.4	off_type	2023
5.376.2.5	pos_type	2023
5.376.2.6	traits_type	2024
5.376.3	Constructor & Destructor Documentation	2024
5.376.3.1	basic_stringbuf	2024
5.376.3.2	basic_stringbuf	2024
5.376.4	Member Function Documentation	2025

---

---

5.376.4.1 eback . . . . .	2025
5.376.4.2 egptr . . . . .	2025
5.376.4.3 epptr . . . . .	2025
5.376.4.4 gbump . . . . .	2026
5.376.4.5 getloc . . . . .	2026
5.376.4.6 gptr . . . . .	2026
5.376.4.7 imbue . . . . .	2027
5.376.4.8 in_avail . . . . .	2027
5.376.4.9 overflow . . . . .	2028
5.376.4.10pbackfail . . . . .	2028
5.376.4.11pbase . . . . .	2029
5.376.4.12pbump . . . . .	2029
5.376.4.13pptr . . . . .	2030
5.376.4.14pubimbue . . . . .	2030
5.376.4.15pubseekoff . . . . .	2031
5.376.4.16pubseekpos . . . . .	2031
5.376.4.17pubsetbuf . . . . .	2031
5.376.4.18pubsync . . . . .	2032
5.376.4.19sbumpc . . . . .	2032
5.376.4.20seekoff . . . . .	2032
5.376.4.21seekpos . . . . .	2033
5.376.4.22setbuf . . . . .	2033
5.376.4.23setbuf . . . . .	2034
5.376.4.24setg . . . . .	2034
5.376.4.25setp . . . . .	2034
5.376.4.26getc . . . . .	2035
5.376.4.27sgetn . . . . .	2035
5.376.4.28showmanyc . . . . .	2035
5.376.4.29nextc . . . . .	2036
5.376.4.30sputbackc . . . . .	2036
5.376.4.31sputc . . . . .	2037

5.376.4.32	sputn . . . . .	2037
5.376.4.33	stoss . . . . .	2038
5.376.4.34	str . . . . .	2038
5.376.4.35	str . . . . .	2038
5.376.4.36	sungetc . . . . .	2039
5.376.4.37	sync . . . . .	2039
5.376.4.38	uflow . . . . .	2039
5.376.4.39	underflow . . . . .	2040
5.376.4.40	xsgetn . . . . .	2041
5.376.4.41	xsputn . . . . .	2041
5.376.5	Member Data Documentation . . . . .	2042
5.376.5.1	_M_buf_locale . . . . .	2042
5.376.5.2	_M_in_beg . . . . .	2042
5.376.5.3	_M_in_cur . . . . .	2042
5.376.5.4	_M_in_end . . . . .	2043
5.376.5.5	_M_mode . . . . .	2043
5.376.5.6	_M_out_beg . . . . .	2043
5.376.5.7	_M_out_cur . . . . .	2044
5.376.5.8	_M_out_end . . . . .	2044
5.377	std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	2045
5.377.1	Detailed Description . . . . .	2052
5.377.2	Member Typedef Documentation . . . . .	2053
5.377.2.1	__ctype_type . . . . .	2053
5.377.2.2	__ctype_type . . . . .	2053
5.377.2.3	__num_get_type . . . . .	2053
5.377.2.4	__num_put_type . . . . .	2053
5.377.2.5	__num_put_type . . . . .	2054
5.377.2.6	char_type . . . . .	2054
5.377.2.7	char_type . . . . .	2054
5.377.2.8	event_callback . . . . .	2054

---

5.377.2.9	fmtflags	2055
5.377.2.10	int_type	2056
5.377.2.11	int_type	2056
5.377.2.12	istate	2056
5.377.2.13	off_type	2057
5.377.2.14	off_type	2057
5.377.2.15	openmode	2057
5.377.2.16	pos_type	2058
5.377.2.17	pos_type	2058
5.377.2.18	seekdir	2058
5.377.2.19	traits_type	2058
5.377.2.20	traits_type	2059
5.377.3	Member Enumeration Documentation	2059
5.377.3.1	event	2059
5.377.4	Constructor & Destructor Documentation	2059
5.377.4.1	basic_stringstream	2059
5.377.4.2	basic_stringstream	2060
5.377.4.3	~basic_stringstream	2060
5.377.5	Member Function Documentation	2060
5.377.5.1	_M_getloc	2060
5.377.5.2	_M_write	2061
5.377.5.3	bad	2061
5.377.5.4	clear	2062
5.377.5.5	copyfmt	2062
5.377.5.6	eof	2062
5.377.5.7	exceptions	2063
5.377.5.8	exceptions	2063
5.377.5.9	fail	2064
5.377.5.10	fill	2064
5.377.5.11	fill	2065
5.377.5.12	flags	2065

---

5.377.5.13	flags	2065
5.377.5.14	flush	2066
5.377.5.15	gcount	2066
5.377.5.16	get	2066
5.377.5.17	get	2067
5.377.5.18	get	2067
5.377.5.19	get	2068
5.377.5.20	get	2069
5.377.5.21	get	2069
5.377.5.22	getline	2070
5.377.5.23	getline	2071
5.377.5.24	getloc	2071
5.377.5.25	good	2072
5.377.5.26	ignore	2072
5.377.5.27	ignore	2072
5.377.5.28	ignore	2073
5.377.5.29	mbue	2073
5.377.5.30	nit	2074
5.377.5.31	word	2074
5.377.5.32	narrow	2075
5.377.5.33	operator void *	2075
5.377.5.34	operator!	2075
5.377.5.35	operator<<	2076
5.377.5.36	operator<<	2076
5.377.5.37	operator<<	2076
5.377.5.38	operator<<	2077
5.377.5.39	operator<<	2077
5.377.5.40	operator<<	2077
5.377.5.41	operator<<	2078
5.377.5.42	operator<<	2078
5.377.5.43	operator<<	2078

---

5.377.5.44operator<<	2079
5.377.5.45operator<<	2079
5.377.5.46operator<<	2080
5.377.5.47operator<<	2080
5.377.5.48operator<<	2080
5.377.5.49operator<<	2081
5.377.5.50operator<<	2082
5.377.5.51operator<<	2082
5.377.5.52operator>>	2082
5.377.5.53operator>>	2083
5.377.5.54operator>>	2083
5.377.5.55operator>>	2084
5.377.5.56operator>>	2084
5.377.5.57operator>>	2084
5.377.5.58operator>>	2085
5.377.5.59operator>>	2085
5.377.5.60operator>>	2085
5.377.5.61operator>>	2086
5.377.5.62operator>>	2086
5.377.5.63operator>>	2087
5.377.5.64operator>>	2087
5.377.5.65operator>>	2088
5.377.5.66operator>>	2088
5.377.5.67operator>>	2088
5.377.5.68operator>>	2088
5.377.5.69peek	2089
5.377.5.70precision	2089
5.377.5.71precision	2090
5.377.5.72put	2090
5.377.5.73putback	2090
5.377.5.74pword	2091

---

5.377.5.75	rdbuf	2092
5.377.5.76	rdbuf	2092
5.377.5.77	rdstate	2093
5.377.5.78	read	2093
5.377.5.79	readsome	2094
5.377.5.80	register_callback	2094
5.377.5.81	seekg	2095
5.377.5.82	seekg	2095
5.377.5.83	seekp	2096
5.377.5.84	seekp	2096
5.377.5.85	setf	2097
5.377.5.86	setf	2097
5.377.5.87	setstate	2098
5.377.5.88	str	2098
5.377.5.89	str	2098
5.377.5.90	sync	2099
5.377.5.91	sync_with_stdio	2099
5.377.5.92	tellg	2100
5.377.5.93	tellp	2100
5.377.5.94	tie	2100
5.377.5.95	tie	2101
5.377.5.96	unget	2101
5.377.5.97	unsetf	2102
5.377.5.98	widen	2102
5.377.5.99	width	2103
5.377.5.100	width	2103
5.377.5.101	write	2103
5.377.5.102	zalloc	2104
5.377.6	Member Data Documentation	2104
5.377.6.1	_M_gcount	2104
5.377.6.2	adjustfield	2104

---

5.377.6.3	app	2105
5.377.6.4	ate	2105
5.377.6.5	badbit	2105
5.377.6.6	basefield	2105
5.377.6.7	beg	2106
5.377.6.8	binary	2106
5.377.6.9	boolalpha	2106
5.377.6.10	cur	2106
5.377.6.11	dec	2106
5.377.6.12	end	2106
5.377.6.13	eofbit	2107
5.377.6.14	failbit	2107
5.377.6.15	fixed	2107
5.377.6.16	floatfield	2107
5.377.6.17	goodbit	2108
5.377.6.18	hex	2108
5.377.6.19	in	2108
5.377.6.20	internal	2108
5.377.6.21	left	2109
5.377.6.22	oct	2109
5.377.6.23	out	2109
5.377.6.24	right	2109
5.377.6.25	scientific	2109
5.377.6.26	showbase	2109
5.377.6.27	showpoint	2110
5.377.6.28	showpos	2110
5.377.6.29	skipws	2110
5.377.6.30	trunc	2110
5.377.6.31	unitbuf	2110
5.377.6.32	uppercase	2110
5.378	std::bernoulli_distribution Class Reference	2111

---

5.378.1 Detailed Description . . . . .	2111
5.378.2 Member Typedef Documentation . . . . .	2111
5.378.2.1 result_type . . . . .	2111
5.378.3 Constructor & Destructor Documentation . . . . .	2112
5.378.3.1 bernoulli_distribution . . . . .	2112
5.378.4 Member Function Documentation . . . . .	2112
5.378.4.1 max . . . . .	2112
5.378.4.2 min . . . . .	2112
5.378.4.3 operator() . . . . .	2112
5.378.4.4 p . . . . .	2112
5.378.4.5 param . . . . .	2113
5.378.4.6 param . . . . .	2113
5.378.4.7 reset . . . . .	2113
5.379std::bernoulli_distribution::param_type Struct Reference . . . . .	2113
5.379.1 Detailed Description . . . . .	2114
5.380std::bidirectional_iterator_tag Struct Reference . . . . .	2114
5.380.1 Detailed Description . . . . .	2115
5.381std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference . . . . .	2115
5.381.1 Detailed Description . . . . .	2116
5.381.2 Member Typedef Documentation . . . . .	2116
5.381.2.1 first_argument_type . . . . .	2116
5.381.2.2 result_type . . . . .	2116
5.381.2.3 second_argument_type . . . . .	2116
5.382std::binary_negate< _Predicate > Class Template Reference . . . . .	2117
5.382.1 Detailed Description . . . . .	2117
5.382.2 Member Typedef Documentation . . . . .	2117
5.382.2.1 first_argument_type . . . . .	2117
5.382.2.2 result_type . . . . .	2118
5.382.2.3 second_argument_type . . . . .	2118
5.383std::binder1st< _Operation > Class Template Reference . . . . .	2118

---

5.383.1 Detailed Description . . . . .	2119
5.383.2 Member Typedef Documentation . . . . .	2119
5.383.2.1 argument_type . . . . .	2119
5.383.2.2 result_type . . . . .	2119
5.384std::binder2nd< _Operation > Class Template Reference . . . . .	2120
5.384.1 Detailed Description . . . . .	2121
5.384.2 Member Typedef Documentation . . . . .	2121
5.384.2.1 argument_type . . . . .	2121
5.384.2.2 result_type . . . . .	2121
5.385std::binomial_distribution< _IntType > Class Template Reference . . . . .	2121
5.385.1 Detailed Description . . . . .	2122
5.385.2 Member Typedef Documentation . . . . .	2123
5.385.2.1 result_type . . . . .	2123
5.385.3 Member Function Documentation . . . . .	2123
5.385.3.1 max . . . . .	2123
5.385.3.2 min . . . . .	2123
5.385.3.3 operator() . . . . .	2123
5.385.3.4 operator() . . . . .	2124
5.385.3.5 p . . . . .	2124
5.385.3.6 param . . . . .	2124
5.385.3.7 param . . . . .	2124
5.385.3.8 reset . . . . .	2125
5.385.3.9 t . . . . .	2125
5.385.4 Friends And Related Function Documentation . . . . .	2125
5.385.4.1 operator<< . . . . .	2125
5.385.4.2 operator== . . . . .	2125
5.385.4.3 operator>> . . . . .	2126
5.386std::binomial_distribution< _IntType >::param_type Struct Reference . . . . .	2126
5.386.1 Detailed Description . . . . .	2127
5.387std::bitset< _Nb > Class Template Reference . . . . .	2127
5.387.1 Detailed Description . . . . .	2130

---

5.387.2 Constructor & Destructor Documentation . . . . .	2132
5.387.2.1 bitset . . . . .	2132
5.387.2.2 bitset . . . . .	2132
5.387.2.3 bitset . . . . .	2132
5.387.2.4 bitset . . . . .	2132
5.387.2.5 bitset . . . . .	2133
5.387.3 Member Function Documentation . . . . .	2133
5.387.3.1 all . . . . .	2133
5.387.3.2 any . . . . .	2134
5.387.3.3 count . . . . .	2134
5.387.3.4 flip . . . . .	2134
5.387.3.5 flip . . . . .	2134
5.387.3.6 none . . . . .	2134
5.387.3.7 operator!= . . . . .	2135
5.387.3.8 operator&= . . . . .	2135
5.387.3.9 operator<< . . . . .	2135
5.387.3.10operator<<= . . . . .	2135
5.387.3.11operator== . . . . .	2136
5.387.3.12operator>> . . . . .	2136
5.387.3.13operator>>= . . . . .	2136
5.387.3.14operator[] . . . . .	2136
5.387.3.15operator[] . . . . .	2137
5.387.3.16operator^= . . . . .	2137
5.387.3.17operator = . . . . .	2138
5.387.3.18operator~ . . . . .	2138
5.387.3.19reset . . . . .	2138
5.387.3.20reset . . . . .	2138
5.387.3.21set . . . . .	2139
5.387.3.22set . . . . .	2139
5.387.3.23size . . . . .	2139
5.387.3.24test . . . . .	2139

---

5.387.3.25to_string . . . . .	2140
5.387.3.26to_ulong . . . . .	2140
5.388std::bitset< _Nb >::reference Class Reference . . . . .	2140
5.388.1 Detailed Description . . . . .	2141
5.389std::cauchy_distribution< _RealType > Class Template Reference . . . . .	2141
5.389.1 Detailed Description . . . . .	2142
5.389.2 Member Typedef Documentation . . . . .	2142
5.389.2.1 result_type . . . . .	2142
5.389.3 Member Function Documentation . . . . .	2142
5.389.3.1 max . . . . .	2142
5.389.3.2 min . . . . .	2143
5.389.3.3 operator() . . . . .	2143
5.389.3.4 param . . . . .	2143
5.389.3.5 param . . . . .	2143
5.389.3.6 reset . . . . .	2144
5.390std::cauchy_distribution< _RealType >::param_type Struct Reference . . . . .	2144
5.390.1 Detailed Description . . . . .	2144
5.391std::char_traits< _CharT > Struct Template Reference . . . . .	2145
5.391.1 Detailed Description . . . . .	2146
5.392std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference . . . . .	2146
5.392.1 Detailed Description . . . . .	2147
5.393std::char_traits< char > Struct Template Reference . . . . .	2147
5.393.1 Detailed Description . . . . .	2148
5.394std::char_traits< wchar_t > Struct Template Reference . . . . .	2148
5.394.1 Detailed Description . . . . .	2149
5.395std::chi_squared_distribution< _RealType > Class Template Reference . . . . .	2149
5.395.1 Detailed Description . . . . .	2150
5.395.2 Member Typedef Documentation . . . . .	2151
5.395.2.1 result_type . . . . .	2151
5.395.3 Member Function Documentation . . . . .	2151

5.395.3.1	max	2151
5.395.3.2	min	2151
5.395.3.3	operator()	2151
5.395.3.4	param	2151
5.395.3.5	param	2152
5.395.3.6	reset	2152
5.395.4	Friends And Related Function Documentation	2152
5.395.4.1	operator<<	2152
5.395.4.2	operator==	2153
5.395.4.3	operator>>	2153
5.396	std::chi_squared_distribution<_RealType>::param_type Struct Reference	2153
5.396.1	Detailed Description	2154
5.397	std::chrono::duration<_Rep, _Period> Struct Template Reference	2154
5.397.1	Detailed Description	2155
5.398	std::chrono::duration_values<_Rep> Struct Template Reference	2155
5.398.1	Detailed Description	2156
5.399	std::chrono::system_clock Struct Reference	2156
5.399.1	Detailed Description	2157
5.400	std::chrono::time_point<_Clock, _Duration> Struct Template Reference	2157
5.400.1	Detailed Description	2157
5.401	std::chrono::treat_as_floating_point<_Rep> Struct Template Reference	2158
5.401.1	Detailed Description	2158
5.402	std::codecvt<_InternT, _ExternT, _StateT> Class Template Reference	2158
5.402.1	Detailed Description	2161
5.402.2	Member Function Documentation	2161
5.402.2.1	do_out	2161
5.402.2.2	in	2161
5.402.2.3	out	2162
5.402.2.4	unshift	2163

---

5.403std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference . . . . .	2164
5.403.1 Detailed Description . . . . .	2166
5.403.2 Member Function Documentation . . . . .	2167
5.403.2.1 do_out . . . . .	2167
5.403.2.2 in . . . . .	2167
5.403.2.3 out . . . . .	2168
5.403.2.4 unshift . . . . .	2169
5.404std::codecvt< char, char, mbstate_t > Class Template Reference . . .	2170
5.404.1 Detailed Description . . . . .	2172
5.404.2 Member Function Documentation . . . . .	2172
5.404.2.1 do_out . . . . .	2172
5.404.2.2 in . . . . .	2172
5.404.2.3 out . . . . .	2173
5.404.2.4 unshift . . . . .	2174
5.405std::codecvt< wchar_t, char, mbstate_t > Class Template Reference .	2175
5.405.1 Detailed Description . . . . .	2177
5.405.2 Member Function Documentation . . . . .	2177
5.405.2.1 do_out . . . . .	2177
5.405.2.2 in . . . . .	2178
5.405.2.3 out . . . . .	2179
5.405.2.4 unshift . . . . .	2180
5.406std::codecvt_base Class Reference . . . . .	2180
5.406.1 Detailed Description . . . . .	2181
5.407std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference . . . . .	2181
5.407.1 Detailed Description . . . . .	2184
5.407.2 Member Function Documentation . . . . .	2184
5.407.2.1 do_out . . . . .	2184
5.407.2.2 in . . . . .	2184
5.407.2.3 out . . . . .	2185
5.407.2.4 unshift . . . . .	2186

---

5.408std::collate<_CharT > Class Template Reference . . . . .	2187
5.408.1 Detailed Description . . . . .	2189
5.408.2 Member Typedef Documentation . . . . .	2190
5.408.2.1 char_type . . . . .	2190
5.408.2.2 string_type . . . . .	2190
5.408.3 Constructor & Destructor Documentation . . . . .	2190
5.408.3.1 collate . . . . .	2190
5.408.3.2 collate . . . . .	2190
5.408.3.3 ~collate . . . . .	2191
5.408.4 Member Function Documentation . . . . .	2191
5.408.4.1 compare . . . . .	2191
5.408.4.2 do_compare . . . . .	2191
5.408.4.3 do_hash . . . . .	2192
5.408.4.4 do_transform . . . . .	2192
5.408.4.5 hash . . . . .	2193
5.408.4.6 transform . . . . .	2193
5.408.5 Member Data Documentation . . . . .	2194
5.408.5.1 id . . . . .	2194
5.409std::collate_byname<_CharT > Class Template Reference . . . . .	2194
5.409.1 Detailed Description . . . . .	2196
5.409.2 Member Typedef Documentation . . . . .	2197
5.409.2.1 char_type . . . . .	2197
5.409.2.2 string_type . . . . .	2197
5.409.3 Member Function Documentation . . . . .	2197
5.409.3.1 compare . . . . .	2197
5.409.3.2 do_compare . . . . .	2198
5.409.3.3 do_hash . . . . .	2198
5.409.3.4 do_transform . . . . .	2199
5.409.3.5 hash . . . . .	2199
5.409.3.6 transform . . . . .	2200
5.409.4 Member Data Documentation . . . . .	2200

---

---

5.409.4.1 id . . . . .	2200
5.410std::complex< _Tp > Struct Template Reference . . . . .	2200
5.410.1 Detailed Description . . . . .	2201
5.410.2 Member Typedef Documentation . . . . .	2201
5.410.2.1 value_type . . . . .	2201
5.410.3 Constructor & Destructor Documentation . . . . .	2202
5.410.3.1 complex . . . . .	2202
5.410.3.2 complex . . . . .	2202
5.410.4 Member Function Documentation . . . . .	2202
5.410.4.1 operator+= . . . . .	2202
5.410.4.2 operator-= . . . . .	2202
5.411std::condition_variable Class Reference . . . . .	2202
5.411.1 Detailed Description . . . . .	2203
5.412std::condition_variable_any Class Reference . . . . .	2203
5.412.1 Detailed Description . . . . .	2204
5.413std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference . . . . .	2204
5.413.1 Detailed Description . . . . .	2205
5.414std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2205
5.414.1 Detailed Description . . . . .	2206
5.414.2 Member Typedef Documentation . . . . .	2206
5.414.2.1 first_argument_type . . . . .	2206
5.414.2.2 result_type . . . . .	2206
5.414.2.3 second_argument_type . . . . .	2206
5.415std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2207
5.415.1 Detailed Description . . . . .	2208
5.415.2 Member Typedef Documentation . . . . .	2208
5.415.2.1 first_argument_type . . . . .	2208
5.415.2.2 result_type . . . . .	2208
5.415.2.3 second_argument_type . . . . .	2208
5.416std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . . .	2208

---

---

5.416.1 Detailed Description . . . . .	2209
5.416.2 Member Typedef Documentation . . . . .	2210
5.416.2.1 argument_type . . . . .	2210
5.416.2.2 result_type . . . . .	2210
5.417std::const_mem_fun_t<_Ret, _Tp > Class Template Reference . . . . .	2210
5.417.1 Detailed Description . . . . .	2211
5.417.2 Member Typedef Documentation . . . . .	2212
5.417.2.1 argument_type . . . . .	2212
5.417.2.2 result_type . . . . .	2212
5.418std::ctype<_CharT > Class Template Reference . . . . .	2212
5.418.1 Detailed Description . . . . .	2215
5.418.2 Member Typedef Documentation . . . . .	2215
5.418.2.1 char_type . . . . .	2215
5.418.3 Member Function Documentation . . . . .	2216
5.418.3.1 do_is . . . . .	2216
5.418.3.2 do_is . . . . .	2216
5.418.3.3 do_narrow . . . . .	2217
5.418.3.4 do_narrow . . . . .	2217
5.418.3.5 do_scan_is . . . . .	2218
5.418.3.6 do_scan_not . . . . .	2218
5.418.3.7 do_tolower . . . . .	2219
5.418.3.8 do_tolower . . . . .	2219
5.418.3.9 do_toupper . . . . .	2220
5.418.3.10do_toupper . . . . .	2220
5.418.3.11do_widen . . . . .	2221
5.418.3.12do_widen . . . . .	2221
5.418.3.13s . . . . .	2222
5.418.3.14s . . . . .	2222
5.418.3.15narrow . . . . .	2223
5.418.3.16narrow . . . . .	2223
5.418.3.17scan_is . . . . .	2224

---

---

5.418.3.18	scan_not	2224
5.418.3.19	tolower	2225
5.418.3.20	tolower	2225
5.418.3.21	toupper	2225
5.418.3.22	toupper	2226
5.418.3.23	widen	2226
5.418.3.24	widen	2227
5.418.4	Member Data Documentation	2227
5.418.4.1	id	2227
5.419	std::ctype< char > Class Template Reference	2228
5.419.1	Detailed Description	2230
5.419.2	Member Typedef Documentation	2231
5.419.2.1	char_type	2231
5.419.3	Constructor & Destructor Documentation	2231
5.419.3.1	ctype	2231
5.419.3.2	ctype	2231
5.419.3.3	~ctype	2231
5.419.4	Member Function Documentation	2232
5.419.4.1	classic_table	2232
5.419.4.2	do_narrow	2232
5.419.4.3	do_narrow	2232
5.419.4.4	do_tolower	2233
5.419.4.5	do_tolower	2233
5.419.4.6	do_toupper	2234
5.419.4.7	do_toupper	2234
5.419.4.8	do_widen	2235
5.419.4.9	do_widen	2235
5.419.4.10	s	2236
5.419.4.11	lis	2236
5.419.4.12	narrow	2236
5.419.4.13	narrow	2237

---

5.419.4.14	scan_is	2238
5.419.4.15	scan_not	2238
5.419.4.16	table	2238
5.419.4.17	tolower	2239
5.419.4.18	tolower	2239
5.419.4.19	toupper	2239
5.419.4.20	toupper	2240
5.419.4.21	widen	2240
5.419.4.22	widen	2241
5.419.5	Member Data Documentation	2241
5.419.5.1	id	2241
5.419.5.2	table_size	2241
5.420	std::ctype< wchar_t > Class Template Reference	2242
5.420.1	Detailed Description	2245
5.420.2	Member Typedef Documentation	2245
5.420.2.1	char_type	2245
5.420.3	Constructor & Destructor Documentation	2246
5.420.3.1	ctype	2246
5.420.3.2	ctype	2246
5.420.3.3	~ctype	2246
5.420.4	Member Function Documentation	2246
5.420.4.1	do_is	2246
5.420.4.2	do_is	2247
5.420.4.3	do_is	2247
5.420.4.4	do_is	2248
5.420.4.5	do_narrow	2248
5.420.4.6	do_narrow	2249
5.420.4.7	do_narrow	2249
5.420.4.8	do_narrow	2250
5.420.4.9	do_scan_is	2250
5.420.4.10	do_scan_is	2251

---

5.420.4.1	do_scan_not	2251
5.420.4.12	do_scan_not	2252
5.420.4.13	do_tolower	2252
5.420.4.14	do_tolower	2252
5.420.4.15	do_tolower	2253
5.420.4.16	do_tolower	2253
5.420.4.17	do_toupper	2254
5.420.4.18	do_toupper	2254
5.420.4.19	do_toupper	2255
5.420.4.20	do_toupper	2255
5.420.4.21	do_widen	2255
5.420.4.22	do_widen	2256
5.420.4.23	do_widen	2256
5.420.4.24	is	2257
5.420.4.25	is	2257
5.420.4.26	narrow	2258
5.420.4.27	narrow	2258
5.420.4.28	scan_is	2259
5.420.4.29	scan_not	2259
5.420.4.30	tolower	2260
5.420.4.31	tolower	2260
5.420.4.32	toupper	2261
5.420.4.33	toupper	2261
5.420.4.34	widen	2261
5.420.4.35	widen	2262
5.420.5	Member Data Documentation	2262
5.420.5.1	id	2262
5.421	std::ctype_base Struct Reference	2263
5.421.1	Detailed Description	2263
5.422	std::ctype_byname<_CharT> Class Template Reference	2264
5.422.1	Detailed Description	2266

---

5.422.2 Member Typedef Documentation . . . . .	2266
5.422.2.1 char_type . . . . .	2266
5.422.3 Member Function Documentation . . . . .	2267
5.422.3.1 do_is . . . . .	2267
5.422.3.2 do_is . . . . .	2267
5.422.3.3 do_narrow . . . . .	2268
5.422.3.4 do_narrow . . . . .	2268
5.422.3.5 do_scan_is . . . . .	2269
5.422.3.6 do_scan_not . . . . .	2269
5.422.3.7 do_tolower . . . . .	2270
5.422.3.8 do_tolower . . . . .	2270
5.422.3.9 do_toupper . . . . .	2271
5.422.3.10do_toupper . . . . .	2271
5.422.3.11do_widen . . . . .	2272
5.422.3.12do_widen . . . . .	2272
5.422.3.13s . . . . .	2273
5.422.3.14is . . . . .	2273
5.422.3.15narrow . . . . .	2274
5.422.3.16narrow . . . . .	2274
5.422.3.17scan_is . . . . .	2275
5.422.3.18scan_not . . . . .	2275
5.422.3.19tolower . . . . .	2276
5.422.3.20tolower . . . . .	2276
5.422.3.21toupper . . . . .	2276
5.422.3.22toupper . . . . .	2277
5.422.3.23widen . . . . .	2277
5.422.3.24widen . . . . .	2278
5.422.4 Member Data Documentation . . . . .	2278
5.422.4.1 id . . . . .	2278
5.423std::ctype_byname< char > Class Template Reference . . . . .	2279
5.423.1 Detailed Description . . . . .	2281

---

---

5.423.2 Member Typedef Documentation . . . . .	2281
5.423.2.1 char_type . . . . .	2281
5.423.3 Member Function Documentation . . . . .	2282
5.423.3.1 classic_table . . . . .	2282
5.423.3.2 do_narrow . . . . .	2282
5.423.3.3 do_narrow . . . . .	2282
5.423.3.4 do_tolower . . . . .	2283
5.423.3.5 do_tolower . . . . .	2283
5.423.3.6 do_toupper . . . . .	2284
5.423.3.7 do_toupper . . . . .	2284
5.423.3.8 do_widen . . . . .	2285
5.423.3.9 do_widen . . . . .	2285
5.423.3.10is . . . . .	2286
5.423.3.11is . . . . .	2286
5.423.3.12narrow . . . . .	2286
5.423.3.13narrow . . . . .	2287
5.423.3.14scan_is . . . . .	2288
5.423.3.15scan_not . . . . .	2288
5.423.3.16table . . . . .	2288
5.423.3.17tolower . . . . .	2289
5.423.3.18tolower . . . . .	2289
5.423.3.19toupper . . . . .	2289
5.423.3.20toupper . . . . .	2290
5.423.3.21widen . . . . .	2290
5.423.3.22widen . . . . .	2291
5.423.4 Member Data Documentation . . . . .	2291
5.423.4.1 id . . . . .	2291
5.423.4.2 table_size . . . . .	2291
5.424std::decay<_Tp> Class Template Reference . . . . .	2292
5.424.1 Detailed Description . . . . .	2292
5.425std::decimal::decimal128 Class Reference . . . . .	2292

---

5.425.1 Detailed Description . . . . .	2294
5.425.2 Constructor & Destructor Documentation . . . . .	2294
5.425.2.1 decimal128 . . . . .	2294
5.426std::decimal::decimal32 Class Reference . . . . .	2294
5.426.1 Detailed Description . . . . .	2296
5.426.2 Constructor & Destructor Documentation . . . . .	2296
5.426.2.1 decimal32 . . . . .	2296
5.427std::decimal::decimal64 Class Reference . . . . .	2296
5.427.1 Detailed Description . . . . .	2298
5.427.2 Constructor & Destructor Documentation . . . . .	2298
5.427.2.1 decimal64 . . . . .	2298
5.428std::default_delete< _Tp > Struct Template Reference . . . . .	2298
5.428.1 Detailed Description . . . . .	2299
5.429std::default_delete< _Tp[]> Struct Template Reference . . . . .	2299
5.429.1 Detailed Description . . . . .	2299
5.430std::defer_lock_t Struct Reference . . . . .	2299
5.430.1 Detailed Description . . . . .	2299
5.431std::deque< _Tp, _Alloc > Class Template Reference . . . . .	2300
5.431.1 Detailed Description . . . . .	2304
5.431.2 Constructor & Destructor Documentation . . . . .	2306
5.431.2.1 deque . . . . .	2306
5.431.2.2 deque . . . . .	2306
5.431.2.3 deque . . . . .	2307
5.431.2.4 deque . . . . .	2307
5.431.2.5 deque . . . . .	2307
5.431.2.6 deque . . . . .	2308
5.431.2.7 deque . . . . .	2308
5.431.2.8 ~deque . . . . .	2309
5.431.3 Member Function Documentation . . . . .	2309
5.431.3.1 _M_fill_initialize . . . . .	2309
5.431.3.2 _M_initialize_map . . . . .	2309

---

---

5.431.3.3	<code>_M_new_elements_at_back</code>	2310
5.431.3.4	<code>_M_new_elements_at_front</code>	2310
5.431.3.5	<code>_M_pop_back_aux</code>	2310
5.431.3.6	<code>_M_pop_front_aux</code>	2311
5.431.3.7	<code>_M_push_back_aux</code>	2311
5.431.3.8	<code>_M_push_front_aux</code>	2311
5.431.3.9	<code>_M_range_check</code>	2311
5.431.3.10	<code>_M_range_initialize</code>	2311
5.431.3.11	<code>_M_range_initialize</code>	2312
5.431.3.12	<code>_M_reallocate_map</code>	2312
5.431.3.13	<code>_M_reserve_elements_at_back</code>	2313
5.431.3.14	<code>_M_reserve_elements_at_front</code>	2313
5.431.3.15	<code>_M_reserve_map_at_back</code>	2313
5.431.3.16	<code>_M_reserve_map_at_front</code>	2314
5.431.3.17	<code>assign</code>	2314
5.431.3.18	<code>assign</code>	2314
5.431.3.19	<code>assign</code>	2315
5.431.3.20	<code>at</code>	2315
5.431.3.21	<code>at</code>	2316
5.431.3.22	<code>back</code>	2316
5.431.3.23	<code>back</code>	2316
5.431.3.24	<code>begin</code>	2316
5.431.3.25	<code>begin</code>	2317
5.431.3.26	<code>cbegin</code>	2317
5.431.3.27	<code>end</code>	2317
5.431.3.28	<code>clear</code>	2317
5.431.3.29	<code>crbegin</code>	2318
5.431.3.30	<code>crend</code>	2318
5.431.3.31	<code>emplace</code>	2318
5.431.3.32	<code>empty</code>	2318
5.431.3.33	<code>end</code>	2319

---

5.431.3.34	end	2319
5.431.3.35	erase	2319
5.431.3.36	erase	2320
5.431.3.37	front	2320
5.431.3.38	front	2320
5.431.3.39	get_allocator	2321
5.431.3.40	insert	2321
5.431.3.41	insert	2321
5.431.3.42	insert	2322
5.431.3.43	insert	2322
5.431.3.44	insert	2323
5.431.3.45	max_size	2323
5.431.3.46	operator=	2323
5.431.3.47	operator=	2324
5.431.3.48	operator=	2324
5.431.3.49	operator[]	2324
5.431.3.50	operator[]	2325
5.431.3.51	pop_back	2325
5.431.3.52	pop_front	2325
5.431.3.53	push_back	2326
5.431.3.54	push_front	2326
5.431.3.55	rbegin	2327
5.431.3.56	rbegin	2327
5.431.3.57	rend	2327
5.431.3.58	rend	2327
5.431.3.59	resize	2327
5.431.3.60	shrink_to_fit	2328
5.431.3.61	size	2328
5.431.3.62	swap	2328
5.432	std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2329

---

---

5.432.1 Detailed Description . . . . .	2330
5.432.2 Member Typedef Documentation . . . . .	2330
5.432.2.1 result_type . . . . .	2330
5.432.3 Constructor & Destructor Documentation . . . . .	2330
5.432.3.1 discard_block_engine . . . . .	2330
5.432.3.2 discard_block_engine . . . . .	2331
5.432.3.3 discard_block_engine . . . . .	2331
5.432.3.4 discard_block_engine . . . . .	2331
5.432.3.5 discard_block_engine . . . . .	2332
5.432.4 Member Function Documentation . . . . .	2332
5.432.4.1 base . . . . .	2332
5.432.4.2 discard . . . . .	2332
5.432.4.3 max . . . . .	2332
5.432.4.4 min . . . . .	2333
5.432.4.5 operator() . . . . .	2333
5.432.4.6 seed . . . . .	2333
5.432.4.7 seed . . . . .	2333
5.432.4.8 seed . . . . .	2334
5.432.5 Friends And Related Function Documentation . . . . .	2334
5.432.5.1 operator<< . . . . .	2334
5.432.5.2 operator== . . . . .	2334
5.432.5.3 operator>> . . . . .	2335
5.433std::discrete_distribution<_IntType > Class Template Reference . . . . .	2335
5.433.1 Detailed Description . . . . .	2336
5.433.2 Member Typedef Documentation . . . . .	2337
5.433.2.1 result_type . . . . .	2337
5.433.3 Member Function Documentation . . . . .	2337
5.433.3.1 max . . . . .	2337
5.433.3.2 min . . . . .	2337
5.433.3.3 operator() . . . . .	2337
5.433.3.4 param . . . . .	2337

---

5.433.3.5 param . . . . .	2338
5.433.3.6 probabilities . . . . .	2338
5.433.3.7 reset . . . . .	2338
5.433.4 Friends And Related Function Documentation . . . . .	2338
5.433.4.1 operator<< . . . . .	2338
5.433.4.2 operator>> . . . . .	2339
5.434std::discrete_distribution< _IntType >::param_type Struct Reference	2339
5.434.1 Detailed Description . . . . .	2340
5.435std::divides< _Tp > Struct Template Reference . . . . .	2340
5.435.1 Detailed Description . . . . .	2341
5.435.2 Member Typedef Documentation . . . . .	2341
5.435.2.1 first_argument_type . . . . .	2341
5.435.2.2 result_type . . . . .	2341
5.435.2.3 second_argument_type . . . . .	2341
5.436std::domain_error Class Reference . . . . .	2342
5.436.1 Detailed Description . . . . .	2342
5.436.2 Member Function Documentation . . . . .	2342
5.436.2.1 what . . . . .	2342
5.437std::enable_if< bool, _Tp > Struct Template Reference . . . . .	2343
5.437.1 Detailed Description . . . . .	2343
5.438std::enable_shared_from_this< _Tp > Class Template Reference . . .	2343
5.438.1 Detailed Description . . . . .	2344
5.439std::equal_to< _Tp > Struct Template Reference . . . . .	2344
5.439.1 Detailed Description . . . . .	2345
5.439.2 Member Typedef Documentation . . . . .	2346
5.439.2.1 first_argument_type . . . . .	2346
5.439.2.2 result_type . . . . .	2346
5.439.2.3 second_argument_type . . . . .	2346
5.440std::error_category Class Reference . . . . .	2346
5.440.1 Detailed Description . . . . .	2347
5.441std::error_code Struct Reference . . . . .	2347

---

5.441.1 Detailed Description . . . . .	2347
5.442std::error_condition Struct Reference . . . . .	2348
5.442.1 Detailed Description . . . . .	2348
5.443std::exception Class Reference . . . . .	2348
5.443.1 Detailed Description . . . . .	2350
5.443.2 Member Function Documentation . . . . .	2350
5.443.2.1 what . . . . .	2350
5.444std::exponential_distribution< _RealType > Class Template Reference	2350
5.444.1 Detailed Description . . . . .	2351
5.444.2 Member Typedef Documentation . . . . .	2351
5.444.2.1 result_type . . . . .	2351
5.444.3 Constructor & Destructor Documentation . . . . .	2352
5.444.3.1 exponential_distribution . . . . .	2352
5.444.4 Member Function Documentation . . . . .	2352
5.444.4.1 lambda . . . . .	2352
5.444.4.2 max . . . . .	2352
5.444.4.3 min . . . . .	2352
5.444.4.4 operator() . . . . .	2352
5.444.4.5 param . . . . .	2353
5.444.4.6 param . . . . .	2353
5.444.4.7 reset . . . . .	2353
5.445std::exponential_distribution< _RealType >::param_type Struct Reference	2354
5.445.1 Detailed Description . . . . .	2354
5.446std::extreme_value_distribution< _RealType > Class Template Reference	2354
5.446.1 Detailed Description . . . . .	2355
5.446.2 Member Typedef Documentation . . . . .	2355
5.446.2.1 result_type . . . . .	2355
5.446.3 Member Function Documentation . . . . .	2356
5.446.3.1 a . . . . .	2356
5.446.3.2 b . . . . .	2356

---

5.446.3.3 max	2356
5.446.3.4 min	2356
5.446.3.5 operator()	2356
5.446.3.6 param	2357
5.446.3.7 param	2357
5.446.3.8 reset	2357
5.447std::extreme_value_distribution< _RealType >::param_type Struct Reference	2357
5.447.1 Detailed Description	2358
5.448std::fisher_f_distribution< _RealType > Class Template Reference	2358
5.448.1 Detailed Description	2359
5.448.2 Member Typedef Documentation	2359
5.448.2.1 result_type	2359
5.448.3 Member Function Documentation	2360
5.448.3.1 max	2360
5.448.3.2 min	2360
5.448.3.3 operator()	2360
5.448.3.4 param	2360
5.448.3.5 param	2360
5.448.3.6 reset	2361
5.448.4 Friends And Related Function Documentation	2361
5.448.4.1 operator<<	2361
5.448.4.2 operator==	2361
5.448.4.3 operator>>	2362
5.449std::fisher_f_distribution< _RealType >::param_type Struct Reference	2362
5.449.1 Detailed Description	2363
5.450std::forward_iterator_tag Struct Reference	2363
5.450.1 Detailed Description	2364
5.451std::forward_list< _Tp, _Alloc > Class Template Reference	2364
5.451.1 Detailed Description	2367
5.451.2 Constructor & Destructor Documentation	2368

---

---

5.451.2.1 forward_list . . . . .	2368
5.451.2.2 forward_list . . . . .	2368
5.451.2.3 forward_list . . . . .	2368
5.451.2.4 forward_list . . . . .	2368
5.451.2.5 forward_list . . . . .	2369
5.451.2.6 forward_list . . . . .	2369
5.451.2.7 forward_list . . . . .	2370
5.451.2.8 forward_list . . . . .	2370
5.451.2.9 forward_list . . . . .	2370
5.451.2.10~forward_list . . . . .	2371
5.451.3 Member Function Documentation . . . . .	2371
5.451.3.1 assign . . . . .	2371
5.451.3.2 assign . . . . .	2371
5.451.3.3 assign . . . . .	2372
5.451.3.4 before_begin . . . . .	2372
5.451.3.5 before_begin . . . . .	2372
5.451.3.6 begin . . . . .	2372
5.451.3.7 begin . . . . .	2373
5.451.3.8 cbefore_begin . . . . .	2373
5.451.3.9 cbegin . . . . .	2373
5.451.3.10cend . . . . .	2373
5.451.3.11clear . . . . .	2373
5.451.3.12emplace_after . . . . .	2374
5.451.3.13emplace_front . . . . .	2374
5.451.3.14empty . . . . .	2375
5.451.3.15end . . . . .	2375
5.451.3.16end . . . . .	2375
5.451.3.17erase_after . . . . .	2375
5.451.3.18erase_after . . . . .	2376
5.451.3.19front . . . . .	2376
5.451.3.20front . . . . .	2376

---

5.451.3.2	lget_allocator	2377
5.451.3.2	insert_after	2377
5.451.3.2	insert_after	2377
5.451.3.2	insert_after	2378
5.451.3.2	insert_after	2378
5.451.3.2	max_size	2379
5.451.3.2	merge	2379
5.451.3.2	merge	2379
5.451.3.2	operator=	2380
5.451.3.2	operator=	2380
5.451.3.3	operator=	2380
5.451.3.3	pop_front	2381
5.451.3.3	push_front	2381
5.451.3.3	remove	2381
5.451.3.3	remove_if	2382
5.451.3.3	resize	2382
5.451.3.3	resize	2382
5.451.3.3	reverse	2383
5.451.3.3	sort	2383
5.451.3.3	sort	2383
5.451.3.4	splice_after	2384
5.451.3.4	splice_after	2384
5.451.3.4	splice_after	2384
5.451.3.4	swap	2385
5.451.3.4	unique	2385
5.451.3.4	unique	2386
5.452	std::fpos<_StateT> Class Template Reference	2386
5.452.1	Detailed Description	2387
5.452.2	Constructor & Destructor Documentation	2387
5.452.2.1	fpos	2387
5.452.3	Member Function Documentation	2387

---

---

5.452.3.1 operator streamoff . . . . .	2387
5.452.3.2 operator+ . . . . .	2387
5.452.3.3 operator+= . . . . .	2387
5.452.3.4 operator- . . . . .	2388
5.452.3.5 operator- . . . . .	2388
5.452.3.6 operator-= . . . . .	2388
5.452.3.7 state . . . . .	2388
5.452.3.8 state . . . . .	2388
5.453std::front_insert_iterator< _Container > Class Template Reference . . . . .	2389
5.453.1 Detailed Description . . . . .	2390
5.453.2 Member Typedef Documentation . . . . .	2390
5.453.2.1 container_type . . . . .	2390
5.453.2.2 difference_type . . . . .	2390
5.453.2.3 iterator_category . . . . .	2390
5.453.2.4 pointer . . . . .	2391
5.453.2.5 reference . . . . .	2391
5.453.2.6 value_type . . . . .	2391
5.453.3 Constructor & Destructor Documentation . . . . .	2391
5.453.3.1 front_insert_iterator . . . . .	2391
5.453.4 Member Function Documentation . . . . .	2391
5.453.4.1 operator* . . . . .	2391
5.453.4.2 operator++ . . . . .	2392
5.453.4.3 operator++ . . . . .	2392
5.453.4.4 operator= . . . . .	2392
5.454std::function< _Res(_ArgTypes...)> Class Template Reference . . . . .	2392
5.454.1 Detailed Description . . . . .	2394
5.454.2 Constructor & Destructor Documentation . . . . .	2395
5.454.2.1 function . . . . .	2395
5.454.2.2 function . . . . .	2395
5.454.2.3 function . . . . .	2395
5.454.2.4 function . . . . .	2395

---

5.454.2.5 function . . . . .	2396
5.454.3 Member Function Documentation . . . . .	2396
5.454.3.1 operator bool . . . . .	2396
5.454.3.2 operator() . . . . .	2397
5.454.3.3 operator= . . . . .	2397
5.454.3.4 operator= . . . . .	2397
5.454.3.5 operator= . . . . .	2398
5.454.3.6 operator= . . . . .	2398
5.454.3.7 operator= . . . . .	2399
5.454.3.8 swap . . . . .	2399
5.454.3.9 target . . . . .	2399
5.454.3.10target . . . . .	2400
5.454.3.11target_type . . . . .	2400
5.455std::future< _Res > Class Template Reference . . . . .	2400
5.455.1 Detailed Description . . . . .	2402
5.455.2 Constructor & Destructor Documentation . . . . .	2402
5.455.2.1 future . . . . .	2402
5.455.3 Member Function Documentation . . . . .	2402
5.455.3.1 _M_get_result . . . . .	2402
5.455.3.2 get . . . . .	2403
5.456std::future< _Res & > Class Template Reference . . . . .	2403
5.456.1 Detailed Description . . . . .	2404
5.456.2 Constructor & Destructor Documentation . . . . .	2405
5.456.2.1 future . . . . .	2405
5.456.3 Member Function Documentation . . . . .	2405
5.456.3.1 _M_get_result . . . . .	2405
5.456.3.2 get . . . . .	2405
5.457std::future< void > Class Template Reference . . . . .	2405
5.457.1 Detailed Description . . . . .	2407
5.457.2 Constructor & Destructor Documentation . . . . .	2407
5.457.2.1 future . . . . .	2407

---

---

5.457.3 Member Function Documentation . . . . .	2407
5.457.3.1 <code>_M_get_result</code> . . . . .	2407
5.457.3.2 <code>get</code> . . . . .	2407
5.458 <code>std::future_error</code> Class Reference . . . . .	2408
5.458.1 Detailed Description . . . . .	2408
5.458.2 Member Function Documentation . . . . .	2409
5.458.2.1 <code>what</code> . . . . .	2409
5.459 <code>std::gamma_distribution&lt;_RealType&gt;</code> Class Template Reference . . . . .	2409
5.459.1 Detailed Description . . . . .	2410
5.459.2 Member Typedef Documentation . . . . .	2411
5.459.2.1 <code>result_type</code> . . . . .	2411
5.459.3 Constructor & Destructor Documentation . . . . .	2411
5.459.3.1 <code>gamma_distribution</code> . . . . .	2411
5.459.4 Member Function Documentation . . . . .	2411
5.459.4.1 <code>alpha</code> . . . . .	2411
5.459.4.2 <code>beta</code> . . . . .	2411
5.459.4.3 <code>max</code> . . . . .	2411
5.459.4.4 <code>min</code> . . . . .	2412
5.459.4.5 <code>operator()</code> . . . . .	2412
5.459.4.6 <code>operator()</code> . . . . .	2412
5.459.4.7 <code>param</code> . . . . .	2412
5.459.4.8 <code>param</code> . . . . .	2413
5.459.4.9 <code>reset</code> . . . . .	2413
5.459.5 Friends And Related Function Documentation . . . . .	2413
5.459.5.1 <code>operator&lt;&lt;</code> . . . . .	2413
5.459.5.2 <code>operator==</code> . . . . .	2414
5.459.5.3 <code>operator&gt;&gt;</code> . . . . .	2414
5.460 <code>std::gamma_distribution&lt;_RealType&gt;::param_type</code> Struct Reference . . . . .	2414
5.460.1 Detailed Description . . . . .	2415
5.461 <code>std::geometric_distribution&lt;_IntType&gt;</code> Class Template Reference . . . . .	2415
5.461.1 Detailed Description . . . . .	2416

---

5.461.2 Member Typedef Documentation . . . . .	2416
5.461.2.1 result_type . . . . .	2416
5.461.3 Member Function Documentation . . . . .	2416
5.461.3.1 max . . . . .	2416
5.461.3.2 min . . . . .	2417
5.461.3.3 operator() . . . . .	2417
5.461.3.4 p . . . . .	2417
5.461.3.5 param . . . . .	2417
5.461.3.6 param . . . . .	2417
5.461.3.7 reset . . . . .	2418
5.462std::geometric_distribution< _IntType >::param_type Struct Reference	2418
5.462.1 Detailed Description . . . . .	2418
5.463std::greater< _Tp > Struct Template Reference . . . . .	2419
5.463.1 Detailed Description . . . . .	2420
5.463.2 Member Typedef Documentation . . . . .	2420
5.463.2.1 first_argument_type . . . . .	2420
5.463.2.2 result_type . . . . .	2420
5.463.2.3 second_argument_type . . . . .	2420
5.464std::greater_equal< _Tp > Struct Template Reference . . . . .	2420
5.464.1 Detailed Description . . . . .	2421
5.464.2 Member Typedef Documentation . . . . .	2422
5.464.2.1 first_argument_type . . . . .	2422
5.464.2.2 result_type . . . . .	2422
5.464.2.3 second_argument_type . . . . .	2422
5.465std::gslice Class Reference . . . . .	2422
5.465.1 Detailed Description . . . . .	2423
5.466std::gslice_array< _Tp > Class Template Reference . . . . .	2423
5.466.1 Detailed Description . . . . .	2424
5.467std::has_nothrow_assign< _Tp > Struct Template Reference . . . . .	2425
5.467.1 Detailed Description . . . . .	2425
5.468std::has_nothrow_copy_constructor< _Tp > Struct Template Reference	2425

---

5.468.1 Detailed Description . . . . .	2425
5.469std::has_nothrow_default_constructor< _Tp > Struct Template Reference . . . . .	2426
5.469.1 Detailed Description . . . . .	2426
5.470std::has_trivial_assign< _Tp > Struct Template Reference . . . . .	2426
5.470.1 Detailed Description . . . . .	2426
5.471std::has_trivial_copy_constructor< _Tp > Struct Template Reference . . . . .	2427
5.471.1 Detailed Description . . . . .	2427
5.472std::has_trivial_default_constructor< _Tp > Struct Template Reference . . . . .	2427
5.472.1 Detailed Description . . . . .	2427
5.473std::has_trivial_destructor< _Tp > Struct Template Reference . . . . .	2427
5.473.1 Detailed Description . . . . .	2428
5.474std::hash< _Tp > Struct Template Reference . . . . .	2428
5.474.1 Detailed Description . . . . .	2430
5.474.2 Member Typedef Documentation . . . . .	2430
5.474.2.1 argument_type . . . . .	2430
5.474.2.2 result_type . . . . .	2430
5.475std::hash< __debug::bitset< _Nb > > Struct Template Reference . . . . .	2430
5.475.1 Detailed Description . . . . .	2431
5.475.2 Member Typedef Documentation . . . . .	2432
5.475.2.1 argument_type . . . . .	2432
5.475.2.2 result_type . . . . .	2432
5.476std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference . . . . .	2432
5.476.1 Detailed Description . . . . .	2433
5.476.2 Member Typedef Documentation . . . . .	2434
5.476.2.1 argument_type . . . . .	2434
5.476.2.2 result_type . . . . .	2434
5.477std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference . . . . .	2434
5.477.1 Detailed Description . . . . .	2435
5.477.2 Member Typedef Documentation . . . . .	2436
5.477.2.1 argument_type . . . . .	2436

---

5.477.2.2 result_type . . . . .	2436
5.478std::hash< __gnu_cxx::throw_value_random > Struct Template Reference . . . . .	2436
5.478.1 Detailed Description . . . . .	2437
5.478.2 Member Typedef Documentation . . . . .	2438
5.478.2.1 argument_type . . . . .	2438
5.478.2.2 result_type . . . . .	2438
5.479std::hash< __profile::bitset< _Nb > > Struct Template Reference . . . . .	2438
5.479.1 Detailed Description . . . . .	2439
5.479.2 Member Typedef Documentation . . . . .	2440
5.479.2.1 argument_type . . . . .	2440
5.479.2.2 result_type . . . . .	2440
5.480std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference . . . . .	2440
5.480.1 Detailed Description . . . . .	2441
5.480.2 Member Typedef Documentation . . . . .	2442
5.480.2.1 argument_type . . . . .	2442
5.480.2.2 result_type . . . . .	2442
5.481std::hash< _Tp * > Struct Template Reference . . . . .	2442
5.481.1 Detailed Description . . . . .	2443
5.481.2 Member Typedef Documentation . . . . .	2444
5.481.2.1 argument_type . . . . .	2444
5.481.2.2 result_type . . . . .	2444
5.482std::hash< error_code > Struct Template Reference . . . . .	2444
5.482.1 Detailed Description . . . . .	2445
5.482.2 Member Typedef Documentation . . . . .	2446
5.482.2.1 argument_type . . . . .	2446
5.482.2.2 result_type . . . . .	2446
5.483std::hash< string > Struct Template Reference . . . . .	2446
5.483.1 Detailed Description . . . . .	2447
5.483.2 Member Typedef Documentation . . . . .	2448
5.483.2.1 argument_type . . . . .	2448

---

5.483.2.2 result_type . . . . .	2448
5.484std::hash< thread::id > Struct Template Reference . . . . .	2448
5.484.1 Detailed Description . . . . .	2449
5.484.2 Member Typedef Documentation . . . . .	2450
5.484.2.1 argument_type . . . . .	2450
5.484.2.2 result_type . . . . .	2450
5.485std::hash< u16string > Struct Template Reference . . . . .	2450
5.485.1 Detailed Description . . . . .	2451
5.485.2 Member Typedef Documentation . . . . .	2452
5.485.2.1 argument_type . . . . .	2452
5.485.2.2 result_type . . . . .	2452
5.486std::hash< u32string > Struct Template Reference . . . . .	2452
5.486.1 Detailed Description . . . . .	2453
5.486.2 Member Typedef Documentation . . . . .	2454
5.486.2.1 argument_type . . . . .	2454
5.486.2.2 result_type . . . . .	2454
5.487std::hash< wstring > Struct Template Reference . . . . .	2454
5.487.1 Detailed Description . . . . .	2455
5.487.2 Member Typedef Documentation . . . . .	2456
5.487.2.1 argument_type . . . . .	2456
5.487.2.2 result_type . . . . .	2456
5.488std::hash<::bitset< _Nb > > Struct Template Reference . . . . .	2456
5.488.1 Detailed Description . . . . .	2457
5.488.2 Member Typedef Documentation . . . . .	2458
5.488.2.1 argument_type . . . . .	2458
5.488.2.2 result_type . . . . .	2458
5.489std::hash<::vector< bool, _Alloc > > Struct Template Reference . . . . .	2458
5.489.1 Detailed Description . . . . .	2459
5.489.2 Member Typedef Documentation . . . . .	2460
5.489.2.1 argument_type . . . . .	2460
5.489.2.2 result_type . . . . .	2460

---

5.490	std::identity< _Tp > Struct Template Reference . . . . .	2460
5.490.1	Detailed Description . . . . .	2460
5.491	std::independent_bits_engine< _RandomNumberEngine, __w, _- UIntType > Class Template Reference . . . . .	2461
5.491.1	Detailed Description . . . . .	2462
5.491.2	Member Typedef Documentation . . . . .	2462
5.491.2.1	result_type . . . . .	2462
5.491.3	Constructor & Destructor Documentation . . . . .	2462
5.491.3.1	independent_bits_engine . . . . .	2462
5.491.3.2	independent_bits_engine . . . . .	2462
5.491.3.3	independent_bits_engine . . . . .	2463
5.491.3.4	independent_bits_engine . . . . .	2463
5.491.3.5	independent_bits_engine . . . . .	2463
5.491.4	Member Function Documentation . . . . .	2464
5.491.4.1	base . . . . .	2464
5.491.4.2	discard . . . . .	2464
5.491.4.3	max . . . . .	2464
5.491.4.4	min . . . . .	2464
5.491.4.5	operator() . . . . .	2465
5.491.4.6	seed . . . . .	2465
5.491.4.7	seed . . . . .	2465
5.491.4.8	seed . . . . .	2465
5.491.5	Friends And Related Function Documentation . . . . .	2466
5.491.5.1	operator== . . . . .	2466
5.491.5.2	operator>> . . . . .	2466
5.492	std::indirect_array< _Tp > Class Template Reference . . . . .	2467
5.492.1	Detailed Description . . . . .	2468
5.492.2	Member Function Documentation . . . . .	2468
5.492.2.1	operator%= . . . . .	2468
5.492.2.2	operator&= . . . . .	2469
5.492.2.3	operator*= . . . . .	2469

---

5.492.2.4	operator+=	2469
5.492.2.5	operator-=	2469
5.492.2.6	operator/=	2469
5.492.2.7	operator<<=	2469
5.492.2.8	operator>>=	2469
5.492.2.9	operator^=	2469
5.492.2.10	operator =	2470
5.493	std::initializer_list<_E> Class Template Reference	2470
5.493.1	Detailed Description	2470
5.494	std::input_iterator_tag Struct Reference	2471
5.494.1	Detailed Description	2471
5.495	std::insert_iterator<_Container> Class Template Reference	2472
5.495.1	Detailed Description	2473
5.495.2	Member Typedef Documentation	2473
5.495.2.1	container_type	2473
5.495.2.2	difference_type	2473
5.495.2.3	iterator_category	2473
5.495.2.4	pointer	2474
5.495.2.5	reference	2474
5.495.2.6	value_type	2474
5.495.3	Constructor & Destructor Documentation	2474
5.495.3.1	insert_iterator	2474
5.495.4	Member Function Documentation	2474
5.495.4.1	operator*	2474
5.495.4.2	operator++	2475
5.495.4.3	operator++	2475
5.495.4.4	operator=	2475
5.496	std::invalid_argument Class Reference	2476
5.496.1	Detailed Description	2476
5.496.2	Member Function Documentation	2476
5.496.2.1	what	2476

---

5.497std::ios_base Class Reference . . . . .	2477
5.497.1 Detailed Description . . . . .	2480
5.497.2 Member Typedef Documentation . . . . .	2480
5.497.2.1 event_callback . . . . .	2480
5.497.2.2 fmtflags . . . . .	2481
5.497.2.3 iostate . . . . .	2482
5.497.2.4 openmode . . . . .	2482
5.497.2.5 seekdir . . . . .	2482
5.497.3 Member Enumeration Documentation . . . . .	2483
5.497.3.1 event . . . . .	2483
5.497.4 Constructor & Destructor Documentation . . . . .	2483
5.497.4.1 ~ios_base . . . . .	2483
5.497.5 Member Function Documentation . . . . .	2483
5.497.5.1 _M_getloc . . . . .	2483
5.497.5.2 flags . . . . .	2484
5.497.5.3 flags . . . . .	2484
5.497.5.4 getloc . . . . .	2484
5.497.5.5 imbue . . . . .	2485
5.497.5.6 iword . . . . .	2485
5.497.5.7 precision . . . . .	2485
5.497.5.8 precision . . . . .	2486
5.497.5.9 pword . . . . .	2486
5.497.5.10register_callback . . . . .	2486
5.497.5.11setf . . . . .	2487
5.497.5.12setf . . . . .	2487
5.497.5.13sync_with_stdio . . . . .	2488
5.497.5.14unsetf . . . . .	2488
5.497.5.15width . . . . .	2488
5.497.5.16width . . . . .	2489
5.497.5.17xalloc . . . . .	2489
5.497.6 Member Data Documentation . . . . .	2489

---

---

5.497.6.1 adjustfield . . . . .	2489
5.497.6.2 app . . . . .	2489
5.497.6.3 ate . . . . .	2490
5.497.6.4 badbit . . . . .	2490
5.497.6.5 basefield . . . . .	2490
5.497.6.6 beg . . . . .	2490
5.497.6.7 binary . . . . .	2490
5.497.6.8 boolalpha . . . . .	2491
5.497.6.9 cur . . . . .	2491
5.497.6.10dec . . . . .	2491
5.497.6.11end . . . . .	2491
5.497.6.12eofbit . . . . .	2491
5.497.6.13failbit . . . . .	2492
5.497.6.14fixed . . . . .	2492
5.497.6.15floatfield . . . . .	2492
5.497.6.16goodbit . . . . .	2492
5.497.6.17hex . . . . .	2493
5.497.6.18n . . . . .	2493
5.497.6.19internal . . . . .	2493
5.497.6.20left . . . . .	2493
5.497.6.21loct . . . . .	2494
5.497.6.22out . . . . .	2494
5.497.6.23right . . . . .	2494
5.497.6.24scientific . . . . .	2494
5.497.6.25showbase . . . . .	2494
5.497.6.26showpoint . . . . .	2494
5.497.6.27showpos . . . . .	2494
5.497.6.28skipws . . . . .	2495
5.497.6.29trunc . . . . .	2495
5.497.6.30unitbuf . . . . .	2495
5.497.6.31uppercase . . . . .	2495

5.498	<code>std::ios_base::failure</code> Class Reference . . . . .	2495
5.498.1	Detailed Description . . . . .	2496
5.498.2	Member Function Documentation . . . . .	2496
5.498.2.1	what . . . . .	2496
5.499	<code>std::is_base_of&lt;_Base, _Derived&gt;</code> Struct Template Reference . . . . .	2497
5.499.1	Detailed Description . . . . .	2497
5.500	<code>std::is_bind_expression&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2497
5.500.1	Detailed Description . . . . .	2497
5.501	<code>std::is_bind_expression&lt;_Bind&lt;_Signature&gt;&gt;</code> Struct Template Reference . . . . .	2498
5.501.1	Detailed Description . . . . .	2498
5.502	<code>std::is_bind_expression&lt;_Bind_result&lt;_Result, _Signature&gt;&gt;</code> Struct Template Reference . . . . .	2498
5.502.1	Detailed Description . . . . .	2498
5.503	<code>std::is_constructible&lt;_Tp, _Args&gt;</code> Struct Template Reference . . . . .	2499
5.503.1	Detailed Description . . . . .	2499
5.504	<code>std::is_convertible&lt;_From, _To&gt;</code> Struct Template Reference . . . . .	2500
5.504.1	Detailed Description . . . . .	2500
5.505	<code>std::is_error_code_enum&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2500
5.505.1	Detailed Description . . . . .	2500
5.506	<code>std::is_error_condition_enum&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2501
5.506.1	Detailed Description . . . . .	2501
5.507	<code>std::is_explicitly_convertible&lt;_From, _To&gt;</code> Struct Template Reference . . . . .	2501
5.507.1	Detailed Description . . . . .	2502
5.508	<code>std::is_lvalue_reference&lt;typename&gt;</code> Struct Template Reference . . . . .	2502
5.508.1	Detailed Description . . . . .	2503
5.509	<code>std::is_placeholder&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2503
5.509.1	Detailed Description . . . . .	2503
5.510	<code>std::is_placeholder&lt;_Placeholder&lt;_Num&gt;&gt;</code> Struct Template Reference . . . . .	2503
5.510.1	Detailed Description . . . . .	2504
5.511	<code>std::is_pod&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2504

---

5.511.1 Detailed Description . . . . .	2504
5.512std::is_reference< _Tp > Struct Template Reference . . . . .	2504
5.512.1 Detailed Description . . . . .	2504
5.513std::is_rvalue_reference< typename > Struct Template Reference . . . . .	2505
5.513.1 Detailed Description . . . . .	2505
5.514std::is_signed< _Tp > Struct Template Reference . . . . .	2505
5.514.1 Detailed Description . . . . .	2505
5.515std::is_standard_layout< _Tp > Struct Template Reference . . . . .	2506
5.515.1 Detailed Description . . . . .	2506
5.516std::is_trivial< _Tp > Struct Template Reference . . . . .	2506
5.516.1 Detailed Description . . . . .	2506
5.517std::is_unsigned< _Tp > Struct Template Reference . . . . .	2506
5.517.1 Detailed Description . . . . .	2507
5.518std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference . . . . .	2507
5.518.1 Detailed Description . . . . .	2508
5.518.2 Member Typedef Documentation . . . . .	2508
5.518.2.1 difference_type . . . . .	2508
5.518.2.2 iterator_category . . . . .	2508
5.518.2.3 pointer . . . . .	2509
5.518.2.4 reference . . . . .	2509
5.518.2.5 value_type . . . . .	2509
5.518.3 Constructor & Destructor Documentation . . . . .	2509
5.518.3.1 istream_iterator . . . . .	2509
5.518.3.2 istream_iterator . . . . .	2509
5.519std::istreambuf_iterator< _CharT, _Traits > Class Template Reference . . . . .	2510
5.519.1 Detailed Description . . . . .	2511
5.519.2 Member Typedef Documentation . . . . .	2511
5.519.2.1 char_type . . . . .	2511
5.519.2.2 difference_type . . . . .	2511
5.519.2.3 int_type . . . . .	2511

---

5.519.2.4	istream_type . . . . .	2512
5.519.2.5	iterator_category . . . . .	2512
5.519.2.6	pointer . . . . .	2512
5.519.2.7	reference . . . . .	2512
5.519.2.8	streambuf_type . . . . .	2512
5.519.2.9	traits_type . . . . .	2512
5.519.2.10	value_type . . . . .	2513
5.519.3	Constructor & Destructor Documentation . . . . .	2513
5.519.3.1	istreambuf_iterator . . . . .	2513
5.519.3.2	istreambuf_iterator . . . . .	2513
5.519.3.3	istreambuf_iterator . . . . .	2513
5.519.4	Member Function Documentation . . . . .	2513
5.519.4.1	equal . . . . .	2513
5.519.4.2	operator* . . . . .	2514
5.519.4.3	operator++ . . . . .	2514
5.519.4.4	operator++ . . . . .	2514
5.520	std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference . . . . .	2514
5.520.1	Detailed Description . . . . .	2515
5.520.2	Member Typedef Documentation . . . . .	2515
5.520.2.1	difference_type . . . . .	2515
5.520.2.2	iterator_category . . . . .	2516
5.520.2.3	pointer . . . . .	2516
5.520.2.4	reference . . . . .	2516
5.520.2.5	value_type . . . . .	2516
5.521	std::iterator_traits< _Iterator > Struct Template Reference . . . . .	2517
5.521.1	Detailed Description . . . . .	2517
5.522	std::iterator_traits< _Tp * > Struct Template Reference . . . . .	2518
5.522.1	Detailed Description . . . . .	2518
5.523	std::iterator_traits< const _Tp * > Struct Template Reference . . . . .	2518
5.523.1	Detailed Description . . . . .	2519

---

---

5.524	<code>std::length_error</code> Class Reference . . . . .	2519
5.524.1	Detailed Description . . . . .	2520
5.524.2	Member Function Documentation . . . . .	2520
5.524.2.1	what . . . . .	2520
5.525	<code>std::less&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2520
5.525.1	Detailed Description . . . . .	2521
5.525.2	Member Typedef Documentation . . . . .	2522
5.525.2.1	first_argument_type . . . . .	2522
5.525.2.2	result_type . . . . .	2522
5.525.2.3	second_argument_type . . . . .	2522
5.526	<code>std::less_equal&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2522
5.526.1	Detailed Description . . . . .	2523
5.526.2	Member Typedef Documentation . . . . .	2524
5.526.2.1	first_argument_type . . . . .	2524
5.526.2.2	result_type . . . . .	2524
5.526.2.3	second_argument_type . . . . .	2524
5.527	<code>std::linear_congruential_engine&lt; _UIntType, __a, __c, __m &gt;</code> Class Template Reference . . . . .	2524
5.527.1	Detailed Description . . . . .	2525
5.527.2	Member Typedef Documentation . . . . .	2526
5.527.2.1	result_type . . . . .	2526
5.527.3	Constructor & Destructor Documentation . . . . .	2526
5.527.3.1	linear_congruential_engine . . . . .	2526
5.527.3.2	linear_congruential_engine . . . . .	2526
5.527.4	Member Function Documentation . . . . .	2527
5.527.4.1	discard . . . . .	2527
5.527.4.2	max . . . . .	2527
5.527.4.3	min . . . . .	2527
5.527.4.4	operator() . . . . .	2528
5.527.4.5	seed . . . . .	2528
5.527.4.6	seed . . . . .	2528

---

5.527.5 Friends And Related Function Documentation . . . . .	2529
5.527.5.1 operator<< . . . . .	2529
5.527.5.2 operator== . . . . .	2529
5.527.5.3 operator>> . . . . .	2530
5.527.6 Member Data Documentation . . . . .	2530
5.527.6.1 increment . . . . .	2530
5.527.6.2 modulus . . . . .	2530
5.527.6.3 multiplier . . . . .	2530
5.528std::list< _Tp, _Alloc > Class Template Reference . . . . .	2531
5.528.1 Detailed Description . . . . .	2535
5.528.2 Constructor & Destructor Documentation . . . . .	2535
5.528.2.1 list . . . . .	2535
5.528.2.2 list . . . . .	2536
5.528.2.3 list . . . . .	2536
5.528.2.4 list . . . . .	2536
5.528.2.5 list . . . . .	2537
5.528.2.6 list . . . . .	2537
5.528.2.7 list . . . . .	2537
5.528.3 Member Function Documentation . . . . .	2538
5.528.3.1 _M_create_node . . . . .	2538
5.528.3.2 assign . . . . .	2538
5.528.3.3 assign . . . . .	2538
5.528.3.4 assign . . . . .	2539
5.528.3.5 back . . . . .	2539
5.528.3.6 back . . . . .	2539
5.528.3.7 begin . . . . .	2539
5.528.3.8 begin . . . . .	2540
5.528.3.9 cbegin . . . . .	2540
5.528.3.10end . . . . .	2540
5.528.3.11clear . . . . .	2540
5.528.3.12rbegin . . . . .	2540

---

5.528.3.13	crend	2541
5.528.3.14	emplace	2541
5.528.3.15	empty	2541
5.528.3.16	end	2542
5.528.3.17	end	2542
5.528.3.18	erase	2542
5.528.3.19	erase	2543
5.528.3.20	front	2543
5.528.3.21	front	2543
5.528.3.22	get_allocator	2543
5.528.3.23	insert	2544
5.528.3.24	insert	2544
5.528.3.25	insert	2545
5.528.3.26	insert	2545
5.528.3.27	insert	2545
5.528.3.28	max_size	2546
5.528.3.29	merge	2546
5.528.3.30	merge	2546
5.528.3.31	operator=	2547
5.528.3.32	operator=	2547
5.528.3.33	operator=	2548
5.528.3.34	pop_back	2548
5.528.3.35	pop_front	2548
5.528.3.36	push_back	2549
5.528.3.37	push_front	2549
5.528.3.38	begin	2549
5.528.3.39	begin	2549
5.528.3.40	remove	2550
5.528.3.41	remove_if	2550
5.528.3.42	rend	2550
5.528.3.43	rend	2550

---

5.528.3.44	resize . . . . .	2551
5.528.3.45	reverse . . . . .	2551
5.528.3.46	size . . . . .	2551
5.528.3.47	sort . . . . .	2551
5.528.3.48	sort . . . . .	2552
5.528.3.49	splice . . . . .	2552
5.528.3.50	splice . . . . .	2552
5.528.3.51	splice . . . . .	2553
5.528.3.52	swap . . . . .	2553
5.528.3.53	unique . . . . .	2554
5.528.3.54	unique . . . . .	2554
5.529	std::locale Class Reference . . . . .	2554
5.529.1	Detailed Description . . . . .	2556
5.529.2	Member Typedef Documentation . . . . .	2556
5.529.2.1	category . . . . .	2556
5.529.3	Constructor & Destructor Documentation . . . . .	2557
5.529.3.1	locale . . . . .	2557
5.529.3.2	locale . . . . .	2557
5.529.3.3	locale . . . . .	2557
5.529.3.4	locale . . . . .	2557
5.529.3.5	locale . . . . .	2558
5.529.3.6	locale . . . . .	2558
5.529.3.7	~locale . . . . .	2558
5.529.4	Member Function Documentation . . . . .	2559
5.529.4.1	classic . . . . .	2559
5.529.4.2	combine . . . . .	2559
5.529.4.3	global . . . . .	2559
5.529.4.4	name . . . . .	2560
5.529.4.5	operator!= . . . . .	2560
5.529.4.6	operator() . . . . .	2560
5.529.4.7	operator= . . . . .	2561

---

5.529.4.8 operator== . . . . .	2561
5.529.5 Friends And Related Function Documentation . . . . .	2561
5.529.5.1 has_facet . . . . .	2561
5.529.5.2 use_facet . . . . .	2562
5.529.6 Member Data Documentation . . . . .	2562
5.529.6.1 all . . . . .	2562
5.529.6.2 collate . . . . .	2562
5.529.6.3 ctype . . . . .	2563
5.529.6.4 messages . . . . .	2563
5.529.6.5 monetary . . . . .	2563
5.529.6.6 none . . . . .	2563
5.529.6.7 numeric . . . . .	2564
5.529.6.8 time . . . . .	2564
5.530std::locale::facet Class Reference . . . . .	2564
5.530.1 Detailed Description . . . . .	2566
5.530.2 Constructor & Destructor Documentation . . . . .	2566
5.530.2.1 facet . . . . .	2566
5.530.2.2 ~facet . . . . .	2566
5.531std::locale::id Class Reference . . . . .	2566
5.531.1 Detailed Description . . . . .	2567
5.531.2 Constructor & Destructor Documentation . . . . .	2567
5.531.2.1 id . . . . .	2567
5.531.3 Friends And Related Function Documentation . . . . .	2567
5.531.3.1 has_facet . . . . .	2567
5.531.3.2 use_facet . . . . .	2568
5.532std::lock_guard< _Mutex > Class Template Reference . . . . .	2568
5.532.1 Detailed Description . . . . .	2569
5.533std::logic_error Class Reference . . . . .	2569
5.533.1 Detailed Description . . . . .	2570
5.533.2 Constructor & Destructor Documentation . . . . .	2570
5.533.2.1 logic_error . . . . .	2570

---

5.533.3 Member Function Documentation . . . . .	2570
5.533.3.1 what . . . . .	2570
5.534std::logical_and< _Tp > Struct Template Reference . . . . .	2570
5.534.1 Detailed Description . . . . .	2571
5.534.2 Member Typedef Documentation . . . . .	2572
5.534.2.1 first_argument_type . . . . .	2572
5.534.2.2 result_type . . . . .	2572
5.534.2.3 second_argument_type . . . . .	2572
5.535std::logical_not< _Tp > Struct Template Reference . . . . .	2572
5.535.1 Detailed Description . . . . .	2573
5.535.2 Member Typedef Documentation . . . . .	2574
5.535.2.1 argument_type . . . . .	2574
5.535.2.2 result_type . . . . .	2574
5.536std::logical_or< _Tp > Struct Template Reference . . . . .	2574
5.536.1 Detailed Description . . . . .	2575
5.536.2 Member Typedef Documentation . . . . .	2576
5.536.2.1 first_argument_type . . . . .	2576
5.536.2.2 result_type . . . . .	2576
5.536.2.3 second_argument_type . . . . .	2576
5.537std::lognormal_distribution< _RealType > Class Template Reference	2576
5.537.1 Detailed Description . . . . .	2577
5.537.2 Member Typedef Documentation . . . . .	2578
5.537.2.1 result_type . . . . .	2578
5.537.3 Member Function Documentation . . . . .	2578
5.537.3.1 max . . . . .	2578
5.537.3.2 min . . . . .	2578
5.537.3.3 operator() . . . . .	2578
5.537.3.4 param . . . . .	2578
5.537.3.5 param . . . . .	2579
5.537.3.6 reset . . . . .	2579
5.537.4 Friends And Related Function Documentation . . . . .	2579

---

5.537.4.1 operator<<	2579
5.537.4.2 operator==	2580
5.537.4.3 operator>>	2580
5.538std::lognormal_distribution<_RealType>::param_type Struct Reference	2580
5.538.1 Detailed Description	2581
5.539std::make_signed<_Tp> Struct Template Reference	2581
5.539.1 Detailed Description	2581
5.540std::make_unsigned<_Tp> Struct Template Reference	2581
5.540.1 Detailed Description	2582
5.541std::map<_Key, _Tp, _Compare, _Alloc> Class Template Reference	2582
5.541.1 Detailed Description	2584
5.541.2 Constructor & Destructor Documentation	2585
5.541.2.1 map	2585
5.541.2.2 map	2585
5.541.2.3 map	2585
5.541.2.4 map	2586
5.541.2.5 map	2586
5.541.2.6 map	2586
5.541.2.7 map	2587
5.541.3 Member Function Documentation	2587
5.541.3.1 at	2587
5.541.3.2 begin	2588
5.541.3.3 begin	2588
5.541.3.4 cbegin	2588
5.541.3.5 cend	2588
5.541.3.6 clear	2589
5.541.3.7 count	2589
5.541.3.8 crbegin	2589
5.541.3.9 crend	2590
5.541.3.10empty	2590

---

5.541.3.1	lend	2590
5.541.3.2	end	2590
5.541.3.3	equal_range	2591
5.541.3.4	equal_range	2591
5.541.3.5	erase	2592
5.541.3.6	erase	2592
5.541.3.7	erase	2593
5.541.3.8	find	2593
5.541.3.9	find	2594
5.541.3.20	get_allocator	2594
5.541.3.2	insert	2594
5.541.3.22	insert	2595
5.541.3.23	insert	2595
5.541.3.24	insert	2596
5.541.3.25	key_comp	2596
5.541.3.26	lower_bound	2596
5.541.3.27	lower_bound	2597
5.541.3.28	max_size	2597
5.541.3.29	operator=	2597
5.541.3.30	operator=	2598
5.541.3.31	operator=	2598
5.541.3.32	operator[]	2599
5.541.3.33	begin	2599
5.541.3.34	begin	2599
5.541.3.35	end	2600
5.541.3.36	end	2600
5.541.3.37	size	2600
5.541.3.38	swap	2600
5.541.3.39	upper_bound	2601
5.541.3.40	upper_bound	2601
5.541.3.41	value_comp	2601

---

5.542std::mask_array< _Tp > Class Template Reference . . . . .	2602
5.542.1 Detailed Description . . . . .	2603
5.542.2 Member Function Documentation . . . . .	2603
5.542.2.1 operator%= . . . . .	2603
5.542.2.2 operator&= . . . . .	2604
5.542.2.3 operator*= . . . . .	2604
5.542.2.4 operator+= . . . . .	2604
5.542.2.5 operator-= . . . . .	2604
5.542.2.6 operator/= . . . . .	2604
5.542.2.7 operator<<= . . . . .	2604
5.542.2.8 operator>>= . . . . .	2604
5.542.2.9 operator^= . . . . .	2604
5.542.2.10operator = . . . . .	2605
5.543std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2605
5.543.1 Detailed Description . . . . .	2606
5.543.2 Member Typedef Documentation . . . . .	2606
5.543.2.1 first_argument_type . . . . .	2606
5.543.2.2 result_type . . . . .	2606
5.543.2.3 second_argument_type . . . . .	2606
5.544std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	2607
5.544.1 Detailed Description . . . . .	2608
5.544.2 Member Typedef Documentation . . . . .	2608
5.544.2.1 first_argument_type . . . . .	2608
5.544.2.2 result_type . . . . .	2608
5.544.2.3 second_argument_type . . . . .	2608
5.545std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . . .	2608
5.545.1 Detailed Description . . . . .	2609
5.545.2 Member Typedef Documentation . . . . .	2610
5.545.2.1 argument_type . . . . .	2610
5.545.2.2 result_type . . . . .	2610
5.546std::mem_fun_t< _Ret, _Tp > Class Template Reference . . . . .	2610

---

5.546.1 Detailed Description . . . . .	2611
5.546.2 Member Typedef Documentation . . . . .	2612
5.546.2.1 argument_type . . . . .	2612
5.546.2.2 result_type . . . . .	2612
5.547std::messages<_CharT> Class Template Reference . . . . .	2612
5.547.1 Detailed Description . . . . .	2614
5.547.2 Member Typedef Documentation . . . . .	2615
5.547.2.1 char_type . . . . .	2615
5.547.2.2 string_type . . . . .	2615
5.547.3 Constructor & Destructor Documentation . . . . .	2615
5.547.3.1 messages . . . . .	2615
5.547.3.2 messages . . . . .	2616
5.547.3.3 ~messages . . . . .	2616
5.547.4 Member Data Documentation . . . . .	2616
5.547.4.1 id . . . . .	2616
5.548std::messages_base Struct Reference . . . . .	2616
5.548.1 Detailed Description . . . . .	2617
5.549std::messages_byname<_CharT> Class Template Reference . . . . .	2617
5.549.1 Detailed Description . . . . .	2619
5.549.2 Member Typedef Documentation . . . . .	2620
5.549.2.1 char_type . . . . .	2620
5.549.2.2 string_type . . . . .	2620
5.549.3 Member Data Documentation . . . . .	2620
5.549.3.1 id . . . . .	2620
5.550std::minus<_Tp> Struct Template Reference . . . . .	2620
5.550.1 Detailed Description . . . . .	2621
5.550.2 Member Typedef Documentation . . . . .	2622
5.550.2.1 first_argument_type . . . . .	2622
5.550.2.2 result_type . . . . .	2622
5.550.2.3 second_argument_type . . . . .	2622
5.551std::modulus<_Tp> Struct Template Reference . . . . .	2622

---

5.551.1 Detailed Description . . . . .	2623
5.551.2 Member Typedef Documentation . . . . .	2624
5.551.2.1 first_argument_type . . . . .	2624
5.551.2.2 result_type . . . . .	2624
5.551.2.3 second_argument_type . . . . .	2624
5.552std::money_base Class Reference . . . . .	2624
5.552.1 Detailed Description . . . . .	2625
5.553std::money_get< _CharT, _InIter > Class Template Reference . . . . .	2626
5.553.1 Detailed Description . . . . .	2628
5.553.2 Member Typedef Documentation . . . . .	2628
5.553.2.1 char_type . . . . .	2628
5.553.2.2 iter_type . . . . .	2628
5.553.2.3 string_type . . . . .	2628
5.553.3 Constructor & Destructor Documentation . . . . .	2629
5.553.3.1 money_get . . . . .	2629
5.553.3.2 ~money_get . . . . .	2629
5.553.4 Member Function Documentation . . . . .	2629
5.553.4.1 do_get . . . . .	2629
5.553.4.2 do_get . . . . .	2630
5.553.4.3 get . . . . .	2630
5.553.4.4 get . . . . .	2631
5.553.5 Member Data Documentation . . . . .	2631
5.553.5.1 id . . . . .	2631
5.554std::money_put< _CharT, _OutIter > Class Template Reference . . . . .	2632
5.554.1 Detailed Description . . . . .	2633
5.554.2 Member Typedef Documentation . . . . .	2634
5.554.2.1 char_type . . . . .	2634
5.554.2.2 iter_type . . . . .	2634
5.554.2.3 string_type . . . . .	2634
5.554.3 Constructor & Destructor Documentation . . . . .	2634
5.554.3.1 money_put . . . . .	2634

---

---

5.554.3.2	~money_put . . . . .	2634
5.554.4	Member Function Documentation . . . . .	2635
5.554.4.1	do_put . . . . .	2635
5.554.4.2	do_put . . . . .	2635
5.554.4.3	put . . . . .	2636
5.554.4.4	put . . . . .	2637
5.554.5	Member Data Documentation . . . . .	2637
5.554.5.1	id . . . . .	2637
5.555	std::moneypunct< _CharT, _Intl > Class Template Reference . . . . .	2638
5.555.1	Detailed Description . . . . .	2640
5.555.2	Member Typedef Documentation . . . . .	2640
5.555.2.1	char_type . . . . .	2640
5.555.2.2	string_type . . . . .	2641
5.555.3	Constructor & Destructor Documentation . . . . .	2641
5.555.3.1	moneypunct . . . . .	2641
5.555.3.2	moneypunct . . . . .	2641
5.555.3.3	moneypunct . . . . .	2641
5.555.3.4	~moneypunct . . . . .	2642
5.555.4	Member Function Documentation . . . . .	2642
5.555.4.1	curr_symbol . . . . .	2642
5.555.4.2	decimal_point . . . . .	2642
5.555.4.3	do_curr_symbol . . . . .	2643
5.555.4.4	do_decimal_point . . . . .	2643
5.555.4.5	do_frac_digits . . . . .	2643
5.555.4.6	do_grouping . . . . .	2644
5.555.4.7	do_neg_format . . . . .	2644
5.555.4.8	do_negative_sign . . . . .	2645
5.555.4.9	do_pos_format . . . . .	2645
5.555.4.10	do_positive_sign . . . . .	2645
5.555.4.11	do_thousands_sep . . . . .	2646
5.555.4.12	frac_digits . . . . .	2646

---

5.555.4.13	grouping	2647
5.555.4.14	neg_format	2647
5.555.4.15	negative_sign	2648
5.555.4.16	pos_format	2648
5.555.4.17	positive_sign	2649
5.555.4.18	thousands_sep	2649
5.555.5	Member Data Documentation	2650
5.555.5.1	id	2650
5.555.5.2	intl	2650
5.556	std::money_punct_byname< _CharT, _Intl > Class Template Reference	2650
5.556.1	Detailed Description	2653
5.556.2	Member Typedef Documentation	2653
5.556.2.1	char_type	2653
5.556.2.2	string_type	2653
5.556.3	Member Function Documentation	2654
5.556.3.1	curr_symbol	2654
5.556.3.2	decimal_point	2654
5.556.3.3	do_curr_symbol	2654
5.556.3.4	do_decimal_point	2655
5.556.3.5	do_frac_digits	2655
5.556.3.6	do_grouping	2656
5.556.3.7	do_neg_format	2656
5.556.3.8	do_negative_sign	2656
5.556.3.9	do_pos_format	2657
5.556.3.10	do_positive_sign	2657
5.556.3.11	do_thousands_sep	2658
5.556.3.12	frac_digits	2658
5.556.3.13	grouping	2658
5.556.3.14	neg_format	2659
5.556.3.15	negative_sign	2660
5.556.3.16	pos_format	2660

---

5.556.3.17	positive_sign . . . . .	2661
5.556.3.18	thousands_sep . . . . .	2661
5.556.4	Member Data Documentation . . . . .	2662
5.556.4.1	id . . . . .	2662
5.556.4.2	intl . . . . .	2662
5.557	std::move_iterator< _Iterator > Class Template Reference . . . . .	2662
5.557.1	Detailed Description . . . . .	2663
5.558	std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . . .	2663
5.558.1	Detailed Description . . . . .	2666
5.558.2	Constructor & Destructor Documentation . . . . .	2666
5.558.2.1	multimap . . . . .	2666
5.558.2.2	multimap . . . . .	2666
5.558.2.3	multimap . . . . .	2667
5.558.2.4	multimap . . . . .	2667
5.558.2.5	multimap . . . . .	2667
5.558.2.6	multimap . . . . .	2668
5.558.2.7	multimap . . . . .	2668
5.558.3	Member Function Documentation . . . . .	2669
5.558.3.1	begin . . . . .	2669
5.558.3.2	begin . . . . .	2669
5.558.3.3	cbegin . . . . .	2669
5.558.3.4	cend . . . . .	2669
5.558.3.5	clear . . . . .	2670
5.558.3.6	count . . . . .	2670
5.558.3.7	crbegin . . . . .	2670
5.558.3.8	crend . . . . .	2670
5.558.3.9	empty . . . . .	2671
5.558.3.10	end . . . . .	2671
5.558.3.11	lend . . . . .	2671
5.558.3.12	equal_range . . . . .	2671

---

5.558.3.13equal_range	2672
5.558.3.14erase	2672
5.558.3.15erase	2673
5.558.3.16erase	2673
5.558.3.17find	2674
5.558.3.18find	2674
5.558.3.19get_allocator	2675
5.558.3.20insert	2675
5.558.3.21insert	2675
5.558.3.22insert	2676
5.558.3.23insert	2676
5.558.3.24key_comp	2677
5.558.3.25lower_bound	2677
5.558.3.26lower_bound	2677
5.558.3.27max_size	2678
5.558.3.28operator=	2678
5.558.3.29operator=	2678
5.558.3.30operator=	2679
5.558.3.31rbegin	2679
5.558.3.32rbegin	2679
5.558.3.33rend	2680
5.558.3.34rend	2680
5.558.3.35size	2680
5.558.3.36swap	2680
5.558.3.37upper_bound	2681
5.558.3.38upper_bound	2681
5.558.3.39value_comp	2681
5.559std::multiplies<_Tp> Struct Template Reference	2682
5.559.1 Detailed Description	2683
5.559.2 Member Typedef Documentation	2683
5.559.2.1 first_argument_type	2683

---

---

5.559.2.2	result_type	2683
5.559.2.3	second_argument_type	2683
5.560	std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2683
5.560.1	Detailed Description	2685
5.560.2	Constructor & Destructor Documentation	2686
5.560.2.1	multiset	2686
5.560.2.2	multiset	2686
5.560.2.3	multiset	2686
5.560.2.4	multiset	2687
5.560.2.5	multiset	2687
5.560.2.6	multiset	2688
5.560.2.7	multiset	2688
5.560.3	Member Function Documentation	2688
5.560.3.1	begin	2688
5.560.3.2	cbegin	2689
5.560.3.3	end	2689
5.560.3.4	clear	2689
5.560.3.5	count	2689
5.560.3.6	crbegin	2690
5.560.3.7	crend	2690
5.560.3.8	empty	2690
5.560.3.9	end	2690
5.560.3.10	equal_range	2690
5.560.3.11	equal_range	2691
5.560.3.12	erase	2691
5.560.3.13	erase	2692
5.560.3.14	erase	2692
5.560.3.15	find	2693
5.560.3.16	find	2693
5.560.3.17	get_allocator	2694
5.560.3.18	insert	2694

---

---

5.560.3.19	insert	2694
5.560.3.20	insert	2695
5.560.3.21	insert	2695
5.560.3.22	key_comp	2696
5.560.3.23	lower_bound	2696
5.560.3.24	lower_bound	2696
5.560.3.25	max_size	2697
5.560.3.26	operator=	2697
5.560.3.27	operator=	2697
5.560.3.28	operator=	2698
5.560.3.29	begin	2698
5.560.3.30	end	2698
5.560.3.31	size	2699
5.560.3.32	swap	2699
5.560.3.33	upper_bound	2699
5.560.3.34	upper_bound	2700
5.560.3.35	value_comp	2700
5.560.4	Friends And Related Function Documentation	2700
5.560.4.1	operator<	2700
5.560.4.2	operator==	2701
5.561	std::mutex Class Reference	2701
5.561.1	Detailed Description	2702
5.562	std::negate< _Tp > Struct Template Reference	2702
5.562.1	Detailed Description	2703
5.562.2	Member Typedef Documentation	2703
5.562.2.1	argument_type	2703
5.562.2.2	result_type	2703
5.563	std::negative_binomial_distribution< _IntType > Class Template Reference	2703
5.563.1	Detailed Description	2705
5.563.2	Member Typedef Documentation	2705

5.563.2.1 result_type . . . . .	2705
5.563.3 Member Function Documentation . . . . .	2705
5.563.3.1 k . . . . .	2705
5.563.3.2 max . . . . .	2705
5.563.3.3 min . . . . .	2705
5.563.3.4 operator() . . . . .	2706
5.563.3.5 p . . . . .	2706
5.563.3.6 param . . . . .	2706
5.563.3.7 param . . . . .	2706
5.563.3.8 reset . . . . .	2706
5.563.4 Friends And Related Function Documentation . . . . .	2707
5.563.4.1 operator<< . . . . .	2707
5.563.4.2 operator== . . . . .	2707
5.563.4.3 operator>> . . . . .	2707
5.564std::negative_binomial_distribution< _IntType >::param_type Struct Reference . . . . .	2708
5.564.1 Detailed Description . . . . .	2708
5.565std::nested_exception Class Reference . . . . .	2709
5.565.1 Detailed Description . . . . .	2709
5.566std::normal_distribution< _RealType > Class Template Reference . . . . .	2709
5.566.1 Detailed Description . . . . .	2711
5.566.2 Member Typedef Documentation . . . . .	2711
5.566.2.1 result_type . . . . .	2711
5.566.3 Constructor & Destructor Documentation . . . . .	2711
5.566.3.1 normal_distribution . . . . .	2711
5.566.4 Member Function Documentation . . . . .	2711
5.566.4.1 max . . . . .	2711
5.566.4.2 mean . . . . .	2712
5.566.4.3 min . . . . .	2712
5.566.4.4 operator() . . . . .	2712
5.566.4.5 operator() . . . . .	2712

---

5.566.4.6 param	2713
5.566.4.7 param	2713
5.566.4.8 reset	2713
5.566.4.9 stddev	2713
5.566.5 Friends And Related Function Documentation	2714
5.566.5.1 operator<<	2714
5.566.5.2 operator==	2714
5.566.5.3 operator>>	2714
5.567std::normal_distribution<_RealType>::param_type Struct Reference	2715
5.567.1 Detailed Description	2715
5.568std::not_equal_to<_Tp> Struct Template Reference	2715
5.568.1 Detailed Description	2716
5.568.2 Member Typedef Documentation	2717
5.568.2.1 first_argument_type	2717
5.568.2.2 result_type	2717
5.568.2.3 second_argument_type	2717
5.569std::num_get<_CharT, _InIter> Class Template Reference	2717
5.569.1 Detailed Description	2720
5.569.2 Member Typedef Documentation	2720
5.569.2.1 char_type	2720
5.569.2.2 iter_type	2721
5.569.3 Constructor & Destructor Documentation	2721
5.569.3.1 num_get	2721
5.569.3.2 ~num_get	2721
5.569.4 Member Function Documentation	2721
5.569.4.1 do_get	2721
5.569.4.2 do_get	2722
5.569.4.3 do_get	2723
5.569.4.4 do_get	2723
5.569.4.5 do_get	2724
5.569.4.6 do_get	2724

---

5.569.4.7 do_get . . . . .	2725
5.569.4.8 do_get . . . . .	2726
5.569.4.9 do_get . . . . .	2726
5.569.4.10do_get . . . . .	2727
5.569.4.1ldo_get . . . . .	2728
5.569.4.12get . . . . .	2728
5.569.4.13get . . . . .	2729
5.569.4.14get . . . . .	2730
5.569.4.15get . . . . .	2731
5.569.4.16get . . . . .	2731
5.569.4.17get . . . . .	2732
5.569.4.18get . . . . .	2733
5.569.4.19get . . . . .	2734
5.569.4.20get . . . . .	2735
5.569.4.21get . . . . .	2735
5.569.4.22get . . . . .	2736
5.569.5 Member Data Documentation . . . . .	2737
5.569.5.1 id . . . . .	2737
5.570std::num_put< _CharT, _OutIter > Class Template Reference . . . . .	2737
5.570.1 Detailed Description . . . . .	2740
5.570.2 Member Typedef Documentation . . . . .	2740
5.570.2.1 char_type . . . . .	2740
5.570.2.2 iter_type . . . . .	2740
5.570.3 Constructor & Destructor Documentation . . . . .	2741
5.570.3.1 num_put . . . . .	2741
5.570.3.2 ~num_put . . . . .	2741
5.570.4 Member Function Documentation . . . . .	2741
5.570.4.1 do_put . . . . .	2741
5.570.4.2 do_put . . . . .	2742
5.570.4.3 do_put . . . . .	2742
5.570.4.4 do_put . . . . .	2743

---

5.570.4.5 do_put . . . . .	2743
5.570.4.6 do_put . . . . .	2744
5.570.4.7 do_put . . . . .	2744
5.570.4.8 do_put . . . . .	2745
5.570.4.9 put . . . . .	2745
5.570.4.10put . . . . .	2746
5.570.4.11put . . . . .	2747
5.570.4.12put . . . . .	2748
5.570.4.13put . . . . .	2748
5.570.4.14put . . . . .	2749
5.570.4.15put . . . . .	2750
5.570.4.16put . . . . .	2751
5.570.5 Member Data Documentation . . . . .	2752
5.570.5.1 id . . . . .	2752
5.571std::numeric_limits< _Tp > Struct Template Reference . . . . .	2752
5.571.1 Detailed Description . . . . .	2754
5.571.2 Member Function Documentation . . . . .	2754
5.571.2.1 denorm_min . . . . .	2754
5.571.2.2 epsilon . . . . .	2754
5.571.2.3 infinity . . . . .	2754
5.571.2.4 lowest . . . . .	2754
5.571.2.5 max . . . . .	2755
5.571.2.6 min . . . . .	2755
5.571.2.7 quiet_NaN . . . . .	2755
5.571.2.8 round_error . . . . .	2755
5.571.2.9 signaling_NaN . . . . .	2755
5.571.3 Member Data Documentation . . . . .	2755
5.571.3.1 digits . . . . .	2755
5.571.3.2 digits10 . . . . .	2756
5.571.3.3 has_denorm . . . . .	2756
5.571.3.4 has_denorm_loss . . . . .	2756

5.571.3.5	has_infinity	2756
5.571.3.6	has_quiet_NaN	2756
5.571.3.7	has_signaling_NaN	2756
5.571.3.8	is_bounded	2757
5.571.3.9	is_exact	2757
5.571.3.10	is_iec559	2757
5.571.3.11	is_integer	2757
5.571.3.12	is_modulo	2757
5.571.3.13	is_signed	2757
5.571.3.14	is_specialized	2758
5.571.3.15	max_digits10	2758
5.571.3.16	max_exponent	2758
5.571.3.17	max_exponent10	2758
5.571.3.18	min_exponent	2758
5.571.3.19	min_exponent10	2759
5.571.3.20	radix	2759
5.571.3.21	round_style	2759
5.571.3.22	traps_before	2759
5.571.3.23	traps	2759
5.572	std::numeric_limits< bool > Struct Template Reference	2760
5.572.1	Detailed Description	2761
5.573	std::numeric_limits< char > Struct Template Reference	2761
5.573.1	Detailed Description	2762
5.574	std::numeric_limits< char16_t > Struct Template Reference	2762
5.574.1	Detailed Description	2763
5.575	std::numeric_limits< char32_t > Struct Template Reference	2763
5.575.1	Detailed Description	2765
5.576	std::numeric_limits< double > Struct Template Reference	2765
5.576.1	Detailed Description	2766
5.577	std::numeric_limits< float > Struct Template Reference	2766
5.577.1	Detailed Description	2767

---

5.578	<a href="#">std::numeric_limits&lt; int &gt; Struct Template Reference</a>	2767
5.578.1	<a href="#">Detailed Description</a>	2769
5.579	<a href="#">std::numeric_limits&lt; long &gt; Struct Template Reference</a>	2769
5.579.1	<a href="#">Detailed Description</a>	2770
5.580	<a href="#">std::numeric_limits&lt; long double &gt; Struct Template Reference</a>	2770
5.580.1	<a href="#">Detailed Description</a>	2771
5.581	<a href="#">std::numeric_limits&lt; long long &gt; Struct Template Reference</a>	2771
5.581.1	<a href="#">Detailed Description</a>	2773
5.582	<a href="#">std::numeric_limits&lt; short &gt; Struct Template Reference</a>	2773
5.582.1	<a href="#">Detailed Description</a>	2774
5.583	<a href="#">std::numeric_limits&lt; signed char &gt; Struct Template Reference</a>	2774
5.583.1	<a href="#">Detailed Description</a>	2775
5.584	<a href="#">std::numeric_limits&lt; unsigned char &gt; Struct Template Reference</a>	2775
5.584.1	<a href="#">Detailed Description</a>	2777
5.585	<a href="#">std::numeric_limits&lt; unsigned int &gt; Struct Template Reference</a>	2777
5.585.1	<a href="#">Detailed Description</a>	2778
5.586	<a href="#">std::numeric_limits&lt; unsigned long &gt; Struct Template Reference</a>	2778
5.586.1	<a href="#">Detailed Description</a>	2779
5.587	<a href="#">std::numeric_limits&lt; unsigned long long &gt; Struct Template Reference</a>	2779
5.587.1	<a href="#">Detailed Description</a>	2781
5.588	<a href="#">std::numeric_limits&lt; unsigned short &gt; Struct Template Reference</a>	2781
5.588.1	<a href="#">Detailed Description</a>	2782
5.589	<a href="#">std::numeric_limits&lt; wchar_t &gt; Struct Template Reference</a>	2782
5.589.1	<a href="#">Detailed Description</a>	2783
5.590	<a href="#">std::numpunct&lt; _CharT &gt; Class Template Reference</a>	2783
5.590.1	<a href="#">Detailed Description</a>	2786
5.590.2	<a href="#">Member Typedef Documentation</a>	2786
5.590.2.1	<a href="#">char_type</a>	2786
5.590.2.2	<a href="#">string_type</a>	2786
5.590.3	<a href="#">Constructor &amp; Destructor Documentation</a>	2786
5.590.3.1	<a href="#">numpunct</a>	2786

5.590.3.2	numpunct	2787
5.590.3.3	numpunct	2787
5.590.3.4	~numpunct	2787
5.590.4	Member Function Documentation	2787
5.590.4.1	decimal_point	2787
5.590.4.2	do_decimal_point	2788
5.590.4.3	do_falsename	2788
5.590.4.4	do_grouping	2789
5.590.4.5	do_thousands_sep	2789
5.590.4.6	do_truename	2789
5.590.4.7	falsename	2790
5.590.4.8	grouping	2790
5.590.4.9	thousands_sep	2791
5.590.4.10	truename	2791
5.590.5	Member Data Documentation	2791
5.590.5.1	id	2791
5.591	std::numpunct_byname<_CharT> Class Template Reference	2792
5.591.1	Detailed Description	2793
5.591.2	Member Typedef Documentation	2794
5.591.2.1	char_type	2794
5.591.2.2	string_type	2794
5.591.3	Member Function Documentation	2794
5.591.3.1	decimal_point	2794
5.591.3.2	do_decimal_point	2794
5.591.3.3	do_falsename	2795
5.591.3.4	do_grouping	2795
5.591.3.5	do_thousands_sep	2796
5.591.3.6	do_truename	2796
5.591.3.7	falsename	2796
5.591.3.8	grouping	2797
5.591.3.9	thousands_sep	2797

---

5.591.3.10	truename . . . . .	2797
5.591.4	Member Data Documentation . . . . .	2798
5.591.4.1	id . . . . .	2798
5.592	std::once_flag Struct Reference . . . . .	2798
5.592.1	Detailed Description . . . . .	2798
5.592.2	Friends And Related Function Documentation . . . . .	2799
5.592.2.1	call_once . . . . .	2799
5.593	std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	2799
5.593.1	Detailed Description . . . . .	2800
5.593.2	Member Typedef Documentation . . . . .	2800
5.593.2.1	char_type . . . . .	2800
5.593.2.2	difference_type . . . . .	2801
5.593.2.3	iterator_category . . . . .	2801
5.593.2.4	ostream_type . . . . .	2801
5.593.2.5	pointer . . . . .	2801
5.593.2.6	reference . . . . .	2801
5.593.2.7	traits_type . . . . .	2801
5.593.2.8	value_type . . . . .	2802
5.593.3	Constructor & Destructor Documentation . . . . .	2802
5.593.3.1	ostream_iterator . . . . .	2802
5.593.3.2	ostream_iterator . . . . .	2802
5.593.3.3	ostream_iterator . . . . .	2802
5.593.4	Member Function Documentation . . . . .	2803
5.593.4.1	operator= . . . . .	2803
5.594	std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	2803
5.594.1	Detailed Description . . . . .	2804
5.594.2	Member Typedef Documentation . . . . .	2805
5.594.2.1	char_type . . . . .	2805
5.594.2.2	difference_type . . . . .	2805
5.594.2.3	iterator_category . . . . .	2805
5.594.2.4	ostream_type . . . . .	2805

---

5.594.2.5 pointer . . . . .	2805
5.594.2.6 reference . . . . .	2805
5.594.2.7 streambuf_type . . . . .	2806
5.594.2.8 traits_type . . . . .	2806
5.594.2.9 value_type . . . . .	2806
5.594.3 Constructor & Destructor Documentation . . . . .	2806
5.594.3.1 ostreambuf_iterator . . . . .	2806
5.594.3.2 ostreambuf_iterator . . . . .	2806
5.594.4 Member Function Documentation . . . . .	2807
5.594.4.1 failed . . . . .	2807
5.594.4.2 operator* . . . . .	2807
5.594.4.3 operator++ . . . . .	2807
5.594.4.4 operator++ . . . . .	2807
5.594.4.5 operator= . . . . .	2807
5.595std::out_of_range Class Reference . . . . .	2808
5.595.1 Detailed Description . . . . .	2808
5.595.2 Member Function Documentation . . . . .	2808
5.595.2.1 what . . . . .	2808
5.596std::output_iterator_tag Struct Reference . . . . .	2809
5.596.1 Detailed Description . . . . .	2809
5.597std::overflow_error Class Reference . . . . .	2810
5.597.1 Detailed Description . . . . .	2810
5.597.2 Member Function Documentation . . . . .	2810
5.597.2.1 what . . . . .	2810
5.598std::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . . .	2811
5.598.1 Detailed Description . . . . .	2811
5.599std::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . . .	2811
5.599.1 Detailed Description . . . . .	2811
5.600std::packaged_task< _Res(_ArgTypes...)> Class Template Reference . . . . .	2812
5.600.1 Detailed Description . . . . .	2812
5.601std::pair< _T1, _T2 > Struct Template Reference . . . . .	2813

---

---

5.601.1 Detailed Description . . . . .	2814
5.601.2 Member Typedef Documentation . . . . .	2814
5.601.2.1 first_type . . . . .	2814
5.601.2.2 second_type . . . . .	2814
5.601.3 Constructor & Destructor Documentation . . . . .	2815
5.601.3.1 pair . . . . .	2815
5.601.3.2 pair . . . . .	2815
5.601.3.3 pair . . . . .	2815
5.601.4 Member Data Documentation . . . . .	2815
5.601.4.1 first . . . . .	2815
5.601.4.2 second . . . . .	2815
5.602std::piecewise_constant_distribution< _RealType > Class Template Reference . . . . .	2816
5.602.1 Detailed Description . . . . .	2817
5.602.2 Member Typedef Documentation . . . . .	2817
5.602.2.1 result_type . . . . .	2817
5.602.3 Member Function Documentation . . . . .	2817
5.602.3.1 densities . . . . .	2817
5.602.3.2 intervals . . . . .	2817
5.602.3.3 max . . . . .	2818
5.602.3.4 min . . . . .	2818
5.602.3.5 operator() . . . . .	2818
5.602.3.6 param . . . . .	2818
5.602.3.7 param . . . . .	2819
5.602.3.8 reset . . . . .	2819
5.602.4 Friends And Related Function Documentation . . . . .	2819
5.602.4.1 operator<< . . . . .	2819
5.602.4.2 operator>> . . . . .	2820
5.603std::piecewise_constant_distribution< _RealType >::param_type Struct Reference . . . . .	2820
5.603.1 Detailed Description . . . . .	2821

---

5.604	<code>std::piecewise_linear_distribution&lt;_RealType&gt;</code> Class Template Reference	2821
5.604.1	Detailed Description	2822
5.604.2	Member Typedef Documentation	2822
5.604.2.1	<code>result_type</code>	2822
5.604.3	Member Function Documentation	2823
5.604.3.1	<code>densities</code>	2823
5.604.3.2	<code>intervals</code>	2823
5.604.3.3	<code>max</code>	2823
5.604.3.4	<code>min</code>	2823
5.604.3.5	<code>operator()</code>	2823
5.604.3.6	<code>param</code>	2824
5.604.3.7	<code>param</code>	2824
5.604.3.8	<code>reset</code>	2824
5.604.4	Friends And Related Function Documentation	2824
5.604.4.1	<code>operator&lt;&lt;</code>	2824
5.604.4.2	<code>operator&gt;&gt;</code>	2825
5.605	<code>std::piecewise_linear_distribution&lt;_RealType&gt;::param_type</code> Struct Reference	2825
5.605.1	Detailed Description	2826
5.606	<code>std::plus&lt;_Tp&gt;</code> Struct Template Reference	2826
5.606.1	Detailed Description	2827
5.606.2	Member Typedef Documentation	2828
5.606.2.1	<code>first_argument_type</code>	2828
5.606.2.2	<code>result_type</code>	2828
5.606.2.3	<code>second_argument_type</code>	2828
5.607	<code>std::pointer_to_binary_function&lt;_Arg1, _Arg2, _Result&gt;</code> Class Template Reference	2828
5.607.1	Detailed Description	2829
5.607.2	Member Typedef Documentation	2830
5.607.2.1	<code>first_argument_type</code>	2830
5.607.2.2	<code>result_type</code>	2830

---

5.607.2.3 second_argument_type . . . . .	2830
5.608std::pointer_to_unary_function< _Arg, _Result > Class Template Reference . . . . .	2830
5.608.1 Detailed Description . . . . .	2831
5.608.2 Member Typedef Documentation . . . . .	2832
5.608.2.1 argument_type . . . . .	2832
5.608.2.2 result_type . . . . .	2832
5.609std::poisson_distribution< _IntType > Class Template Reference . . . . .	2832
5.609.1 Detailed Description . . . . .	2833
5.609.2 Member Typedef Documentation . . . . .	2833
5.609.2.1 result_type . . . . .	2833
5.609.3 Member Function Documentation . . . . .	2834
5.609.3.1 max . . . . .	2834
5.609.3.2 mean . . . . .	2834
5.609.3.3 min . . . . .	2834
5.609.3.4 operator() . . . . .	2834
5.609.3.5 operator() . . . . .	2835
5.609.3.6 param . . . . .	2835
5.609.3.7 param . . . . .	2835
5.609.3.8 reset . . . . .	2835
5.609.4 Friends And Related Function Documentation . . . . .	2836
5.609.4.1 operator<< . . . . .	2836
5.609.4.2 operator== . . . . .	2836
5.609.4.3 operator>> . . . . .	2836
5.610std::poisson_distribution< _IntType >::param_type Struct Reference . . . . .	2837
5.610.1 Detailed Description . . . . .	2837
5.611std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference . . . . .	2837
5.611.1 Detailed Description . . . . .	2838
5.611.2 Constructor & Destructor Documentation . . . . .	2839
5.611.2.1 priority_queue . . . . .	2839
5.611.2.2 priority_queue . . . . .	2840

---

5.611.3 Member Function Documentation . . . . .	2840
5.611.3.1 empty . . . . .	2840
5.611.3.2 pop . . . . .	2840
5.611.3.3 push . . . . .	2841
5.611.3.4 size . . . . .	2841
5.611.3.5 top . . . . .	2841
5.612std::promise< _Res > Class Template Reference . . . . .	2842
5.612.1 Detailed Description . . . . .	2842
5.613std::promise< _Res & > Class Template Reference . . . . .	2843
5.613.1 Detailed Description . . . . .	2843
5.614std::promise< void > Class Template Reference . . . . .	2843
5.614.1 Detailed Description . . . . .	2844
5.615std::queue< _Tp, _Sequence > Class Template Reference . . . . .	2844
5.615.1 Detailed Description . . . . .	2845
5.615.2 Constructor & Destructor Documentation . . . . .	2846
5.615.2.1 queue . . . . .	2846
5.615.3 Member Function Documentation . . . . .	2846
5.615.3.1 back . . . . .	2846
5.615.3.2 back . . . . .	2846
5.615.3.3 empty . . . . .	2846
5.615.3.4 front . . . . .	2846
5.615.3.5 front . . . . .	2847
5.615.3.6 pop . . . . .	2847
5.615.3.7 push . . . . .	2847
5.615.3.8 size . . . . .	2847
5.615.4 Member Data Documentation . . . . .	2848
5.615.4.1 c . . . . .	2848
5.616std::random_access_iterator_tag Struct Reference . . . . .	2848
5.616.1 Detailed Description . . . . .	2849
5.617std::random_device Class Reference . . . . .	2849
5.617.1 Detailed Description . . . . .	2850

---

5.617.2 Member Typedef Documentation . . . . .	2850
5.617.2.1 result_type . . . . .	2850
5.618std::range_error Class Reference . . . . .	2851
5.618.1 Detailed Description . . . . .	2851
5.618.2 Member Function Documentation . . . . .	2851
5.618.2.1 what . . . . .	2851
5.619std::ratio< _Num, _Den > Struct Template Reference . . . . .	2852
5.619.1 Detailed Description . . . . .	2852
5.620std::ratio_add< _R1, _R2 > Struct Template Reference . . . . .	2853
5.620.1 Detailed Description . . . . .	2853
5.621std::ratio_divide< _R1, _R2 > Struct Template Reference . . . . .	2853
5.621.1 Detailed Description . . . . .	2854
5.622std::ratio_equal< _R1, _R2 > Struct Template Reference . . . . .	2854
5.622.1 Detailed Description . . . . .	2854
5.623std::ratio_greater< _R1, _R2 > Struct Template Reference . . . . .	2854
5.623.1 Detailed Description . . . . .	2854
5.624std::ratio_greater_equal< _R1, _R2 > Struct Template Reference . . . . .	2855
5.624.1 Detailed Description . . . . .	2855
5.625std::ratio_less< _R1, _R2 > Struct Template Reference . . . . .	2855
5.625.1 Detailed Description . . . . .	2855
5.626std::ratio_less_equal< _R1, _R2 > Struct Template Reference . . . . .	2856
5.626.1 Detailed Description . . . . .	2856
5.627std::ratio_multiply< _R1, _R2 > Struct Template Reference . . . . .	2856
5.627.1 Detailed Description . . . . .	2856
5.628std::ratio_not_equal< _R1, _R2 > Struct Template Reference . . . . .	2857
5.628.1 Detailed Description . . . . .	2857
5.629std::ratio_subtract< _R1, _R2 > Struct Template Reference . . . . .	2857
5.629.1 Detailed Description . . . . .	2857
5.630std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference . . . . .	2858
5.630.1 Detailed Description . . . . .	2859

---

---

5.630.2 Member Typedef Documentation . . . . .	2859
5.630.2.1 difference_type . . . . .	2859
5.630.2.2 iterator_category . . . . .	2859
5.630.2.3 pointer . . . . .	2859
5.630.2.4 reference . . . . .	2859
5.630.2.5 value_type . . . . .	2860
5.631 std::recursive_mutex Class Reference . . . . .	2860
5.631.1 Detailed Description . . . . .	2860
5.632 std::recursive_timed_mutex Class Reference . . . . .	2860
5.632.1 Detailed Description . . . . .	2861
5.633 std::reference_wrapper<_Tp> Class Template Reference . . . . .	2861
5.633.1 Detailed Description . . . . .	2863
5.634 std::remove_reference<_Tp> Struct Template Reference . . . . .	2863
5.634.1 Detailed Description . . . . .	2863
5.635 std::reverse_iterator<_Iterator> Class Template Reference . . . . .	2863
5.635.1 Detailed Description . . . . .	2865
5.635.2 Member Typedef Documentation . . . . .	2865
5.635.2.1 difference_type . . . . .	2865
5.635.2.2 iterator_category . . . . .	2865
5.635.2.3 pointer . . . . .	2866
5.635.2.4 reference . . . . .	2866
5.635.2.5 value_type . . . . .	2866
5.635.3 Constructor & Destructor Documentation . . . . .	2866
5.635.3.1 reverse_iterator . . . . .	2866
5.635.3.2 reverse_iterator . . . . .	2867
5.635.3.3 reverse_iterator . . . . .	2867
5.635.3.4 reverse_iterator . . . . .	2867
5.635.4 Member Function Documentation . . . . .	2867
5.635.4.1 base . . . . .	2867
5.635.4.2 operator* . . . . .	2867
5.635.4.3 operator+ . . . . .	2868

---

5.635.4.4 operator++	2868
5.635.4.5 operator++	2868
5.635.4.6 operator+=	2869
5.635.4.7 operator-	2869
5.635.4.8 operator--	2869
5.635.4.9 operator--	2870
5.635.4.10 operator-=	2870
5.635.4.11 operator->	2870
5.635.4.12 operator[]	2870
5.636 std::runtime_error Class Reference	2871
5.636.1 Detailed Description	2871
5.636.2 Constructor & Destructor Documentation	2872
5.636.2.1 runtime_error	2872
5.636.3 Member Function Documentation	2872
5.636.3.1 what	2872
5.637 std::seed_seq Class Reference	2872
5.637.1 Detailed Description	2873
5.637.2 Member Typedef Documentation	2873
5.637.2.1 result_type	2873
5.637.3 Constructor & Destructor Documentation	2873
5.637.3.1 seed_seq	2873
5.638 std::set<_Key, _Compare, _Alloc > Class Template Reference	2873
5.638.1 Detailed Description	2875
5.638.2 Member Typedef Documentation	2876
5.638.2.1 allocator_type	2876
5.638.2.2 const_iterator	2876
5.638.2.3 const_pointer	2876
5.638.2.4 const_reference	2876
5.638.2.5 const_reverse_iterator	2877
5.638.2.6 difference_type	2877
5.638.2.7 iterator	2877

---

5.638.2.8	key_compare	2877
5.638.2.9	key_type	2877
5.638.2.10	pointer	2878
5.638.2.11	reference	2878
5.638.2.12	reverse_iterator	2878
5.638.2.13	size_type	2878
5.638.2.14	value_compare	2878
5.638.2.15	value_type	2879
5.638.3	Constructor & Destructor Documentation	2879
5.638.3.1	set	2879
5.638.3.2	set	2879
5.638.3.3	set	2879
5.638.3.4	set	2880
5.638.3.5	set	2880
5.638.3.6	set	2881
5.638.3.7	set	2881
5.638.4	Member Function Documentation	2881
5.638.4.1	begin	2881
5.638.4.2	cbegin	2882
5.638.4.3	cend	2882
5.638.4.4	clear	2882
5.638.4.5	count	2882
5.638.4.6	crbegin	2883
5.638.4.7	crend	2883
5.638.4.8	empty	2883
5.638.4.9	end	2883
5.638.4.10	equal_range	2883
5.638.4.11	equal_range	2884
5.638.4.12	erase	2884
5.638.4.13	erase	2885
5.638.4.14	erase	2885

---

5.638.4.15	find	2886
5.638.4.16	find	2886
5.638.4.17	get_allocator	2887
5.638.4.18	insert	2887
5.638.4.19	insert	2887
5.638.4.20	insert	2888
5.638.4.21	insert	2888
5.638.4.22	key_comp	2889
5.638.4.23	lower_bound	2889
5.638.4.24	lower_bound	2889
5.638.4.25	max_size	2890
5.638.4.26	operator=	2890
5.638.4.27	operator=	2890
5.638.4.28	operator=	2891
5.638.4.29	begin	2891
5.638.4.30	end	2891
5.638.4.31	size	2891
5.638.4.32	swap	2892
5.638.4.33	upper_bound	2892
5.638.4.34	upper_bound	2892
5.638.4.35	value_comp	2893
5.639	std::shared_future< _Res > Class Template Reference	2893
5.639.1	Detailed Description	2894
5.639.2	Constructor & Destructor Documentation	2894
5.639.2.1	shared_future	2894
5.639.2.2	shared_future	2895
5.639.2.3	shared_future	2895
5.639.3	Member Function Documentation	2895
5.639.3.1	_M_get_result	2895
5.639.3.2	get	2895
5.640	std::shared_future< _Res & > Class Template Reference	2895

---

---

5.640.1 Detailed Description . . . . .	2897
5.640.2 Constructor & Destructor Documentation . . . . .	2897
5.640.2.1 shared_future . . . . .	2897
5.640.2.2 shared_future . . . . .	2897
5.640.2.3 shared_future . . . . .	2897
5.640.3 Member Function Documentation . . . . .	2898
5.640.3.1 _M_get_result . . . . .	2898
5.640.3.2 get . . . . .	2898
5.641std::shared_future< void > Class Template Reference . . . . .	2898
5.641.1 Detailed Description . . . . .	2900
5.641.2 Constructor & Destructor Documentation . . . . .	2900
5.641.2.1 shared_future . . . . .	2900
5.641.2.2 shared_future . . . . .	2900
5.641.2.3 shared_future . . . . .	2900
5.641.3 Member Function Documentation . . . . .	2901
5.641.3.1 _M_get_result . . . . .	2901
5.642std::shared_ptr< _Tp > Class Template Reference . . . . .	2901
5.642.1 Detailed Description . . . . .	2902
5.642.2 Constructor & Destructor Documentation . . . . .	2902
5.642.2.1 shared_ptr . . . . .	2902
5.642.2.2 shared_ptr . . . . .	2903
5.642.2.3 shared_ptr . . . . .	2903
5.642.2.4 shared_ptr . . . . .	2904
5.642.2.5 shared_ptr . . . . .	2904
5.642.2.6 shared_ptr . . . . .	2905
5.642.2.7 shared_ptr . . . . .	2905
5.642.2.8 shared_ptr . . . . .	2905
5.642.2.9 shared_ptr . . . . .	2906
5.642.3 Friends And Related Function Documentation . . . . .	2906
5.642.3.1 allocate_shared . . . . .	2906

---

---

5.643std::shuffle_order_engine< _RandomNumberEngine, __k > Class	
Template Reference . . . . .	2907
5.643.1 Detailed Description . . . . .	2908
5.643.2 Member Typedef Documentation . . . . .	2908
5.643.2.1 result_type . . . . .	2908
5.643.3 Constructor & Destructor Documentation . . . . .	2908
5.643.3.1 shuffle_order_engine . . . . .	2908
5.643.3.2 shuffle_order_engine . . . . .	2909
5.643.3.3 shuffle_order_engine . . . . .	2909
5.643.3.4 shuffle_order_engine . . . . .	2909
5.643.3.5 shuffle_order_engine . . . . .	2910
5.643.4 Member Function Documentation . . . . .	2910
5.643.4.1 base . . . . .	2910
5.643.4.2 discard . . . . .	2910
5.643.4.3 max . . . . .	2910
5.643.4.4 min . . . . .	2911
5.643.4.5 operator() . . . . .	2911
5.643.4.6 seed . . . . .	2911
5.643.4.7 seed . . . . .	2911
5.643.4.8 seed . . . . .	2912
5.643.5 Friends And Related Function Documentation . . . . .	2912
5.643.5.1 operator<< . . . . .	2912
5.643.5.2 operator== . . . . .	2912
5.643.5.3 operator>> . . . . .	2913
5.644std::slice Class Reference . . . . .	2913
5.644.1 Detailed Description . . . . .	2913
5.645std::slice_array< _Tp > Class Template Reference . . . . .	2914
5.645.1 Detailed Description . . . . .	2915
5.645.2 Member Function Documentation . . . . .	2916
5.645.2.1 operator%= . . . . .	2916
5.645.2.2 operator&= . . . . .	2916

5.645.2.3 operator*= 5.645.2.4 operator+= 5.645.2.5 operator-= 5.645.2.6 operator/=	2916 2916 2916 2916
5.645.2.7 operator<<= 5.645.2.8 operator>>= 5.645.2.9 operator^= 5.645.2.10operator =	2916 2916 2917 2917
5.646std::stack< _Tp, _Sequence > Class Template Reference	2917
5.646.1 Detailed Description	2918
5.646.2 Constructor & Destructor Documentation	2918
5.646.2.1 stack	2918
5.646.3 Member Function Documentation	2919
5.646.3.1 empty 5.646.3.2 pop 5.646.3.3 push 5.646.3.4 size 5.646.3.5 top 5.646.3.6 top	2919 2919 2919 2919 2920 2920
5.647std::student_t_distribution< _RealType > Class Template Reference	2920
5.647.1 Detailed Description	2921
5.647.2 Member Typedef Documentation	2921
5.647.2.1 result_type	2921
5.647.3 Member Function Documentation	2922
5.647.3.1 max 5.647.3.2 min 5.647.3.3 operator() 5.647.3.4 param 5.647.3.5 param 5.647.3.6 reset	2922 2922 2922 2922 2922 2923
5.647.4 Friends And Related Function Documentation	2923

---

5.647.4.1 operator<<	2923
5.647.4.2 operator==	2923
5.647.4.3 operator>>	2924
5.648std::student_t_distribution<_RealType>::param_type Struct Reference	2924
5.648.1 Detailed Description	2925
5.649std::system_error Class Reference	2925
5.649.1 Detailed Description	2926
5.649.2 Member Function Documentation	2926
5.649.2.1 what	2926
5.650std::thread Class Reference	2926
5.650.1 Detailed Description	2927
5.650.2 Member Function Documentation	2927
5.650.2.1 native_handle	2927
5.651std::thread::id Class Reference	2927
5.651.1 Detailed Description	2928
5.652std::time_base Class Reference	2928
5.652.1 Detailed Description	2929
5.653std::time_get<_CharT, _InIter> Class Template Reference	2930
5.653.1 Detailed Description	2932
5.653.2 Member Typedef Documentation	2932
5.653.2.1 char_type	2932
5.653.2.2 iter_type	2932
5.653.3 Constructor & Destructor Documentation	2933
5.653.3.1 time_get	2933
5.653.3.2 ~time_get	2933
5.653.4 Member Function Documentation	2933
5.653.4.1 date_order	2933
5.653.4.2 do_date_order	2934
5.653.4.3 do_get_date	2934
5.653.4.4 do_get_monthname	2935
5.653.4.5 do_get_time	2935

---

5.653.4.6 do_get_weekday . . . . .	2936
5.653.4.7 do_get_year . . . . .	2937
5.653.4.8 get_date . . . . .	2937
5.653.4.9 get_monthname . . . . .	2938
5.653.4.10get_time . . . . .	2939
5.653.4.1lget_weekday . . . . .	2939
5.653.4.12get_year . . . . .	2940
5.653.5 Member Data Documentation . . . . .	2941
5.653.5.1 id . . . . .	2941
5.654std::time_get_byname< _CharT, _InIter > Class Template Reference . . . . .	2941
5.654.1 Detailed Description . . . . .	2943
5.654.2 Member Typedef Documentation . . . . .	2943
5.654.2.1 char_type . . . . .	2943
5.654.2.2 iter_type . . . . .	2944
5.654.3 Member Function Documentation . . . . .	2944
5.654.3.1 date_order . . . . .	2944
5.654.3.2 do_date_order . . . . .	2944
5.654.3.3 do_get_date . . . . .	2945
5.654.3.4 do_get_monthname . . . . .	2945
5.654.3.5 do_get_time . . . . .	2946
5.654.3.6 do_get_weekday . . . . .	2947
5.654.3.7 do_get_year . . . . .	2947
5.654.3.8 get_date . . . . .	2948
5.654.3.9 get_monthname . . . . .	2949
5.654.3.10get_time . . . . .	2949
5.654.3.1lget_weekday . . . . .	2950
5.654.3.12get_year . . . . .	2951
5.654.4 Member Data Documentation . . . . .	2951
5.654.4.1 id . . . . .	2951
5.655std::time_put< _CharT, _OutIter > Class Template Reference . . . . .	2952
5.655.1 Detailed Description . . . . .	2953

---

5.655.2 Member Typedef Documentation . . . . .	2954
5.655.2.1 char_type . . . . .	2954
5.655.2.2 iter_type . . . . .	2954
5.655.3 Constructor & Destructor Documentation . . . . .	2954
5.655.3.1 time_put . . . . .	2954
5.655.3.2 ~time_put . . . . .	2954
5.655.4 Member Function Documentation . . . . .	2955
5.655.4.1 do_put . . . . .	2955
5.655.4.2 put . . . . .	2955
5.655.4.3 put . . . . .	2956
5.655.5 Member Data Documentation . . . . .	2957
5.655.5.1 id . . . . .	2957
5.656std::time_put_byname< _CharT, _OutIter > Class Template Reference	2957
5.656.1 Detailed Description . . . . .	2958
5.656.2 Member Typedef Documentation . . . . .	2959
5.656.2.1 char_type . . . . .	2959
5.656.2.2 iter_type . . . . .	2959
5.656.3 Member Function Documentation . . . . .	2959
5.656.3.1 do_put . . . . .	2959
5.656.3.2 put . . . . .	2960
5.656.3.3 put . . . . .	2960
5.656.4 Member Data Documentation . . . . .	2961
5.656.4.1 id . . . . .	2961
5.657std::timed_mutex Class Reference . . . . .	2961
5.657.1 Detailed Description . . . . .	2962
5.658std::tr1::__is_member_pointer_helper< _Tp > Struct Template Refer- ence . . . . .	2962
5.658.1 Detailed Description . . . . .	2963
5.659std::tr1::add_const< _Tp > Struct Template Reference . . . . .	2963
5.659.1 Detailed Description . . . . .	2964
5.660std::tr1::add_cv< _Tp > Struct Template Reference . . . . .	2964

---

5.660.1 Detailed Description . . . . .	2964
5.661std::tr1::add_pointer< _Tp > Struct Template Reference . . . . .	2964
5.661.1 Detailed Description . . . . .	2965
5.662std::tr1::add_volatile< _Tp > Struct Template Reference . . . . .	2965
5.662.1 Detailed Description . . . . .	2965
5.663std::tr1::alignment_of< _Tp > Struct Template Reference . . . . .	2965
5.663.1 Detailed Description . . . . .	2966
5.664std::tr1::array< _Tp, _Nm > Struct Template Reference . . . . .	2967
5.664.1 Detailed Description . . . . .	2968
5.665std::tr1::bad_weak_ptr Class Reference . . . . .	2968
5.665.1 Detailed Description . . . . .	2969
5.665.2 Member Function Documentation . . . . .	2969
5.665.2.1 what . . . . .	2969
5.666std::tr1::extent< typename, _UInt > Struct Template Reference . . . . .	2970
5.666.1 Detailed Description . . . . .	2970
5.667std::tr1::has_virtual_destructor< _Tp > Struct Template Reference . . . . .	2971
5.667.1 Detailed Description . . . . .	2972
5.668std::tr1::integral_constant< _Tp, __v > Struct Template Reference . . . . .	2972
5.668.1 Detailed Description . . . . .	2974
5.669std::tr1::is_abstract< _Tp > Struct Template Reference . . . . .	2974
5.669.1 Detailed Description . . . . .	2975
5.670std::tr1::is_arithmetic< _Tp > Struct Template Reference . . . . .	2975
5.670.1 Detailed Description . . . . .	2976
5.671std::tr1::is_array< typename > Struct Template Reference . . . . .	2976
5.671.1 Detailed Description . . . . .	2977
5.672std::tr1::is_class< _Tp > Struct Template Reference . . . . .	2977
5.672.1 Detailed Description . . . . .	2978
5.673std::tr1::is_compound< _Tp > Struct Template Reference . . . . .	2978
5.673.1 Detailed Description . . . . .	2979
5.674std::tr1::is_const< typename > Struct Template Reference . . . . .	2979
5.674.1 Detailed Description . . . . .	2980

---

5.675	<code>std::tr1::is_empty&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2980
5.675.1	Detailed Description . . . . .	2981
5.676	<code>std::tr1::is_enum&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2982
5.676.1	Detailed Description . . . . .	2982
5.677	<code>std::tr1::is_floating_point&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2983
5.677.1	Detailed Description . . . . .	2983
5.678	<code>std::tr1::is_function&lt; typename &gt;</code> Struct Template Reference . . . . .	2984
5.678.1	Detailed Description . . . . .	2984
5.679	<code>std::tr1::is_fundamental&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2985
5.679.1	Detailed Description . . . . .	2985
5.680	<code>std::tr1::is_integral&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2985
5.680.1	Detailed Description . . . . .	2986
5.681	<code>std::tr1::is_member_function_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2986
5.681.1	Detailed Description . . . . .	2987
5.682	<code>std::tr1::is_member_object_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2987
5.682.1	Detailed Description . . . . .	2988
5.683	<code>std::tr1::is_object&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2988
5.683.1	Detailed Description . . . . .	2988
5.684	<code>std::tr1::is_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2989
5.684.1	Detailed Description . . . . .	2989
5.685	<code>std::tr1::is_polymorphic&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2989
5.685.1	Detailed Description . . . . .	2990
5.686	<code>std::tr1::is_same&lt; typename, typename &gt;</code> Struct Template Reference . . . . .	2991
5.686.1	Detailed Description . . . . .	2991
5.687	<code>std::tr1::is_scalar&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2992
5.687.1	Detailed Description . . . . .	2992
5.688	<code>std::tr1::is_union&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2992
5.688.1	Detailed Description . . . . .	2993
5.689	<code>std::tr1::is_void&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2994
5.689.1	Detailed Description . . . . .	2994

---

5.690	<code>std::tr1::is_volatile&lt; typename &gt;</code> Struct Template Reference . . . . .	2994
5.690.1	Detailed Description . . . . .	2995
5.691	<code>std::tr1::rank&lt; typename &gt;</code> Struct Template Reference . . . . .	2995
5.691.1	Detailed Description . . . . .	2996
5.692	<code>std::tr1::remove_all_extents&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2997
5.692.1	Detailed Description . . . . .	2997
5.693	<code>std::tr1::remove_const&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2997
5.693.1	Detailed Description . . . . .	2997
5.694	<code>std::tr1::remove_cv&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2998
5.694.1	Detailed Description . . . . .	2998
5.695	<code>std::tr1::remove_extent&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2998
5.695.1	Detailed Description . . . . .	2998
5.696	<code>std::tr1::remove_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2999
5.696.1	Detailed Description . . . . .	2999
5.697	<code>std::tr1::remove_volatile&lt; _Tp &gt;</code> Struct Template Reference . . . . .	2999
5.697.1	Detailed Description . . . . .	2999
5.698	<code>std::try_to_lock_t</code> Struct Reference . . . . .	3000
5.698.1	Detailed Description . . . . .	3000
5.699	<code>std::tuple&lt; _Elements &gt;</code> Class Template Reference . . . . .	3000
5.699.1	Detailed Description . . . . .	3001
5.700	<code>std::tuple&lt; _T1, _T2 &gt;</code> Class Template Reference . . . . .	3001
5.700.1	Detailed Description . . . . .	3002
5.701	<code>std::tuple_element&lt; 0, tuple&lt; _Head, _Tail...&gt; &gt;</code> Struct Template Reference . . . . .	3002
5.701.1	Detailed Description . . . . .	3002
5.702	<code>std::tuple_element&lt; __i, tuple&lt; _Head, _Tail...&gt; &gt;</code> Struct Template Reference . . . . .	3003
5.702.1	Detailed Description . . . . .	3003
5.703	<code>std::tuple_size&lt; tuple&lt; _Elements...&gt; &gt;</code> Struct Template Reference . . . . .	3003
5.703.1	Detailed Description . . . . .	3003
5.704	<code>std::type_info</code> Class Reference . . . . .	3004
5.704.1	Detailed Description . . . . .	3004

---

5.704.2 Constructor & Destructor Documentation . . . . .	3005
5.704.2.1 ~type_info . . . . .	3005
5.704.3 Member Function Documentation . . . . .	3005
5.704.3.1 name . . . . .	3005
5.705std::unary_function< _Arg, _Result > Struct Template Reference . . . . .	3007
5.705.1 Detailed Description . . . . .	3008
5.705.2 Member Typedef Documentation . . . . .	3008
5.705.2.1 argument_type . . . . .	3008
5.705.2.2 result_type . . . . .	3008
5.706std::unary_negate< _Predicate > Class Template Reference . . . . .	3008
5.706.1 Detailed Description . . . . .	3009
5.706.2 Member Typedef Documentation . . . . .	3010
5.706.2.1 argument_type . . . . .	3010
5.706.2.2 result_type . . . . .	3010
5.707std::underflow_error Class Reference . . . . .	3011
5.707.1 Detailed Description . . . . .	3011
5.707.2 Member Function Documentation . . . . .	3011
5.707.2.1 what . . . . .	3011
5.708std::uniform_int_distribution< _IntType > Class Template Reference . . . . .	3012
5.708.1 Detailed Description . . . . .	3013
5.708.2 Member Typedef Documentation . . . . .	3013
5.708.2.1 result_type . . . . .	3013
5.708.3 Constructor & Destructor Documentation . . . . .	3013
5.708.3.1 uniform_int_distribution . . . . .	3013
5.708.4 Member Function Documentation . . . . .	3013
5.708.4.1 max . . . . .	3013
5.708.4.2 min . . . . .	3014
5.708.4.3 operator() . . . . .	3014
5.708.4.4 param . . . . .	3014
5.708.4.5 param . . . . .	3014
5.708.4.6 reset . . . . .	3015

5.709	<a href="#">std::uniform_int_distribution&lt; _IntType &gt;::param_type Struct Reference</a>	3015
	5.709.1 Detailed Description	3015
5.710	<a href="#">std::uniform_real_distribution&lt; _RealType &gt; Class Template Reference</a>	3016
	5.710.1 Detailed Description	3016
	5.710.2 Member Typedef Documentation	3017
	5.710.2.1 result_type	3017
	5.710.3 Constructor & Destructor Documentation	3017
	5.710.3.1 uniform_real_distribution	3017
	5.710.4 Member Function Documentation	3017
	5.710.4.1 max	3017
	5.710.4.2 min	3017
	5.710.4.3 operator()	3018
	5.710.4.4 param	3018
	5.710.4.5 param	3018
	5.710.4.6 reset	3018
5.711	<a href="#">std::uniform_real_distribution&lt; _RealType &gt;::param_type Struct Reference</a>	3019
	5.711.1 Detailed Description	3019
5.712	<a href="#">std::unique_lock&lt; _Mutex &gt; Class Template Reference</a>	3019
	5.712.1 Detailed Description	3020
5.713	<a href="#">std::unique_ptr&lt; _Tp, _Tp_Deleter &gt; Class Template Reference</a>	3021
	5.713.1 Detailed Description	3022
5.714	<a href="#">std::unique_ptr&lt; _Tp[], _Tp_Deleter &gt; Class Template Reference</a>	3022
	5.714.1 Detailed Description	3023
5.715	<a href="#">std::unordered_map&lt; _Key, _Tp, _Hash, _Pred, _Alloc &gt; Class Template Reference</a>	3023
	5.715.1 Detailed Description	3026
5.716	<a href="#">std::unordered_multimap&lt; _Key, _Tp, _Hash, _Pred, _Alloc &gt; Class Template Reference</a>	3026
	5.716.1 Detailed Description	3028
5.717	<a href="#">std::unordered_multiset&lt; _Value, _Hash, _Pred, _Alloc &gt; Class Template Reference</a>	3029

---

5.717.1 Detailed Description . . . . .	3031
5.718std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3032
5.718.1 Detailed Description . . . . .	3034
5.719std::valarray< _Tp > Class Template Reference . . . . .	3034
5.719.1 Detailed Description . . . . .	3037
5.719.2 Constructor & Destructor Documentation . . . . .	3038
5.719.2.1 valarray . . . . .	3038
5.720std::vector< _Tp, _Alloc > Class Template Reference . . . . .	3038
5.720.1 Detailed Description . . . . .	3042
5.720.2 Constructor & Destructor Documentation . . . . .	3042
5.720.2.1 vector . . . . .	3042
5.720.2.2 vector . . . . .	3042
5.720.2.3 vector . . . . .	3043
5.720.2.4 vector . . . . .	3043
5.720.2.5 vector . . . . .	3043
5.720.2.6 vector . . . . .	3044
5.720.2.7 vector . . . . .	3044
5.720.2.8 ~vector . . . . .	3044
5.720.3 Member Function Documentation . . . . .	3045
5.720.3.1 _M_allocate_and_copy . . . . .	3045
5.720.3.2 _M_range_check . . . . .	3045
5.720.3.3 assign . . . . .	3045
5.720.3.4 assign . . . . .	3046
5.720.3.5 assign . . . . .	3046
5.720.3.6 at . . . . .	3046
5.720.3.7 at . . . . .	3047
5.720.3.8 back . . . . .	3047
5.720.3.9 back . . . . .	3047
5.720.3.10begin . . . . .	3048
5.720.3.11begin . . . . .	3048

---

5.720.3.12	capacity	3048
5.720.3.13	begin	3048
5.720.3.14	end	3049
5.720.3.15	clear	3049
5.720.3.16	crbegin	3049
5.720.3.17	crend	3049
5.720.3.18	data	3049
5.720.3.19	emplace	3050
5.720.3.20	empty	3050
5.720.3.21	end	3050
5.720.3.22	end	3050
5.720.3.23	erase	3051
5.720.3.24	erase	3051
5.720.3.25	front	3052
5.720.3.26	front	3052
5.720.3.27	insert	3052
5.720.3.28	insert	3053
5.720.3.29	insert	3053
5.720.3.30	insert	3054
5.720.3.31	insert	3054
5.720.3.32	max_size	3054
5.720.3.33	operator=	3055
5.720.3.34	operator=	3055
5.720.3.35	operator=	3055
5.720.3.36	operator[]	3056
5.720.3.37	operator[]	3056
5.720.3.38	pop_back	3056
5.720.3.39	push_back	3057
5.720.3.40	begin	3057
5.720.3.41	lbegin	3057
5.720.3.42	rend	3058

5.720.3.43	end	3058
5.720.3.44	reserve	3058
5.720.3.45	resize	3058
5.720.3.46	shrink_to_fit	3059
5.720.3.47	size	3059
5.720.3.48	swap	3059
5.721	std::vector< bool, _Alloc > Class Template Reference	3060
5.721.1	Detailed Description	3063
5.722	std::weak_ptr< _Tp > Class Template Reference	3064
5.722.1	Detailed Description	3064
5.723	std::weibull_distribution< _RealType > Class Template Reference	3065
5.723.1	Detailed Description	3065
5.723.2	Member Typedef Documentation	3066
5.723.2.1	result_type	3066
5.723.3	Member Function Documentation	3066
5.723.3.1	a	3066
5.723.3.2	b	3066
5.723.3.3	max	3066
5.723.3.4	min	3066
5.723.3.5	operator()	3066
5.723.3.6	param	3067
5.723.3.7	param	3067
5.723.3.8	reset	3067
5.724	std::weibull_distribution< _RealType >::param_type Struct Reference	3068
5.724.1	Detailed Description	3068
<b>6</b>	<b>File Documentation</b>	<b>3069</b>
6.1	algo.h File Reference	3069
6.1.1	Detailed Description	3083
6.2	algbase.h File Reference	3083
6.2.1	Detailed Description	3085

---

6.3	algorithm File Reference . . . . .	3085
6.3.1	Detailed Description . . . . .	3085
6.4	algorithm File Reference . . . . .	3085
6.4.1	Detailed Description . . . . .	3087
6.5	algorithm File Reference . . . . .	3087
6.5.1	Detailed Description . . . . .	3087
6.6	algorithmfwd.h File Reference . . . . .	3087
6.6.1	Detailed Description . . . . .	3094
6.7	algorithmfwd.h File Reference . . . . .	3094
6.7.1	Detailed Description . . . . .	3106
6.8	allocator.h File Reference . . . . .	3107
6.8.1	Detailed Description . . . . .	3107
6.9	array File Reference . . . . .	3107
6.9.1	Detailed Description . . . . .	3107
6.10	array File Reference . . . . .	3108
6.10.1	Detailed Description . . . . .	3109
6.11	array_allocator.h File Reference . . . . .	3109
6.11.1	Detailed Description . . . . .	3109
6.12	assoc_container.hpp File Reference . . . . .	3110
6.12.1	Detailed Description . . . . .	3111
6.13	atomic File Reference . . . . .	3111
6.13.1	Detailed Description . . . . .	3115
6.14	atomic_0.h File Reference . . . . .	3115
6.14.1	Detailed Description . . . . .	3116
6.15	atomic_2.h File Reference . . . . .	3116
6.15.1	Detailed Description . . . . .	3116
6.16	atomic_base.h File Reference . . . . .	3116
6.16.1	Detailed Description . . . . .	3118
6.17	atomic_word.h File Reference . . . . .	3118
6.17.1	Detailed Description . . . . .	3118
6.18	atomicfwd_c.h File Reference . . . . .	3118

---

6.18.1 Detailed Description . . . . .	3119
6.19 atomicfwd_cxx.h File Reference . . . . .	3120
6.19.1 Detailed Description . . . . .	3120
6.20 atomicity.h File Reference . . . . .	3120
6.20.1 Detailed Description . . . . .	3121
6.21 auto_ptr.h File Reference . . . . .	3121
6.21.1 Detailed Description . . . . .	3122
6.22 balanced_quicksort.h File Reference . . . . .	3122
6.22.1 Detailed Description . . . . .	3123
6.23 base.h File Reference . . . . .	3123
6.23.1 Detailed Description . . . . .	3125
6.24 base.h File Reference . . . . .	3125
6.24.1 Detailed Description . . . . .	3125
6.25 basic_file.h File Reference . . . . .	3125
6.25.1 Detailed Description . . . . .	3125
6.26 basic_ios.h File Reference . . . . .	3126
6.26.1 Detailed Description . . . . .	3126
6.27 basic_ios.tcc File Reference . . . . .	3126
6.27.1 Detailed Description . . . . .	3126
6.28 basic_iterator.h File Reference . . . . .	3127
6.28.1 Detailed Description . . . . .	3127
6.29 basic_string.h File Reference . . . . .	3127
6.29.1 Detailed Description . . . . .	3130
6.30 basic_string.tcc File Reference . . . . .	3131
6.30.1 Detailed Description . . . . .	3131
6.31 basic_types.hpp File Reference . . . . .	3131
6.31.1 Detailed Description . . . . .	3132
6.32 binders.h File Reference . . . . .	3132
6.32.1 Detailed Description . . . . .	3133
6.33 bitmap_allocator.h File Reference . . . . .	3133
6.33.1 Detailed Description . . . . .	3134

---

6.33.2	Define Documentation	3134
6.33.2.1	<code>_BALLOC_ALIGN_BYTES</code>	3134
6.34	<code>bitset</code> File Reference	3134
6.34.1	Detailed Description	3135
6.35	<code>bitset</code> File Reference	3136
6.35.1	Detailed Description	3136
6.36	<code>bitset</code> File Reference	3137
6.36.1	Detailed Description	3137
6.37	<code>boost_concept_check.h</code> File Reference	3138
6.37.1	Detailed Description	3138
6.38	<code>boost_sp_counted_base.h</code> File Reference	3139
6.38.1	Detailed Description	3139
6.39	<code>c++0x_warning.h</code> File Reference	3139
6.39.1	Detailed Description	3139
6.40	<code>c++allocator.h</code> File Reference	3139
6.40.1	Detailed Description	3140
6.41	<code>c++config.h</code> File Reference	3140
6.41.1	Detailed Description	3144
6.42	<code>c++io.h</code> File Reference	3145
6.42.1	Detailed Description	3145
6.43	<code>c++locale.h</code> File Reference	3145
6.43.1	Detailed Description	3146
6.44	<code>c++locale_internal.h</code> File Reference	3146
6.44.1	Detailed Description	3146
6.45	<code>cassert</code> File Reference	3146
6.45.1	Detailed Description	3146
6.46	<code>ccomplex</code> File Reference	3146
6.46.1	Detailed Description	3146
6.47	<code>ccomplex</code> File Reference	3147
6.47.1	Detailed Description	3147
6.48	<code>cctype</code> File Reference	3147

---

6.48.1 Detailed Description . . . . .	3147
6.49 ctype File Reference . . . . .	3147
6.49.1 Detailed Description . . . . .	3147
6.50 ctype File Reference . . . . .	3148
6.50.1 Detailed Description . . . . .	3148
6.51 cerrno File Reference . . . . .	3148
6.51.1 Detailed Description . . . . .	3148
6.52 cenv File Reference . . . . .	3148
6.52.1 Detailed Description . . . . .	3148
6.53 cenv File Reference . . . . .	3148
6.53.1 Detailed Description . . . . .	3149
6.54 cenv File Reference . . . . .	3149
6.54.1 Detailed Description . . . . .	3149
6.55 cfloat File Reference . . . . .	3149
6.55.1 Detailed Description . . . . .	3149
6.56 cfloat File Reference . . . . .	3149
6.56.1 Detailed Description . . . . .	3150
6.57 char_traits.h File Reference . . . . .	3150
6.57.1 Detailed Description . . . . .	3150
6.58 checkers.h File Reference . . . . .	3151
6.58.1 Detailed Description . . . . .	3151
6.59 chrono File Reference . . . . .	3151
6.59.1 Detailed Description . . . . .	3154
6.60 cinttypes File Reference . . . . .	3154
6.60.1 Detailed Description . . . . .	3154
6.61 cinttypes File Reference . . . . .	3154
6.61.1 Detailed Description . . . . .	3155
6.62 cinttypes File Reference . . . . .	3155
6.62.1 Detailed Description . . . . .	3155
6.63 ciso646 File Reference . . . . .	3155
6.63.1 Detailed Description . . . . .	3155

---

6.64	climits File Reference . . . . .	3155
6.64.1	Detailed Description . . . . .	3156
6.65	climits File Reference . . . . .	3156
6.65.1	Detailed Description . . . . .	3156
6.66	locale File Reference . . . . .	3156
6.66.1	Detailed Description . . . . .	3156
6.67	cmath File Reference . . . . .	3157
6.67.1	Detailed Description . . . . .	3159
6.68	cmath File Reference . . . . .	3160
6.68.1	Detailed Description . . . . .	3163
6.69	cmath File Reference . . . . .	3163
6.69.1	Detailed Description . . . . .	3163
6.70	cmath.tcc File Reference . . . . .	3164
6.70.1	Detailed Description . . . . .	3164
6.71	codecv.h File Reference . . . . .	3164
6.71.1	Detailed Description . . . . .	3165
6.72	codecv_specializations.h File Reference . . . . .	3165
6.72.1	Detailed Description . . . . .	3165
6.73	compatibility.h File Reference . . . . .	3166
6.73.1	Detailed Description . . . . .	3166
6.74	compatibility.h File Reference . . . . .	3166
6.74.1	Detailed Description . . . . .	3166
6.75	compiletime_settings.h File Reference . . . . .	3167
6.75.1	Detailed Description . . . . .	3167
6.75.2	Define Documentation . . . . .	3167
6.75.2.1	_GLIBCXX_ASSERTIONS . . . . .	3167
6.75.2.2	_GLIBCXX_CALL . . . . .	3167
6.75.2.3	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_L1 . . . . .	3168
6.75.2.4	_GLIBCXX_RANDOM_SHUFFLE_- CONSIDER_TLB . . . . .	3168
6.75.2.5	_GLIBCXX_SCALE_DOWN_FPU . . . . .	3168

---

6.75.2.6	<a href="#">_GLIBCXX_VERBOSE_LEVEL</a>	3168
6.76	<a href="#">complex</a> File Reference	3169
6.76.1	<a href="#">Detailed Description</a>	3172
6.77	<a href="#">complex</a> File Reference	3172
6.77.1	<a href="#">Detailed Description</a>	3173
6.78	<a href="#">complex</a> File Reference	3173
6.78.1	<a href="#">Detailed Description</a>	3175
6.79	<a href="#">complex.h</a> File Reference	3175
6.79.1	<a href="#">Detailed Description</a>	3175
6.80	<a href="#">concept_check.h</a> File Reference	3175
6.80.1	<a href="#">Detailed Description</a>	3175
6.81	<a href="#">concurrency.h</a> File Reference	3175
6.81.1	<a href="#">Detailed Description</a>	3176
6.82	<a href="#">cond_dealtor.hpp</a> File Reference	3176
6.82.1	<a href="#">Detailed Description</a>	3176
6.83	<a href="#">condition_variable</a> File Reference	3177
6.83.1	<a href="#">Detailed Description</a>	3177
6.84	<a href="#">constructors_destructor_fn_imps.hpp</a> File Reference	3177
6.84.1	<a href="#">Detailed Description</a>	3178
6.85	<a href="#">container_base_dispatch.hpp</a> File Reference	3178
6.85.1	<a href="#">Detailed Description</a>	3178
6.86	<a href="#">cpp_type_traits.h</a> File Reference	3178
6.86.1	<a href="#">Detailed Description</a>	3179
6.87	<a href="#">cpu_defines.h</a> File Reference	3179
6.87.1	<a href="#">Detailed Description</a>	3179
6.88	<a href="#">csetjmp</a> File Reference	3179
6.88.1	<a href="#">Detailed Description</a>	3179
6.89	<a href="#">csignal</a> File Reference	3180
6.89.1	<a href="#">Detailed Description</a>	3180
6.90	<a href="#">cstdarg</a> File Reference	3180
6.90.1	<a href="#">Detailed Description</a>	3180

---

6.91	cstdarg File Reference . . . . .	3181
6.91.1	Detailed Description . . . . .	3181
6.92	cstdbool File Reference . . . . .	3181
6.92.1	Detailed Description . . . . .	3181
6.93	cstdbool File Reference . . . . .	3181
6.93.1	Detailed Description . . . . .	3181
6.94	cstddef File Reference . . . . .	3181
6.94.1	Detailed Description . . . . .	3181
6.95	cstdint File Reference . . . . .	3182
6.95.1	Detailed Description . . . . .	3182
6.96	cstdint File Reference . . . . .	3182
6.96.1	Detailed Description . . . . .	3182
6.97	cstdint File Reference . . . . .	3182
6.97.1	Detailed Description . . . . .	3182
6.98	cstdio File Reference . . . . .	3183
6.98.1	Detailed Description . . . . .	3183
6.99	cstdio File Reference . . . . .	3183
6.99.1	Detailed Description . . . . .	3183
6.100	cstdio File Reference . . . . .	3183
6.100.1	Detailed Description . . . . .	3184
6.101	cstdlib File Reference . . . . .	3184
6.101.1	Detailed Description . . . . .	3184
6.102	cstdlib File Reference . . . . .	3184
6.102.1	Detailed Description . . . . .	3185
6.103	cstdlib File Reference . . . . .	3185
6.103.1	Detailed Description . . . . .	3185
6.104	cstring File Reference . . . . .	3185
6.104.1	Detailed Description . . . . .	3185
6.105	ctgmath File Reference . . . . .	3186
6.105.1	Detailed Description . . . . .	3186
6.106	ctgmath File Reference . . . . .	3186

---

6.106.1 Detailed Description . . . . .	3186
6.107ctime File Reference . . . . .	3186
6.107.1 Detailed Description . . . . .	3186
6.108ctime File Reference . . . . .	3186
6.108.1 Detailed Description . . . . .	3186
6.109ctype_base.h File Reference . . . . .	3187
6.109.1 Detailed Description . . . . .	3187
6.110ctype_inline.h File Reference . . . . .	3187
6.110.1 Detailed Description . . . . .	3187
6.111ctype_noninline.h File Reference . . . . .	3187
6.111.1 Detailed Description . . . . .	3187
6.112cwchar File Reference . . . . .	3188
6.112.1 Detailed Description . . . . .	3188
6.113cwchar File Reference . . . . .	3188
6.113.1 Detailed Description . . . . .	3188
6.114cwchar File Reference . . . . .	3189
6.114.1 Detailed Description . . . . .	3189
6.115cwctype File Reference . . . . .	3189
6.115.1 Detailed Description . . . . .	3189
6.116cwctype File Reference . . . . .	3189
6.116.1 Detailed Description . . . . .	3190
6.117cwctype File Reference . . . . .	3190
6.117.1 Detailed Description . . . . .	3190
6.118cxxabi-forced.h File Reference . . . . .	3190
6.118.1 Detailed Description . . . . .	3190
6.119cxxabi.h File Reference . . . . .	3190
6.119.1 Detailed Description . . . . .	3192
6.120cxxabi_tweaks.h File Reference . . . . .	3192
6.120.1 Detailed Description . . . . .	3193
6.121debug.h File Reference . . . . .	3193
6.121.1 Detailed Description . . . . .	3194

---

6.122	debug_allocator.h File Reference	3194
6.122.1	Detailed Description	3194
6.123	debug_map_base.hpp File Reference	3194
6.123.1	Detailed Description	3194
6.124	decimal File Reference	3194
6.124.1	Detailed Description	3206
6.125	deque File Reference	3206
6.125.1	Detailed Description	3206
6.126	deque File Reference	3207
6.126.1	Detailed Description	3207
6.127	deque File Reference	3208
6.127.1	Detailed Description	3208
6.128	deque.tcc File Reference	3209
6.128.1	Detailed Description	3209
6.129	enc_filebuf.h File Reference	3209
6.129.1	Detailed Description	3210
6.130	equally_split.h File Reference	3210
6.130.1	Detailed Description	3210
6.131	error_constants.h File Reference	3210
6.131.1	Detailed Description	3211
6.132	exception File Reference	3212
6.132.1	Detailed Description	3212
6.133	exception.hpp File Reference	3212
6.133.1	Detailed Description	3213
6.134	exception_ptr.h File Reference	3213
6.134.1	Detailed Description	3213
6.135	extptr_allocator.h File Reference	3214
6.135.1	Detailed Description	3214
6.136	features.h File Reference	3214
6.136.1	Detailed Description	3215
6.136.2	Define Documentation	3215

---

6.136.2.1	<a href="#">_GLIBCXX_BAL_QUICKSORT</a>	3215
6.136.2.2	<a href="#">_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS</a>	3215
6.136.2.3	<a href="#">_GLIBCXX_FIND_EQUAL_SPLIT</a>	3215
6.136.2.4	<a href="#">_GLIBCXX_FIND_GROWING_BLOCKS</a>	3216
6.136.2.5	<a href="#">_GLIBCXX_MERGESORT</a>	3216
6.136.2.6	<a href="#">_GLIBCXX_QUICKSORT</a>	3216
6.136.2.7	<a href="#">_GLIBCXX_TREE_DYNAMIC_BALANCING</a>	3216
6.136.2.8	<a href="#">_GLIBCXX_TREE_FULL_COPY</a>	3217
6.136.2.9	<a href="#">_GLIBCXX_TREE_INITIAL_SPLITTING</a>	3217
6.137	<a href="#">fenv.h File Reference</a>	3217
6.137.1	<a href="#">Detailed Description</a>	3217
6.138	<a href="#">find.h File Reference</a>	3217
6.138.1	<a href="#">Detailed Description</a>	3218
6.139	<a href="#">find_selectors.h File Reference</a>	3218
6.139.1	<a href="#">Detailed Description</a>	3219
6.140	<a href="#">for_each.h File Reference</a>	3219
6.140.1	<a href="#">Detailed Description</a>	3219
6.141	<a href="#">for_each_selectors.h File Reference</a>	3220
6.141.1	<a href="#">Detailed Description</a>	3221
6.142	<a href="#">formatter.h File Reference</a>	3222
6.142.1	<a href="#">Detailed Description</a>	3222
6.143	<a href="#">forward_list.h File Reference</a>	3223
6.143.1	<a href="#">Detailed Description</a>	3224
6.144	<a href="#">forward_list.tcc File Reference</a>	3224
6.144.1	<a href="#">Detailed Description</a>	3224
6.145	<a href="#">fstream File Reference</a>	3225
6.145.1	<a href="#">Detailed Description</a>	3225
6.146	<a href="#">fstream.tcc File Reference</a>	3226
6.146.1	<a href="#">Detailed Description</a>	3226
6.147	<a href="#">functexcept.h File Reference</a>	3226
6.147.1	<a href="#">Detailed Description</a>	3227

---

6.148functional File Reference . . . . .	3227
6.148.1 Detailed Description . . . . .	3231
6.149functional File Reference . . . . .	3231
6.149.1 Detailed Description . . . . .	3232
6.150functional_hash.h File Reference . . . . .	3232
6.150.1 Detailed Description . . . . .	3233
6.151functions.h File Reference . . . . .	3233
6.151.1 Detailed Description . . . . .	3235
6.152future File Reference . . . . .	3235
6.152.1 Detailed Description . . . . .	3238
6.153gslice.h File Reference . . . . .	3238
6.153.1 Detailed Description . . . . .	3238
6.154gslice_array.h File Reference . . . . .	3238
6.154.1 Detailed Description . . . . .	3239
6.155hash_fun.h File Reference . . . . .	3239
6.155.1 Detailed Description . . . . .	3239
6.156hash_map File Reference . . . . .	3239
6.156.1 Detailed Description . . . . .	3240
6.157hash_policy.hpp File Reference . . . . .	3240
6.157.1 Detailed Description . . . . .	3241
6.158hash_set File Reference . . . . .	3241
6.158.1 Detailed Description . . . . .	3242
6.159hashtable.h File Reference . . . . .	3242
6.159.1 Detailed Description . . . . .	3243
6.160hashtable.h File Reference . . . . .	3243
6.160.1 Detailed Description . . . . .	3243
6.161hashtable_policy.h File Reference . . . . .	3244
6.161.1 Detailed Description . . . . .	3244
6.162indirect_array.h File Reference . . . . .	3245
6.162.1 Detailed Description . . . . .	3245
6.163initializer_list File Reference . . . . .	3245

---

6.163.1 Detailed Description . . . . .	3245
6.164iomnip File Reference . . . . .	3246
6.164.1 Detailed Description . . . . .	3247
6.165ios File Reference . . . . .	3247
6.165.1 Detailed Description . . . . .	3247
6.166ios_base.h File Reference . . . . .	3247
6.166.1 Detailed Description . . . . .	3250
6.167iosfwd File Reference . . . . .	3250
6.167.1 Detailed Description . . . . .	3251
6.168iostream File Reference . . . . .	3251
6.168.1 Detailed Description . . . . .	3251
6.169istream File Reference . . . . .	3252
6.169.1 Detailed Description . . . . .	3253
6.170istream.tcc File Reference . . . . .	3253
6.170.1 Detailed Description . . . . .	3253
6.171iterator File Reference . . . . .	3254
6.171.1 Detailed Description . . . . .	3254
6.172iterator File Reference . . . . .	3254
6.172.1 Detailed Description . . . . .	3254
6.173iterator.h File Reference . . . . .	3254
6.173.1 Detailed Description . . . . .	3255
6.174iterator_tracker.h File Reference . . . . .	3255
6.174.1 Detailed Description . . . . .	3257
6.175limits File Reference . . . . .	3257
6.175.1 Detailed Description . . . . .	3259
6.176list File Reference . . . . .	3259
6.176.1 Detailed Description . . . . .	3259
6.177list File Reference . . . . .	3260
6.177.1 Detailed Description . . . . .	3260
6.178list File Reference . . . . .	3261
6.178.1 Detailed Description . . . . .	3261

---

6.179list.tcc File Reference . . . . .	3262
6.179.1 Detailed Description . . . . .	3262
6.180list_partition.h File Reference . . . . .	3262
6.180.1 Detailed Description . . . . .	3262
6.181list_update_policy.hpp File Reference . . . . .	3263
6.181.1 Detailed Description . . . . .	3263
6.182locale File Reference . . . . .	3263
6.182.1 Detailed Description . . . . .	3263
6.183locale_classes.h File Reference . . . . .	3263
6.183.1 Detailed Description . . . . .	3264
6.184locale_classes.tcc File Reference . . . . .	3264
6.184.1 Detailed Description . . . . .	3265
6.185locale_facets.h File Reference . . . . .	3265
6.185.1 Detailed Description . . . . .	3267
6.186locale_facets.tcc File Reference . . . . .	3268
6.186.1 Detailed Description . . . . .	3268
6.187locale_facets_nonio.h File Reference . . . . .	3268
6.187.1 Detailed Description . . . . .	3270
6.188locale_facets_nonio.tcc File Reference . . . . .	3270
6.188.1 Detailed Description . . . . .	3270
6.189localefwd.h File Reference . . . . .	3270
6.189.1 Detailed Description . . . . .	3271
6.190losertree.h File Reference . . . . .	3271
6.190.1 Detailed Description . . . . .	3273
6.191macros.h File Reference . . . . .	3273
6.191.1 Detailed Description . . . . .	3273
6.191.2 Define Documentation . . . . .	3274
6.191.2.1 __glibcxx_check_erase . . . . .	3274
6.191.2.2 __glibcxx_check_erase_range . . . . .	3274
6.191.2.3 __glibcxx_check_heap_pred . . . . .	3274
6.191.2.4 __glibcxx_check_insert . . . . .	3274

---

6.191.2.5	<code>__glibcxx_check_insert_range</code>	3274
6.191.2.6	<code>__glibcxx_check_partitioned_lower</code>	3275
6.191.2.7	<code>__glibcxx_check_partitioned_lower_pred</code>	3275
6.191.2.8	<code>__glibcxx_check_partitioned_upper_pred</code>	3275
6.191.2.9	<code>__glibcxx_check_sorted_pred</code>	3275
6.191.2.10	<code>GLIBCXX_DEBUG_VERIFY</code>	3275
6.192	<code>malloc_allocator.h</code> File Reference	3276
6.192.1	Detailed Description	3276
6.193	<code>map</code> File Reference	3276
6.193.1	Detailed Description	3276
6.194	<code>map</code> File Reference	3277
6.194.1	Detailed Description	3277
6.195	<code>map</code> File Reference	3277
6.195.1	Detailed Description	3277
6.196	<code>map.h</code> File Reference	3277
6.196.1	Detailed Description	3278
6.197	<code>map.h</code> File Reference	3278
6.197.1	Detailed Description	3279
6.198	<code>mask_array.h</code> File Reference	3279
6.198.1	Detailed Description	3280
6.199	<code>memory</code> File Reference	3280
6.199.1	Detailed Description	3280
6.200	<code>memory</code> File Reference	3280
6.200.1	Detailed Description	3281
6.201	<code>merge.h</code> File Reference	3281
6.201.1	Detailed Description	3282
6.202	<code>messages_members.h</code> File Reference	3282
6.202.1	Detailed Description	3282
6.203	<code>move.h</code> File Reference	3282
6.203.1	Detailed Description	3283
6.204	<code>mt_allocator.h</code> File Reference	3283

---

6.204.1 Detailed Description . . . . .	3285
6.205multimap.h File Reference . . . . .	3285
6.205.1 Detailed Description . . . . .	3286
6.206multimap.h File Reference . . . . .	3286
6.206.1 Detailed Description . . . . .	3287
6.207multiset_selection.h File Reference . . . . .	3287
6.207.1 Detailed Description . . . . .	3288
6.208multiset.h File Reference . . . . .	3288
6.208.1 Detailed Description . . . . .	3289
6.209multiset.h File Reference . . . . .	3289
6.209.1 Detailed Description . . . . .	3290
6.210multiway_merge.h File Reference . . . . .	3290
6.210.1 Detailed Description . . . . .	3295
6.210.2 Define Documentation . . . . .	3296
6.210.2.1 _GLIBCXX_PARALLEL_LENGTH . . . . .	3296
6.211multiway_mergesort.h File Reference . . . . .	3296
6.211.1 Detailed Description . . . . .	3297
6.212mutex File Reference . . . . .	3297
6.212.1 Detailed Description . . . . .	3299
6.213nested_exception.h File Reference . . . . .	3299
6.213.1 Detailed Description . . . . .	3300
6.214new File Reference . . . . .	3300
6.214.1 Detailed Description . . . . .	3301
6.214.2 Function Documentation . . . . .	3301
6.214.2.1 operator delete . . . . .	3301
6.214.2.2 operator delete . . . . .	3301
6.214.2.3 operator delete . . . . .	3302
6.214.2.4 operator delete[] . . . . .	3302
6.214.2.5 operator delete[] . . . . .	3302
6.214.2.6 operator delete[] . . . . .	3303
6.214.2.7 operator new . . . . .	3303

---

6.214.2.8 operator new . . . . .	3303
6.214.2.9 operator new . . . . .	3304
6.214.2.10operator new[] . . . . .	3304
6.214.2.11operator new[] . . . . .	3304
6.214.2.12operator new[] . . . . .	3305
6.215new_allocator.h File Reference . . . . .	3305
6.215.1 Detailed Description . . . . .	3306
6.216numeric File Reference . . . . .	3306
6.216.1 Detailed Description . . . . .	3306
6.217numeric File Reference . . . . .	3306
6.217.1 Detailed Description . . . . .	3306
6.218numeric File Reference . . . . .	3307
6.218.1 Detailed Description . . . . .	3310
6.219numeric_traits.h File Reference . . . . .	3310
6.219.1 Detailed Description . . . . .	3310
6.220numeric_fwd.h File Reference . . . . .	3310
6.220.1 Detailed Description . . . . .	3313
6.221omp_loop.h File Reference . . . . .	3313
6.221.1 Detailed Description . . . . .	3313
6.222omp_loop_static.h File Reference . . . . .	3314
6.222.1 Detailed Description . . . . .	3314
6.223os_defines.h File Reference . . . . .	3314
6.223.1 Detailed Description . . . . .	3314
6.224ostream File Reference . . . . .	3315
6.224.1 Detailed Description . . . . .	3316
6.225ostream.tcc File Reference . . . . .	3316
6.225.1 Detailed Description . . . . .	3317
6.226ostream_insert.h File Reference . . . . .	3317
6.226.1 Detailed Description . . . . .	3317
6.227par_loop.h File Reference . . . . .	3317
6.227.1 Detailed Description . . . . .	3318

---

6.228parallel.h File Reference . . . . .	3318
6.228.1 Detailed Description . . . . .	3318
6.229partial_sum.h File Reference . . . . .	3318
6.229.1 Detailed Description . . . . .	3319
6.230partition.h File Reference . . . . .	3319
6.230.1 Detailed Description . . . . .	3320
6.230.2 Define Documentation . . . . .	3320
6.230.2.1 _GLIBCXX_VOLATILE . . . . .	3320
6.231pod_char_traits.h File Reference . . . . .	3320
6.231.1 Detailed Description . . . . .	3321
6.232pointer.h File Reference . . . . .	3321
6.232.1 Detailed Description . . . . .	3323
6.233pool_allocator.h File Reference . . . . .	3324
6.233.1 Detailed Description . . . . .	3324
6.234postypes.h File Reference . . . . .	3324
6.234.1 Detailed Description . . . . .	3325
6.235priority_queue.hpp File Reference . . . . .	3325
6.235.1 Detailed Description . . . . .	3326
6.236priority_queue_base_dispatch.hpp File Reference . . . . .	3326
6.236.1 Detailed Description . . . . .	3326
6.237profiler.h File Reference . . . . .	3326
6.237.1 Detailed Description . . . . .	3329
6.238profiler_hashtable_size.h File Reference . . . . .	3329
6.238.1 Detailed Description . . . . .	3330
6.239profiler_list_to_slist.h File Reference . . . . .	3330
6.239.1 Detailed Description . . . . .	3331
6.240profiler_list_to_vector.h File Reference . . . . .	3331
6.240.1 Detailed Description . . . . .	3331
6.241profiler_map_to_unordered_map.h File Reference . . . . .	3332
6.241.1 Detailed Description . . . . .	3333
6.242profiler_node.h File Reference . . . . .	3333

---

6.242.1 Detailed Description . . . . .	3334
6.243 profiler_state.h File Reference . . . . .	3334
6.243.1 Detailed Description . . . . .	3334
6.244 profiler_trace.h File Reference . . . . .	3335
6.244.1 Detailed Description . . . . .	3337
6.245 profiler_vector_size.h File Reference . . . . .	3337
6.245.1 Detailed Description . . . . .	3338
6.246 profiler_vector_to_list.h File Reference . . . . .	3338
6.246.1 Detailed Description . . . . .	3339
6.247 queue File Reference . . . . .	3339
6.247.1 Detailed Description . . . . .	3339
6.248 queue.h File Reference . . . . .	3339
6.248.1 Detailed Description . . . . .	3340
6.248.2 Define Documentation . . . . .	3340
6.248.2.1 _GLIBCXX_VOLATILE . . . . .	3340
6.249 quicksort.h File Reference . . . . .	3340
6.249.1 Detailed Description . . . . .	3341
6.250 random File Reference . . . . .	3341
6.250.1 Detailed Description . . . . .	3341
6.251 random.h File Reference . . . . .	3341
6.251.1 Detailed Description . . . . .	3348
6.252 random.tcc File Reference . . . . .	3348
6.252.1 Detailed Description . . . . .	3353
6.253 random_number.h File Reference . . . . .	3353
6.253.1 Detailed Description . . . . .	3354
6.254 random_shuffle.h File Reference . . . . .	3354
6.254.1 Detailed Description . . . . .	3355
6.255 ratio File Reference . . . . .	3355
6.255.1 Detailed Description . . . . .	3357
6.256 rb_tree File Reference . . . . .	3357
6.256.1 Detailed Description . . . . .	3357

---

6.257rc_string_base.h File Reference . . . . .	3357
6.257.1 Detailed Description . . . . .	3357
6.258regex File Reference . . . . .	3358
6.258.1 Detailed Description . . . . .	3358
6.259rope File Reference . . . . .	3358
6.259.1 Detailed Description . . . . .	3362
6.260ropeimpl.h File Reference . . . . .	3362
6.260.1 Detailed Description . . . . .	3363
6.261safe_base.h File Reference . . . . .	3363
6.261.1 Detailed Description . . . . .	3363
6.262safe_iterator.h File Reference . . . . .	3363
6.262.1 Detailed Description . . . . .	3365
6.263safe_iterator.tcc File Reference . . . . .	3365
6.263.1 Detailed Description . . . . .	3365
6.264safe_sequence.h File Reference . . . . .	3365
6.264.1 Detailed Description . . . . .	3366
6.265search.h File Reference . . . . .	3366
6.265.1 Detailed Description . . . . .	3366
6.266set File Reference . . . . .	3366
6.266.1 Detailed Description . . . . .	3366
6.267set File Reference . . . . .	3367
6.267.1 Detailed Description . . . . .	3367
6.268set File Reference . . . . .	3367
6.268.1 Detailed Description . . . . .	3367
6.269set.h File Reference . . . . .	3367
6.269.1 Detailed Description . . . . .	3368
6.270set.h File Reference . . . . .	3368
6.270.1 Detailed Description . . . . .	3369
6.271set_operations.h File Reference . . . . .	3369
6.271.1 Detailed Description . . . . .	3370
6.272settings.h File Reference . . . . .	3370

---

6.272.1 Detailed Description . . . . .	3371
6.272.2 parallelization_decision . . . . .	3371
6.272.3 Define Documentation . . . . .	3372
6.272.3.1 _GLIBCXX_PARALLEL_CONDITION . . . . .	3372
6.273shared_ptr.h File Reference . . . . .	3372
6.273.1 Detailed Description . . . . .	3373
6.274shared_ptr_base.h File Reference . . . . .	3374
6.274.1 Detailed Description . . . . .	3374
6.275slice_array.h File Reference . . . . .	3374
6.275.1 Detailed Description . . . . .	3374
6.276slist File Reference . . . . .	3374
6.276.1 Detailed Description . . . . .	3376
6.277sort.h File Reference . . . . .	3376
6.277.1 Detailed Description . . . . .	3377
6.278sso_string_base.h File Reference . . . . .	3377
6.278.1 Detailed Description . . . . .	3377
6.279sstream File Reference . . . . .	3377
6.279.1 Detailed Description . . . . .	3378
6.280sstream.tcc File Reference . . . . .	3378
6.280.1 Detailed Description . . . . .	3378
6.281stack File Reference . . . . .	3378
6.281.1 Detailed Description . . . . .	3378
6.282standard_policies.hpp File Reference . . . . .	3379
6.282.1 Detailed Description . . . . .	3379
6.283stdatomic.h File Reference . . . . .	3379
6.283.1 Detailed Description . . . . .	3379
6.284stdexcept File Reference . . . . .	3379
6.284.1 Detailed Description . . . . .	3380
6.285stdio_filebuf.h File Reference . . . . .	3380
6.285.1 Detailed Description . . . . .	3380
6.286stdio_sync_filebuf.h File Reference . . . . .	3380

---

6.286.1 Detailed Description . . . . .	3381
6.287stl_algo.h File Reference . . . . .	3381
6.287.1 Detailed Description . . . . .	3394
6.288stl_algo.h File Reference . . . . .	3394
6.288.1 Detailed Description . . . . .	3397
6.289stl_bvector.h File Reference . . . . .	3397
6.289.1 Detailed Description . . . . .	3398
6.290stl_construct.h File Reference . . . . .	3398
6.290.1 Detailed Description . . . . .	3398
6.291stl_deque.h File Reference . . . . .	3399
6.291.1 Detailed Description . . . . .	3402
6.291.2 Define Documentation . . . . .	3402
6.291.2.1 _GLIBCXX_DEQUE_BUF_SIZE . . . . .	3402
6.292stl_function.h File Reference . . . . .	3402
6.292.1 Detailed Description . . . . .	3405
6.293stl_heap.h File Reference . . . . .	3405
6.293.1 Detailed Description . . . . .	3407
6.294stl_iterator.h File Reference . . . . .	3407
6.294.1 Detailed Description . . . . .	3411
6.295stl_iterator_base_funcs.h File Reference . . . . .	3411
6.295.1 Detailed Description . . . . .	3412
6.296stl_iterator_base_types.h File Reference . . . . .	3412
6.296.1 Detailed Description . . . . .	3413
6.297stl_list.h File Reference . . . . .	3413
6.297.1 Detailed Description . . . . .	3415
6.298stl_map.h File Reference . . . . .	3415
6.298.1 Detailed Description . . . . .	3416
6.299stl_multimap.h File Reference . . . . .	3416
6.299.1 Detailed Description . . . . .	3417
6.300stl_multiset.h File Reference . . . . .	3417
6.300.1 Detailed Description . . . . .	3418

---

6.301	stl_numeric.h File Reference	3418
6.301.1	Detailed Description	3419
6.302	stl_pair.h File Reference	3419
6.302.1	Detailed Description	3420
6.303	stl_queue.h File Reference	3420
6.303.1	Detailed Description	3421
6.304	stl_raw_storage_iter.h File Reference	3421
6.304.1	Detailed Description	3422
6.305	stl_relops.h File Reference	3422
6.305.1	Detailed Description	3422
6.306	stl_set.h File Reference	3423
6.306.1	Detailed Description	3423
6.307	stl_stack.h File Reference	3424
6.307.1	Detailed Description	3424
6.308	stl_tempbuf.h File Reference	3425
6.308.1	Detailed Description	3425
6.309	stl_tree.h File Reference	3425
6.309.1	Detailed Description	3427
6.310	stl_uninitialized.h File Reference	3427
6.310.1	Detailed Description	3428
6.311	stl_vector.h File Reference	3429
6.311.1	Detailed Description	3429
6.312	stream_iterator.h File Reference	3430
6.312.1	Detailed Description	3430
6.313	streambuf File Reference	3430
6.313.1	Detailed Description	3431
6.314	streambuf.tcc File Reference	3431
6.314.1	Detailed Description	3432
6.315	streambuf_iterator.h File Reference	3432
6.315.1	Detailed Description	3433
6.316	string File Reference	3433

---

---

6.316.1 Detailed Description . . . . .	3433
6.317string File Reference . . . . .	3433
6.317.1 Detailed Description . . . . .	3436
6.318stringfwd.h File Reference . . . . .	3436
6.318.1 Detailed Description . . . . .	3436
6.319system_error File Reference . . . . .	3437
6.319.1 Detailed Description . . . . .	3438
6.320tag_and_trait.hpp File Reference . . . . .	3438
6.320.1 Detailed Description . . . . .	3440
6.321tags.h File Reference . . . . .	3440
6.321.1 Detailed Description . . . . .	3442
6.322tgmath.h File Reference . . . . .	3442
6.322.1 Detailed Description . . . . .	3442
6.323thread File Reference . . . . .	3442
6.323.1 Detailed Description . . . . .	3443
6.324throw_allocator.h File Reference . . . . .	3443
6.324.1 Detailed Description . . . . .	3446
6.325time_members.h File Reference . . . . .	3446
6.325.1 Detailed Description . . . . .	3446
6.326tree_policy.hpp File Reference . . . . .	3446
6.326.1 Detailed Description . . . . .	3447
6.327tree_trace_base.hpp File Reference . . . . .	3447
6.327.1 Detailed Description . . . . .	3447
6.328trie_policy.hpp File Reference . . . . .	3447
6.328.1 Detailed Description . . . . .	3447
6.329tuple File Reference . . . . .	3448
6.329.1 Detailed Description . . . . .	3450
6.330type_traits File Reference . . . . .	3450
6.330.1 Detailed Description . . . . .	3452
6.331type_traits File Reference . . . . .	3452
6.331.1 Detailed Description . . . . .	3456

---

6.332	<a href="#">type_traits.h</a> File Reference . . . . .	3456
6.332.1	Detailed Description . . . . .	3456
6.333	<a href="#">type_utils.hpp</a> File Reference . . . . .	3456
6.333.1	Detailed Description . . . . .	3457
6.334	<a href="#">typeinfo</a> File Reference . . . . .	3457
6.334.1	Detailed Description . . . . .	3457
6.335	<a href="#">typelist.h</a> File Reference . . . . .	3458
6.335.1	Detailed Description . . . . .	3459
6.336	<a href="#">types.h</a> File Reference . . . . .	3459
6.336.1	Detailed Description . . . . .	3460
6.337	<a href="#">types_traits.hpp</a> File Reference . . . . .	3460
6.337.1	Detailed Description . . . . .	3460
6.338	<a href="#">unique_copy.h</a> File Reference . . . . .	3460
6.338.1	Detailed Description . . . . .	3461
6.339	<a href="#">unique_ptr.h</a> File Reference . . . . .	3461
6.339.1	Detailed Description . . . . .	3462
6.340	<a href="#">unordered_map</a> File Reference . . . . .	3462
6.340.1	Detailed Description . . . . .	3462
6.341	<a href="#">unordered_map</a> File Reference . . . . .	3462
6.341.1	Detailed Description . . . . .	3463
6.342	<a href="#">unordered_map</a> File Reference . . . . .	3463
6.342.1	Detailed Description . . . . .	3465
6.343	<a href="#">unordered_map.h</a> File Reference . . . . .	3465
6.343.1	Detailed Description . . . . .	3467
6.344	<a href="#">unordered_set</a> File Reference . . . . .	3467
6.344.1	Detailed Description . . . . .	3467
6.345	<a href="#">unordered_set</a> File Reference . . . . .	3467
6.345.1	Detailed Description . . . . .	3468
6.346	<a href="#">unordered_set</a> File Reference . . . . .	3468
6.346.1	Detailed Description . . . . .	3469
6.347	<a href="#">unordered_set.h</a> File Reference . . . . .	3469

---

6.347.1 Detailed Description . . . . .	3471
6.348utility File Reference . . . . .	3471
6.348.1 Detailed Description . . . . .	3471
6.349utility File Reference . . . . .	3471
6.349.1 Detailed Description . . . . .	3472
6.350valarray File Reference . . . . .	3472
6.350.1 Detailed Description . . . . .	3477
6.351valarray_after.h File Reference . . . . .	3477
6.351.1 Detailed Description . . . . .	3492
6.352valarray_array.h File Reference . . . . .	3492
6.352.1 Detailed Description . . . . .	3502
6.353valarray_array.tcc File Reference . . . . .	3503
6.353.1 Detailed Description . . . . .	3504
6.354valarray_before.h File Reference . . . . .	3504
6.354.1 Detailed Description . . . . .	3504
6.355vector File Reference . . . . .	3504
6.355.1 Detailed Description . . . . .	3504
6.356vector File Reference . . . . .	3504
6.356.1 Detailed Description . . . . .	3505
6.357vector File Reference . . . . .	3505
6.357.1 Detailed Description . . . . .	3506
6.358vector.tcc File Reference . . . . .	3507
6.358.1 Detailed Description . . . . .	3507
6.359vstring.h File Reference . . . . .	3507
6.359.1 Detailed Description . . . . .	3511
6.360vstring.tcc File Reference . . . . .	3511
6.360.1 Detailed Description . . . . .	3512
6.361vstring_fwd.h File Reference . . . . .	3512
6.361.1 Detailed Description . . . . .	3513
6.362vstring_util.h File Reference . . . . .	3513
6.362.1 Detailed Description . . . . .	3513

---

6.363workstealing.h File Reference . . . . .	3513
6.363.1 Detailed Description . . . . .	3514



# Chapter 1

## Todo List

**Member `__glibcxx_check_insert_range`**(`_Position`, `_First`, `_Last`) We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

**Member `__gnu_cxx::distance`**(`_InputIterator` `__first`, `_InputIterator` `__last`, `Distance` & `__n`)  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::hash_map`**< `_Key`, `_Tp`, `_HashFn`, `_EqualKey`, `_Alloc` >  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::hash_multimap`**< `_Key`, `_Tp`, `_HashFn`, `_EqualKey`, `_Alloc` >  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::hash_multiset`**< `_Value`, `_HashFn`, `_EqualKey`, `_Alloc` >  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::iota(_ForwardIter __first, _ForwardIter __last, _Tp __value)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::is_heap(_RandomAccessIterator __first, _RandomAccessIterator __last)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::is_heap(_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::is_sorted(_ForwardIterator __first, _ForwardIterator __last)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::is_sorted(_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::power(_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `__gnu_cxx::power(_Tp __x, _Integer __n)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

---

**Member `__gnu_cxx::random_sample`**(`InputIterator __first`, `InputIterator __last`, `RandomAccessIterator __out_first`, `RandomAccessIterator __out_last`)  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `__gnu_cxx::random_sample`**(`InputIterator __first`, `InputIterator __last`, `RandomAccessIterator __out_first`, `RandomAccessIterator __out_last`)  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `__gnu_cxx::random_sample_n`**(`ForwardIterator __first`, `ForwardIterator __last`, `OutputIterator __out`, `const_iterator __n`)  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `__gnu_cxx::random_sample_n`**(`ForwardIterator __first`, `ForwardIterator __last`, `OutputIterator __out`, `const_iterator __n`, `const_iterator __seed`)  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::rb_tree`**< `_Key`, `_Value`, `_KeyOfValue`, `_Compare`, `_Alloc` >  
Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::rope`**< `_CharT`, `_Alloc` > Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Class `__gnu_cxx::slist`**< `_Tp`, `_Alloc` > Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `__gnu_debug::Safe_iterator`**< `_Iterator`, `_Sequence` >::`operator->()` const  
Make this correct w.r.t. iterators that return proxies  
Use `addressof()` instead of `&` operator

Class `std::basic_string<_CharT, _Traits, _Alloc>`

Doc me! See [doc/doxygen/TODO](http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard(unsigned long long __z)`  
Look for a faster way to do discard.

Member `std::discard_block_engine<_RandomNumberEngine, __p, __r>::max() const`  
This should be constexpr.

Member `std::discard_block_engine<_RandomNumberEngine, __p, __r>::min() const`  
This should be constexpr.

Member `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::discard(unsigned long long __z)`  
Look for a faster way to do discard.

Member `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max() const`  
This should be constexpr.

Member `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min() const`  
This should be constexpr.

Member `std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard(unsigned long long __z)`  
Look for a faster way to do discard.

Member `std::linear_congruential_engine<_UIntType, __a, __c, __m>::max() const`  
This should be constexpr.

Member `std::linear_congruential_engine<_UIntType, __a, __c, __m>::min() const`  
This should be constexpr.

Member `std::reverse_iterator<_Iterator>::operator*() const`

Doc me! See [doc/doxygen/TODO](http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

---

**Member `std::reverse_iterator<_Iterator>::operator+(difference_type __n) const`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator++()`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator++(int)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator+=(difference_type __n)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator-(difference_type __n) const`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator--()`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator--(int)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

**Member `std::reverse_iterator<_Iterator>::operator--=(difference_type __n)`**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator->()` const

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator[](difference_type __n)` const

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::discard(unsigned long long __z)`

Look for a faster way to do discard.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::max()` const

This should be constexpr.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::min()` const

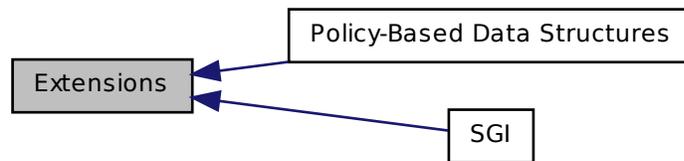
This should be constexpr.

## Chapter 2

# Module Documentation

### 2.1 Extensions

Collaboration diagram for Extensions:



#### Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >`  
*Template class `__versa_string`.*  
*Data structure managing sequences of characters and character-like objects.*

#### Modules

- [SGI](#)

- [Policy-Based Data Structures](#)

### 2.1.1 Detailed Description

Components generally useful that are not part of any standard.

## 2.2 SGI

Collaboration diagram for SGI:



### Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`  
*An SGI extension .*
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`  
*An SGI extension .*
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`  
*An SGI extension .*
- struct `__gnu_cxx::constant_void_fun< _Result >`  
*An SGI extension .*
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`  
*An SGI extension .*

- struct `__gnu_cxx::project2nd<_Arg1, _Arg2 >`  
*An SGI extension .*
- struct `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope<_CharT, _Alloc >`
- struct `__gnu_cxx::select1st<_Pair >`  
*An SGI extension .*
- struct `__gnu_cxx::select2nd<_Pair >`  
*An SGI extension .*
- class `__gnu_cxx::slist<_Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose<_Operation1, _Operation2 >`  
*An SGI extension .*

## Functions

- `template<typename _Tp >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `size_t std::bitset::_Find_first () const`
- `size_t std::bitset::_Find_next (size_t __prev) const`
- `template<class _Operation1, class _Operation2 >`  
`unary_compose<_Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`  
`binary_compose<_Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun<_Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun<_Result, _Result > __gnu_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant_binary_fun<_Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)`

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > \_\_gnu\_cxx::copy\_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::multiplies< _Tp >)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::plus< _Tp >)`
- `template<typename _ForwardIter, typename _Tp >`  
`void \_\_gnu\_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_gnu\_cxx::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool \_\_gnu\_cxx::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`  
`bool \_\_gnu\_cxx::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`  
`bool \_\_gnu\_cxx::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int \_\_gnu\_cxx::lexicographical\_compare\_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator \_\_gnu\_cxx::random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator \_\_gnu\_cxx::random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n (_ForwardIterator __first, _-`  
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _-`  
`RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::uninitialized\_copy\_n (_InputIter`  
`__first, _Size __count, _ForwardIter __result)`
- `bitset< _Nb > & std::bitset::\_Unchecked\_set (size_t __pos)`
- `bitset< _Nb > & std::bitset::\_Unchecked\_set (size_t __pos, int __val)`
- `bitset< _Nb > & std::bitset::\_Unchecked\_reset (size_t __pos)`
- `bitset< _Nb > & std::bitset::\_Unchecked\_flip (size_t __pos)`
- `bool std::bitset::\_Unchecked\_test (size_t __pos) const`

### 2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

**NB** that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

## 2.2.2 Function Documentation

### 2.2.2.1 `template<typename _Tp> const _Tp& __gnu_cxx::__median ( const _Tp & __a, const _Tp & __b, const _Tp & __c )`

Find the median of three values.

#### Parameters

- a* A value.
- b* A value.
- c* A value.

#### Returns

One of *a*, *b* or *c*.

If  $\{1,m,n\}$  is some convolution of  $\{a,b,c\}$  such that  $1 \leq m \leq n$  then the value returned will be  $m$ . This is an SGI extension.

Definition at line 537 of file ext/algorithm.

```
2.2.2.2 template<typename _Tp , typename _Compare > const _Tp&
  __gnu_cxx::__median ( const _Tp & __a, const _Tp & __b, const _Tp
    & __c, _Compare __comp )
```

Find the median of three values using a predicate for comparison.

#### Parameters

- a* A value.
- b* A value.
- c* A value.
- comp* A binary predicate.

#### Returns

One of  $a$ ,  $b$  or  $c$ .

If  $\{1,m,n\}$  is some convolution of  $\{a,b,c\}$  such that  $\text{comp}(1, m)$  and  $\text{comp}(m, n)$  are both true then the value returned will be  $m$ . This is an SGI extension.

Definition at line 571 of file ext/algorithm.

```
2.2.2.3 template<size_t _Nb> size_t std::bitset<_Nb >::_Find_first ( ) const
  [inline, inherited]
```

Finds the index of the first "on" bit.

#### Returns

The index of the first bit set, or [size\(\)](#) if not found.

#### See also

[\\_Find\\_next](#)

Definition at line 1304 of file bitset.

```
2.2.2.4 template<size_t _Nb> size_t std::bitset<_Nb >::_Find_next ( size_t
  __prev ) const [inline, inherited]
```

Finds the index of the next "on" bit after *prev*.

**Returns**

The index of the next bit set, or `size()` if not found.

**Parameters**

*prev* Where to start searching.

**See also**

`_Find_first`

Definition at line 1315 of file `bitset`.

**2.2.2.5** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb  
>::_Unchecked_flip ( size_t __pos ) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 987 of file `bitset`.

**2.2.2.6** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb  
>::_Unchecked_reset ( size_t __pos ) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 980 of file `bitset`.

**2.2.2.7** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb  
>::_Unchecked_set ( size_t __pos ) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 963 of file `bitset`.

**2.2.2.8** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb  
>::_Unchecked_set ( size_t __pos, int __val ) [inline,  
inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 970 of file `bitset`.

**2.2.2.9** `template<size_t _Nb> bool std::bitset<_Nb>::_Unchecked_test ( size_t __pos ) const [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 994 of file `bitset`.

**2.2.2.10** `template<class _Operation1 , class _Operation2 > unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1 ( const _Operation1 & __fn1, const _Operation2 & __fn2 ) [inline]`

An [SGI extension](#) .

Definition at line 145 of file `ext/functional`.

**2.2.2.11** `template<class _Operation1 , class _Operation2 , class _Operation3 > binary_compose<_Operation1, _Operation2, _Operation3> __gnu_cxx::compose2 ( const _Operation1 & __fn1, const _Operation2 & __fn2, const _Operation3 & __fn3 ) [inline]`

An [SGI extension](#) .

Definition at line 172 of file `ext/functional`.

**2.2.2.12** `template<class _Result > constant_void_fun<_Result> __gnu_cxx::constant0 ( const _Result & __val ) [inline]`

An [SGI extension](#) .

Definition at line 326 of file `ext/functional`.

**2.2.2.13** `template<class _Result > constant_unary_fun<_Result, _Result> __gnu_cxx::constant1 ( const _Result & __val ) [inline]`

An [SGI extension](#) .

Definition at line 332 of file `ext/functional`.

**2.2.2.14** `template<class _Result > constant_binary_fun<_Result, _Result, _Result> __gnu_cxx::constant2 ( const _Result & __val ) [inline]`

An [SGI extension](#) .

Definition at line 338 of file ext/functional.

```
2.2.2.15 template<typename _InputIterator , typename _Size , typename
           _OutputIterator > pair<_InputIterator, _OutputIterator>
           __gnu_cxx::copy_n ( _InputIterator __first, _Size __count,
           _OutputIterator __result ) [inline]
```

Copies the range [first,first+count) into [result,result+count).

#### Parameters

*first* An input iterator.

*count* The number of elements to copy.

*result* An output iterator.

#### Returns

A [std::pair](#) composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to memmove whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 119 of file ext/algorithm.

References [std::\\_\\_iterator\\_category\(\)](#).

```
2.2.2.16 template<typename _InputIterator , typename _Distance > void
           __gnu_cxx::distance ( _InputIterator __first, _InputIterator __last,
           _Distance & __n ) [inline]
```

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

Definition at line 103 of file ext/iterator.

References [std::\\_\\_iterator\\_category\(\)](#).

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), and [std::list<\\_Tp, \\_Alloc >::size\(\)](#).

**2.2.2.17** `template<class _Tp > _Tp __gnu_cxx::identity_element ( std::plus< _Tp > ) [inline]`

An [SGI extension](#) .

Definition at line 87 of file ext/functional.

**2.2.2.18** `template<class _Tp > _Tp __gnu_cxx::identity_element ( std::multiplies< _Tp > ) [inline]`

An [SGI extension](#) .

Definition at line 93 of file ext/functional.

**2.2.2.19** `template<typename _ForwardIter , typename _Tp > void __gnu_cxx::iota ( _ForwardIter __first, _ForwardIter __last, _Tp __value )`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 132 of file ext/numeric.

**2.2.2.20** `template<typename _RandomAccessIterator , typename _StrictWeakOrdering > bool __gnu_cxx::is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp ) [inline]`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 453 of file ext/algorithm.

**2.2.2.21** `template<typename _RandomAccessIterator > bool __gnu_cxx::is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last ) [inline]`

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 434 of file ext/algorithm.

**2.2.2.22** `template<typename _ForwardIterator > bool __gnu_cxx::is_sorted (`  
`_FForwardIterator __first, _ForwardIterator __last )`

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 478 of file ext/algorithm.

**2.2.2.23** `template<typename _ForwardIterator , typename`  
`_StrictWeakOrdering > bool __gnu_cxx::is_sorted (`  
`_FForwardIterator __first, _ForwardIterator __last,`  
`_StrictWeakOrdering __comp )`

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 503 of file ext/algorithm.

**2.2.2.24** `template<typename _InputIterator1 , typename _InputIterator2 >`  
`int __gnu_cxx::lexicographical_compare_3way ( _InputIterator1`  
`__first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2 )`

memcmp on steroids.

**Parameters**

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

*last2* An input iterator.

### Returns

An int, as with `memcmp`.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 200 of file `ext/algorithm`.

**2.2.2.25** `template<typename _Tp , typename _Integer > _Tp  
__gnu_cxx::power ( _Tp __x, _Integer __n ) [inline]`

This is an SGI extension.

### Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 121 of file `ext/numeric`.

**2.2.2.26** `template<typename _Tp , typename _Integer , typename  
_MonoidOperation > _Tp __gnu_cxx::power ( _Tp __x, _Integer  
__n, _MonoidOperation __monoid_op ) [inline]`

This is an SGI extension.

### Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 111 of file `ext/numeric`.

**2.2.2.27** `template<typename _InputIterator , typename  
_RandomAccessIterator , typename _RandomNumberGenerator >  
_RandomAccessIterator __gnu_cxx::random_sample ( _InputIterator  
__first, _InputIterator __last, _RandomAccessIterator __out_first,  
_RandomAccessIterator __out_last, _RandomNumberGenerator &  
__rand ) [inline]`

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 410 of file ext/algorithm.

```
2.2.2.28 template<typename _InputIterator , typename
_RandomAccessIterator > _RandomAccessIterator
_gnu_cxx::random_sample ( _InputIterator __first, _InputIterator
__last, _RandomAccessIterator __out_first, _RandomAccessIterator
__out_last ) [inline]
```

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 387 of file ext/algorithm.

```
2.2.2.29 template<typename _ForwardIterator , typename _OutputIterator ,
typename _Distance , typename _RandomNumberGenerator >
_OutputIterator _gnu_cxx::random_sample_n ( _ForwardIterator
__first, _ForwardIterator __last, _OutputIterator __out, const
_Distance __n, _RandomNumberGenerator & __rand )
```

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 300 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.30 template<typename _ForwardIterator , typename _OutputIterator ,
typename _Distance > _OutputIterator _gnu_cxx::random_sample_n
( _ForwardIterator __first, _ForwardIterator __last,
_OutputIterator __out, const _Distance __n )
```

This is an SGI extension.

### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 266 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.31 template<typename _InputIter, typename _Size,
               typename _ForwardIter > pair<_InputIter, _ForwardIter>
  __gnu_cxx::uninitialized_copy_n( _InputIter __first, _Size __count,
    _ForwardIter __result ) [inline]
```

Copies the range `[first,last)` into `result`.

### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

### Returns

`result + (first - last)`

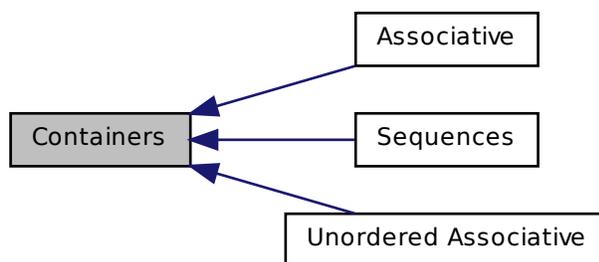
Like `copy()`, but does not require an initialized output range.

Definition at line 121 of file ext/memory.

References `std::__iterator_category()`.

## 2.3 Containers

Collaboration diagram for Containers:



### Classes

- class `std::bitset<_Nb>`

*The bitset class represents a fixed-size sequence of bits.*

### Modules

- [Sequences](#)
- [Associative](#)
- [Unordered Associative](#)

#### 2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in [tables](#).

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

## 2.4 Sequences

Collaboration diagram for Sequences:



### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`  
*Managing sequences of characters and character-like objects.*
- class `std::deque<_Tp, _Alloc>`  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*
- class `std::forward_list<_Tp, _Alloc>`  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*
- class `std::list<_Tp, _Alloc>`  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class `std::priority_queue<_Tp, _Sequence, _Compare >`  
*A standard container automatically sorting its contents.*
- class `std::queue<_Tp, _Sequence >`  
*A standard container giving FIFO behavior.*
- class `std::stack<_Tp, _Sequence >`  
*A standard container giving FILO behavior.*
- struct `std::tr1::array<_Tp, _Nm >`  
*A standard container for storing a fixed size sequence of elements.*
- class `std::vector<_Tp, _Alloc >`  
*A standard container which offers fixed time access to individual elements in any order.*
- class `std::vector<bool, _Alloc >`  
*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*

### 2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

## 2.5 Associative

Collaboration diagram for Associative:



### Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- class `std::multiset< _Key, _Compare, _Alloc >`  
*A standard container made up of elements, which can be retrieved in logarithmic time.*
- class `std::set< _Key, _Compare, _Alloc >`  
*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

### 2.5.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

## 2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



### Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >`  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >`  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >`  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc >`  
*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

### 2.6.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

## 2.7 Diagnostics

Collaboration diagram for Diagnostics:



### Modules

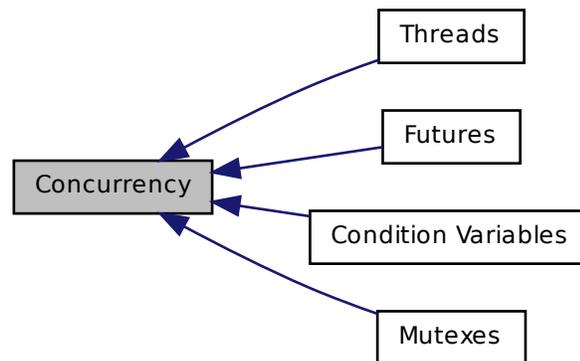
- [Exceptions](#)

#### 2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

## 2.8 Concurrency

Collaboration diagram for Concurrency:



### Modules

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

### 2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

## 2.9 Exceptions

Collaboration diagram for Exceptions:



### Classes

- class `__cxxabiv1::__forced_unwind`  
*Thrown as part of forced unwinding.  
 A magic placeholder class that can be caught by reference to recognize forced unwinding.*
- struct `__gnu_cxx::forced_error`  
*Thrown by exception safety machinery.*
- class `__gnu_cxx::recursive_init_error`  
*Exception thrown by `__cxa_guard_acquire`.  
 6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `std::__exception_ptr::exception_ptr`  
*An opaque pointer to an arbitrary exception.*
- class `std::bad_alloc`  
*Exception possibly thrown by `new`.  
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*
- class `std::bad_cast`  
*Thrown during incorrect typecasting.  
 If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `std::bad_exception`
- class `std::bad_function_call`

*Exception class thrown when class template function's operator() is called with an empty target.*

- class [std::bad\\_typeid](#)

*Thrown when a NULL pointer in a typeid expression is used.*

- class [std::domain\\_error](#)
- class [std::exception](#)

*Base class for all library exceptions.*

- class [std::future\\_error](#)

*Exception type thrown by futures.*

- class [std::invalid\\_argument](#)
- class [std::ios\\_base::failure](#)

*These are thrown to indicate problems with io.  
27.4.2.1.1 Class ios\_base::failure.*

- class [std::length\\_error](#)
- class [std::logic\\_error](#)

*One of two subclasses of exception.*

- class [std::nested\\_exception](#)

*Exception class with exception\_ptr data member.*

- class [std::out\\_of\\_range](#)
- class [std::overflow\\_error](#)
- class [std::range\\_error](#)
- class [std::runtime\\_error](#)

*One of two subclasses of exception.*

- class [std::system\\_error](#)

*Thrown to indicate error code of underlying system.*

- class [std::tr1::bad\\_weak\\_ptr](#)

*Exception possibly thrown by shared\_ptr.*

- class [std::underflow\\_error](#)

## Typedefs

- typedef void(\* [std::terminate\\_handler](#) )()
- typedef void(\* [std::unexpected\\_handler](#) )()

## Functions

- `template<typename _Ex >`  
`const nested_exception * std::__get_nested_exception (const _Ex &__ex)`
- `template<typename _Ex >`  
`void std::__throw_with_nested (_Ex &&, const nested_exception *__ex) __-`  
`attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void std::__throw_with_nested (_Ex &&, ...) __attribute__((__noreturn__))`
- `void __gnu_cxx::__verbose_terminate_handler ()`
- `template<typename _Ex >`  
`exception_ptr std::copy_exception (_Ex __ex) throw ()`
- `exception_ptr std::current_exception () throw ()`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`  
`void std::rethrow_if_nested (const _Ex &__ex)`
- `terminate_handler std::set_terminate (terminate_handler) throw ()`
- `unexpected_handler std::set_unexpected (unexpected_handler) throw ()`
- `void std::terminate () __attribute__((__noreturn__)) throw ()`
- `template<typename _Ex >`  
`void std::throw_with_nested (_Ex __ex)`
- `bool std::uncaught_exception () __attribute__((__pure__)) throw ()`
- `void std::unexpected () __attribute__((__noreturn__))`

### 2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

### 2.9.2 Typedef Documentation

#### 2.9.2.1 typedef void(\* std::terminate\_handler)()

If you write a replacement terminate handler, it must be of this type.

Definition at line 88 of file exception.

#### 2.9.2.2 typedef void(\* std::unexpected\_handler)()

If you write a replacement unexpected handler, it must be of this type.

Definition at line 91 of file exception.

### 2.9.3 Function Documentation

#### 2.9.3.1 void \_\_gnu\_cxx::\_\_verbose\_terminate\_handler ( )

A replacement for the standard terminate\_handler which prints more information about the terminating exception (if any) on stderr.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06>. In 3.4 and later, this is on by default.

#### 2.9.3.2 template<typename \_Ex > exception\_ptr std::copy\_exception ( \_Ex \_\_ex ) throw ()

Obtain an exception\_ptr pointing to a copy of the supplied object.

Definition at line 149 of file exception\_ptr.h.

References std::current\_exception().

#### 2.9.3.3 exception\_ptr std::current\_exception ( ) throw ()

Obtain an exception\_ptr to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by std::copy\_exception().

#### 2.9.3.4 void std::rethrow\_exception ( exception\_ptr )

Throw the object pointed to by the exception\_ptr.

Referenced by std::\_\_basic\_future< \_Res & >::\_M\_get\_result().

#### 2.9.3.5 void std::rethrow\_if\_nested ( const nested\_exception & \_\_ex ) [inline]

Overload, See N2619.

Definition at line 156 of file nested\_exception.h.

**2.9.3.6** `template<typename _Ex > void std::rethrow_if_nested ( const _Ex & __ex ) [inline]`

If `__ex` is derived from `nested_exception`, `__ex.rethrow_nested()`.

Definition at line 148 of file `nested_exception.h`.

**2.9.3.7** `terminate_handler std::set_terminate ( terminate_handler ) throw ()`

Takes a new handler function as an argument, returns the old function.

**2.9.3.8** `unexpected_handler std::set_unexpected ( unexpected_handler ) throw ()`

Takes a new handler function as an argument, returns the old function.

**2.9.3.9** `void std::terminate ( ) throw ()`

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

**2.9.3.10** `template<typename _Ex > void std::throw_with_nested ( _Ex __ex ) [inline]`

If `__ex` is derived from `nested_exception`, `__ex`. // Else, an implementation-defined object derived from both.

Definition at line 138 of file `nested_exception.h`.

**2.9.3.11** `bool std::uncaught_exception ( ) throw ()`

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

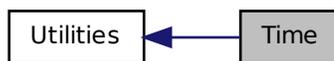
Referenced by `std::basic_ostream<_CharT, _Traits >::sentry::~sentry()`.

### 2.9.3.12 void std::unexpected ( )

The runtime will call this function if an exception is thrown which violates the function's exception specification.

## 2.10 Time

Collaboration diagram for Time:



### Namespaces

- namespace [std::chrono](#)

### 2.10.1 Detailed Description

Classes and functions for time.

## 2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



## Classes

- struct `std::complex< _Tp >`

## Functions

- `std::complex< float >::complex` (const `complex< double > &`)
- `std::complex< float >::complex` (const `complex< long double > &`)
- `std::complex< double >::complex` (const `complex< long double > &`)
- `template<typename _Tp > _Tp std::__complex_abs` (const `complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_acos` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_acosh` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > _Tp std::__complex_arg` (const `complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_asin` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_asinh` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_atan` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > std::complex< _Tp > std::tr1::__complex_atanh` (const `std::complex< _Tp > &_z`)
- `template<typename _Tp > complex< _Tp > std::__complex_cos` (const `complex< _Tp > &_z`)
- `template<typename _Tp > complex< _Tp > std::__complex_cosh` (const `complex< _Tp > &_z`)
- `template<typename _Tp > complex< _Tp > std::__complex_exp` (const `complex< _Tp > &_z`)
- `template<typename _Tp > complex< _Tp > std::__complex_log` (const `complex< _Tp > &_z`)
- `template<typename _Tp > complex< _Tp > std::__complex_pow` (const `complex< _Tp > &_x`, const `complex< _Tp > &_y`)
- `template<typename _Tp > complex< _Tp > std::__complex_sin` (const `complex< _Tp > &_z`)

- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`

- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex::operator\*= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator\*= (const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex::operator-= (const complex< _Up > &)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex::operator/= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator/= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex::operator= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp >`  
`&)`

- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up >`  
`&__y)`
- `template<typename _Tp >`  
`_Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__-`  
`__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__-`  
`__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex<`  
`_Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__-`  
`y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__-`  
`y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex<`  
`_Tp > &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &_`  
`__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &_`  
`__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex<`  
`_Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &_`  
`__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

### 2.11.1 Detailed Description

Classes and functions for complex numbers.

### 2.11.2 Function Documentation

#### 2.11.2.1 `template<typename _Tp > _Tp std::abs ( const complex< _Tp > & __z ) [inline]`

Return magnitude of  $z$ .

Definition at line 594 of file complex.

Referenced by `std::tr1::fabs()`, `std::binomial_distribution<_IntType>::operator()`, and `std::poisson_distribution<_IntType>::operator()`.

**2.11.2.2** `template<typename _Tp> std::complex<_Tp> std::tr1::acos ( const std::complex<_Tp> & __z ) [inline]`

`acos(__z)` [8.1.2].

Definition at line 86 of file `tr1_impl/complex`.

**2.11.2.3** `template<typename _Tp> std::complex<_Tp> std::tr1::acosh ( const std::complex<_Tp> & __z ) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 205 of file `tr1_impl/complex`.

**2.11.2.4** `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type std::tr1::arg ( _Tp __x ) [inline]`

Additional overloads [8.1.9].

Definition at line 311 of file `tr1_impl/complex`.

References `std::arg()`.

**2.11.2.5** `template<typename _Tp> _Tp std::arg ( const complex<_Tp> & __z ) [inline]`

Return phase angle of  $z$ .

Definition at line 621 of file `complex`.

Referenced by `std::tr1::arg()`.

**2.11.2.6** `template<typename _Tp> std::complex<_Tp> std::tr1::asin ( const std::complex<_Tp> & __z ) [inline]`

`asin(__z)` [8.1.3].

Definition at line 122 of file `tr1_impl/complex`.

**2.11.2.7** `template<typename _Tp > std::complex< _Tp > std::tr1::asinh ( const std::complex< _Tp > & __z ) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 244 of file `tr1_impl/complex`.

**2.11.2.8** `template<typename _Tp > std::complex< _Tp > std::tr1::atan ( const std::complex< _Tp > & __z ) [inline]`

`atan(__z)` [8.1.4].

Definition at line 166 of file `tr1_impl/complex`.

**2.11.2.9** `template<typename _Tp > std::complex< _Tp > std::tr1::atanh ( const std::complex< _Tp > & __z ) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 288 of file `tr1_impl/complex`.

**2.11.2.10** `template<typename _Tp > complex< _Tp > std::conj ( const complex< _Tp > & __z ) [inline]`

Return complex conjugate of  $z$ .

Definition at line 667 of file `complex`.

**2.11.2.11** `template<typename _Tp > complex< _Tp > std::cos ( const complex< _Tp > & __z ) [inline]`

Return complex cosine of  $z$ .

Definition at line 699 of file `complex`.

Referenced by `std::polar()`.

**2.11.2.12** `template<typename _Tp > complex< _Tp > std::cosh ( const complex< _Tp > & __z ) [inline]`

Return complex hyperbolic cosine of  $z$ .

Definition at line 729 of file `complex`.

**2.11.2.13** `template<typename _Tp > complex< _Tp > std::exp ( const complex< _Tp > & __z ) [inline]`

Return complex base e exponential of z.

Definition at line 755 of file complex.

Referenced by `std::pow()`.

**2.11.2.14** `template<typename _Tp > _Tp std::tr1::fabs ( const std::complex< _Tp > & __z ) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 301 of file `tr1_impl/complex`.

References `std::abs()`.

**2.11.2.15** `template<typename _Tp > complex< _Tp > std::log ( const complex< _Tp > & __z ) [inline]`

Return complex natural logarithm of z.

Definition at line 782 of file complex.

Referenced by `std::generate_canonical()`, `std::log10()`, `std::gamma_distribution< _RealType >::operator()`, `std::normal_distribution< _RealType >::operator()`, `std::binomial_distribution< _IntType >::operator()`, `std::poisson_distribution< _IntType >::operator()`, `std::independent_bits_engine< _RandomNumberEngine, _w, _UIntType >::operator()`, and `std::pow()`.

**2.11.2.16** `template<typename _Tp > complex< _Tp > std::log10 ( const complex< _Tp > & __z ) [inline]`

Return complex base 10 logarithm of z.

Definition at line 787 of file complex.

References `std::log()`.

**2.11.2.17** `template<typename _Tp > _Tp std::norm ( const complex< _Tp > & __z ) [inline]`

Return z magnitude squared.

Definition at line 654 of file complex.

Referenced by `std::complex< _Tp >::operator/=(())`.

**2.11.2.18** `template<typename _Tp > bool std::operator!=( const complex<_Tp > & __x, const complex<_Tp > & __y ) [inline]`

Return false if  $x$  is equal to  $y$ .

Definition at line 469 of file complex.

**2.11.2.19** `template<typename _Tp > bool std::operator!=( const complex<_Tp > & __x, const _Tp & __y ) [inline]`

Return false if  $x$  is equal to  $y$ .

Definition at line 474 of file complex.

**2.11.2.20** `template<typename _Tp > bool std::operator!=( const _Tp & __x, const complex<_Tp > & __y ) [inline]`

Return false if  $x$  is equal to  $y$ .

Definition at line 479 of file complex.

**2.11.2.21** `template<typename _Tp > complex<_Tp> std::operator*( const complex<_Tp > & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  times  $y$ .

Definition at line 379 of file complex.

**2.11.2.22** `template<typename _Tp > complex<_Tp> std::operator*( const complex<_Tp > & __x, const _Tp & __y ) [inline]`

Return new complex value  $x$  times  $y$ .

Definition at line 388 of file complex.

**2.11.2.23** `template<typename _Tp > complex<_Tp> std::operator*( const _Tp & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  times  $y$ .

Definition at line 397 of file complex.

**2.11.2.24** `template<typename _Tp > complex<_Tp > & std::complex<_Tp >::operator*=( const _Tp & __t ) [inherited]`

Multiply this complex number by  $t$ .

Definition at line 238 of file complex.

**2.11.2.25** `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator*=( const complex<_Up > & __z ) [inherited]`

Multiply this complex number by  $z$ .

Definition at line 292 of file complex.

**2.11.2.26** `template<typename _Tp > complex<_Tp> std::operator+( const complex<_Tp > & __x ) [inline]`

Return  $x$ .

Definition at line 438 of file complex.

**2.11.2.27** `template<typename _Tp > complex<_Tp> std::operator+( const _Tp & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  plus  $y$ .

Definition at line 337 of file complex.

**2.11.2.28** `template<typename _Tp > complex<_Tp> std::operator+( const complex<_Tp > & __x, const _Tp & __y ) [inline]`

Return new complex value  $x$  plus  $y$ .

Definition at line 328 of file complex.

**2.11.2.29** `template<typename _Tp > complex<_Tp> std::operator+( const complex<_Tp > & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  plus  $y$ .

Definition at line 319 of file complex.

**2.11.2.30** `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator+=( const complex<_Up > & _z ) [inherited]`

Add  $z$  to this complex number.

Definition at line 269 of file complex.

**2.11.2.31** `template<typename _Tp > complex<_Tp> std::operator-( const complex<_Tp > & _x ) [inline]`

Return complex negation of  $x$ .

Definition at line 444 of file complex.

**2.11.2.32** `template<typename _Tp > complex<_Tp> std::operator-( const complex<_Tp > & _x, const complex<_Tp > & _y ) [inline]`

Return new complex value  $x$  minus  $y$ .

Definition at line 349 of file complex.

**2.11.2.33** `template<typename _Tp > complex<_Tp> std::operator-( const complex<_Tp > & _x, const _Tp & _y ) [inline]`

Return new complex value  $x$  minus  $y$ .

Definition at line 358 of file complex.

**2.11.2.34** `template<typename _Tp > complex<_Tp> std::operator-( const _Tp & _x, const complex<_Tp > & _y ) [inline]`

Return new complex value  $x$  minus  $y$ .

Definition at line 367 of file complex.

**2.11.2.35** `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator-=( const complex<_Up > & _z ) [inherited]`

Subtract  $z$  from this complex number.

Definition at line 280 of file complex.

**2.11.2.36** `template<typename _Tp > complex<_Tp> std::operator/ ( const complex<_Tp > & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  divided by  $y$ .

Definition at line 409 of file `complex`.

**2.11.2.37** `template<typename _Tp > complex<_Tp> std::operator/ ( const complex<_Tp > & __x, const _Tp & __y ) [inline]`

Return new complex value  $x$  divided by  $y$ .

Definition at line 418 of file `complex`.

**2.11.2.38** `template<typename _Tp > complex<_Tp> std::operator/ ( const _Tp & __x, const complex<_Tp > & __y ) [inline]`

Return new complex value  $x$  divided by  $y$ .

Definition at line 427 of file `complex`.

**2.11.2.39** `template<typename _Tp > complex<_Tp > & std::complex<_Tp >::operator/= ( const _Tp & __t ) [inherited]`

Divide this complex number by  $t$ .

Definition at line 248 of file `complex`.

**2.11.2.40** `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator/= ( const complex<_Up > & __z ) [inherited]`

Divide this complex number by  $z$ .

Definition at line 305 of file `complex`.

References `std::norm()`.

**2.11.2.41** `template<typename _Tp , typename _CharT , class _Traits > basic_ostream<_CharT, _Traits>& std::operator<< ( basic_ostream<_CharT, _Traits > & __os, const complex<_Tp > & __x )`

Insertion operator for complex values.

Definition at line 519 of file complex.

References `std::ios_base::flags()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::ios_base::precision()`, and `std::basic_ostringstream<_CharT, _Traits, _Alloc>::str()`.

**2.11.2.42** `template<typename _Tp> complex<_Tp> & std::complex<_Tp>::operator=( const _Tp & __t ) [inherited]`

Assign this complex number to scalar  $t$ .

Definition at line 228 of file complex.

**2.11.2.43** `template<typename _Tp> template<typename _Up> complex<_Tp> & std::complex<_Tp>::operator=( const complex<_Up> & __z ) [inherited]`

Assign this complex number to complex  $z$ .

Definition at line 258 of file complex.

**2.11.2.44** `template<typename _Tp> bool std::operator==( const complex<_Tp> & __x, const _Tp & __y ) [inline]`

Return true if  $x$  is equal to  $y$ .

Definition at line 456 of file complex.

**2.11.2.45** `template<typename _Tp> bool std::operator==( const complex<_Tp> & __x, const complex<_Tp> & __y ) [inline]`

Return true if  $x$  is equal to  $y$ .

Definition at line 451 of file complex.

**2.11.2.46** `template<typename _Tp> bool std::operator==( const _Tp & __x, const complex<_Tp> & __y ) [inline]`

Return true if  $x$  is equal to  $y$ .

Definition at line 461 of file complex.

**2.11.2.47** `template<typename _Tp , typename _CharT , class _Traits  
> basic_istream<_CharT, _Traits>& std::operator>> (   
basic_istream< _CharT, _Traits > & __is, complex< _Tp > & __x )`

Extraction operator for complex values.

Definition at line 486 of file `complex`.

References `std::ios_base::failbit`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**2.11.2.48** `template<typename _Tp > complex< _Tp > std::polar ( const _Tp  
& __rho, const _Tp & __theta = 0 ) [inline]`

Return complex with magnitude *rho* and angle *theta*.

Definition at line 662 of file `complex`.

References `std::cos()`, and `std::sin()`.

Referenced by `std::pow()`.

**2.11.2.49** `template<typename _Tp > complex< _Tp > std::pow ( const _Tp &  
__x, const complex< _Tp > & __y ) [inline]`

Return *x* to the *y*'th power.

Definition at line 1009 of file `complex`.

References `std::log()`, `std::polar()`, and `std::pow()`.

**2.11.2.50** `template<typename _Tp > complex< _Tp > std::pow ( const  
complex< _Tp > & __x, const _Tp & __y )`

Return *x* to the *y*'th power.

Definition at line 964 of file `complex`.

References `std::exp()`, `std::log()`, and `std::polar()`.

Referenced by `std::gamma_distribution< _RealType >::operator()()`, and `std::pow()`.

**2.11.2.51** `template<typename _Tp > complex< _Tp > std::pow ( const  
complex< _Tp > & __x, const complex< _Tp > & __y )  
[inline]`

Return *x* to the *y*'th power.

Definition at line 1003 of file `complex`.

**2.11.2.52** `template<typename _Tp> complex<_Tp> std::sin ( const  
complex<_Tp> & _z ) [inline]`

Return complex sine of  $z$ .

Definition at line 817 of file `complex`.

Referenced by `std::polar()`.

**2.11.2.53** `template<typename _Tp> complex<_Tp> std::sinh ( const  
complex<_Tp> & _z ) [inline]`

Return complex hyperbolic sine of  $z$ .

Definition at line 847 of file `complex`.

**2.11.2.54** `template<typename _Tp> complex<_Tp> std::sqrt ( const  
complex<_Tp> & _z ) [inline]`

Return complex square root of  $z$ .

Definition at line 891 of file `complex`.

Referenced by `std::normal_distribution<_RealType>::operator()`, and  
`std::student_t_distribution<_RealType>::operator()`.

**2.11.2.55** `template<typename _Tp> complex<_Tp> std::tan ( const  
complex<_Tp> & _z ) [inline]`

Return complex tangent of  $z$ .

Definition at line 918 of file `complex`.

**2.11.2.56** `template<typename _Tp> complex<_Tp> std::tanh ( const  
complex<_Tp> & _z ) [inline]`

Return complex hyperbolic tangent of  $z$ .

Definition at line 946 of file `complex`.

## 2.12 Condition Variables

Collaboration diagram for Condition Variables:



### Classes

- class [std::condition\\_variable](#)  
*condition\_variable*
- class [std::condition\\_variable\\_any](#)  
*condition\_variable\_any*

### Enumerations

- enum [std::cv\\_status](#) { **no\_timeout**, **timeout** }

#### 2.12.1 Detailed Description

Classes for [condition\\_variable](#) support.

#### 2.12.2 Enumeration Type Documentation

##### 2.12.2.1 enum [std::cv\\_status](#)

[cv\\_status](#)

Definition at line 54 of file [condition\\_variable](#).

## 2.13 Futures

Collaboration diagram for Futures:



### Classes

- class `std::__basic_future< _Res >`  
*Common implementation for future and `shared_future`.*
- struct `std::__future_base`  
*Base class and enclosing scope.*
- struct `std::__future_base::_Result< _Res & >`  
*Partial specialization for reference types.*
- struct `std::__future_base::_Result< void >`  
*Explicit specialization for void.*
- class `std::future< _Res >`  
*Primary template for future.*
- class `std::future< _Res & >`  
*Partial specialization for `future<R&>`*
- class `std::future< void >`  
*Explicit specialization for `future<void>`*
- class `std::future_error`  
*Exception type thrown by futures.*
- class `std::packaged_task< _Res(_ArgTypes...)>`  
*`packaged_task`*

- class `std::promise<_Res >`  
*Primary template for promise.*
- class `std::promise<_Res & >`  
*Partial specialization for promise<R&>*
- class `std::promise< void >`  
*Explicit specialization for promise<void>*
- class `std::shared_future<_Res >`  
*Primary template for shared\_future.*
- class `std::shared_future<_Res & >`  
*Partial specialization for shared\_future<R&>*
- class `std::shared_future< void >`  
*Explicit specialization for shared\_future<void>*

## Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

## Functions

- `std::__basic_future::__basic_future` (const shared\_future<\_Res > &)
- `std::__basic_future::__basic_future` (shared\_future<\_Res > &&)
- `std::__basic_future::__basic_future` (future<\_Res > &&)
- static `_Setter< void, void > std::__future_base::State::__setter` (promise<void > \*\_\_prom)
- `template<typename _Fn, typename... _Args>`  
`enable_if<!is_same< typename decay<_Fn >::type, launch >::value, future< decltype(std::declval<_Fn >)(std::declval<_Args >)...> >::type std::async`  
(`_Fn &&__fn, _Args &&...__args`)
- `template<typename _Fn, typename... _Args>`  
`future< typename result_of<_Fn(_Args...)>::type > std::async` (launch `__policy, _Fn &&__fn, _Args &&...__args`)
- error\_code `std::make_error_code` (future\_errc `__errc`)
- error\_condition `std::make_error_condition` (future\_errc `__errc`)

- void **std::promise**< **void** >::**set\_value** ()
- template<typename **\_Res** , typename... **\_ArgTypes**>  
void **std::swap** (packaged\_task< **\_Res**(**\_ArgTypes**...)> &\_\_x, packaged\_task< **\_Res**(**\_ArgTypes**...)> &\_\_y)
- template<typename **\_Res** >  
void **std::swap** (promise< **\_Res** > &\_\_x, promise< **\_Res** > &\_\_y)

## Variables

- const error\_category \*const [std::future\\_category](#)

### 2.13.1 Detailed Description

Classes for futures support.

### 2.13.2 Enumeration Type Documentation

#### 2.13.2.1 enum [std::future\\_errc](#)

Error code for futures.

Definition at line 59 of file future.

### 2.13.3 Variable Documentation

#### 2.13.3.1 [const error\\_category\\*](#) [const std::future\\_category](#)

Points to a statically-allocated object derived from [error\\_category](#).

## 2.14 I/O

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf](#)< **\_CharT**, **\_Traits** >  
*Provides a layer of compatibility for C/POSIX.  
This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., [stdio\\_filebuf](#)<char>.*
- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf](#)< **\_CharT**, **\_Traits** >

*Provides a layer of compatibility for C.*

*This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `std::filebuf<char>`.*

- class [std::basic\\_filebuf<\\_CharT, \\_Traits >](#)  
*The actual work of input and output (for files).  
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.*
- class [std::basic\\_fstream<\\_CharT, \\_Traits >](#)  
*Controlling input and output for files.  
This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ifstream<\\_CharT, \\_Traits >](#)  
*Controlling input for files.  
This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ios<\\_CharT, \\_Traits >](#)  
*Virtual base class for all stream classes.  
Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*
- class [std::basic\\_iostream<\\_CharT, \\_Traits >](#)  
*Merging istream and ostream capabilities.  
This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class [std::basic\\_istream<\\_CharT, \\_Traits >](#)  
*Controlling input.  
This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.*
- class [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc >](#)  
*Controlling input for `std::string`.  
This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ofstream<\\_CharT, \\_Traits >](#)

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ostream<_CharT, _Traits >`

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

- class `std::basic_ostringstream<_CharT, _Traits, _Alloc >`

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::basic_streambuf<_CharT, _Traits >`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

- class `std::basic_stringbuf<_CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

- class `std::basic_stringstream<_CharT, _Traits, _Alloc >`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::ios_base`

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

## Typedefs

- typedef `basic_filebuf< char >` `std::filebuf`
- typedef `basic_fstream< char >` `std::fstream`

- `typedef basic_ifstream< char > std::ifstream`
- `typedef basic_ios< char > std::ios`
- `typedef basic_iostream< char > std::iostream`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istreamstream< char > std::istreamstream`
- `typedef basic_ofstream< char > std::ofstream`
- `typedef basic_ostream< char > std::ostream`
- `typedef basic_ostreamstream< char > std::ostreamstream`
- `typedef basic_streambuf< char > std::streambuf`
- `typedef basic_stringbuf< char > std::stringbuf`
- `typedef basic_stringstream< char > std::stringstream`
- `typedef basic_filebuf< wchar_t > std::wfilebuf`
- `typedef basic_fstream< wchar_t > std::wfstream`
- `typedef basic_ifstream< wchar_t > std::wifstream`
- `typedef basic_ios< wchar_t > std::wios`
- `typedef basic_iostream< wchar_t > std::wiostream`
- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_istreamstream< wchar_t > std::wistreamstream`
- `typedef basic_ofstream< wchar_t > std::wofstream`
- `typedef basic_ostream< wchar_t > std::wostream`
- `typedef basic_ostreamstream< wchar_t > std::wostreamstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream< wchar_t > std::wstringstream`

### 2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt>

## 2.14.2 Typedef Documentation

### 2.14.2.1 typedef basic\_filebuf<char> std::filebuf

One of the [I/O](#) typedefs.

Definition at line 137 of file iosfwd.

### 2.14.2.2 typedef basic\_fstream<char> std::fstream

One of the [I/O](#) typedefs.

Definition at line 140 of file iosfwd.

### 2.14.2.3 typedef basic\_ifstream<char> std::ifstream

One of the [I/O](#) typedefs.

Definition at line 138 of file iosfwd.

### 2.14.2.4 typedef basic\_ios<char> std::ios

One of the [I/O](#) typedefs.

Definition at line 123 of file iosfwd.

### 2.14.2.5 typedef basic\_iostream<char> std::iostream

One of the [I/O](#) typedefs.

Definition at line 132 of file iosfwd.

### 2.14.2.6 typedef basic\_istream<char> std::istream

One of the [I/O](#) typedefs.

Definition at line 130 of file iosfwd.

### 2.14.2.7 typedef basic\_istringstream<char> std::istringstream

One of the [I/O](#) typedefs.

Definition at line 134 of file iosfwd.

**2.14.2.8 typedef basic\_ofstream<char> std::ofstream**

One of the [I/O](#) typedefs.

Definition at line 139 of file iosfwd.

**2.14.2.9 typedef basic\_ostream<char> std::ostream**

One of the [I/O](#) typedefs.

Definition at line 131 of file iosfwd.

**2.14.2.10 typedef basic\_ostringstream<char> std::ostringstream**

One of the [I/O](#) typedefs.

Definition at line 135 of file iosfwd.

**2.14.2.11 typedef basic\_streambuf<char> std::streambuf**

One of the [I/O](#) typedefs.

Definition at line 129 of file iosfwd.

**2.14.2.12 typedef basic\_stringbuf<char> std::stringbuf**

One of the [I/O](#) typedefs.

Definition at line 133 of file iosfwd.

**2.14.2.13 typedef basic\_stringstream<char> std::stringstream**

One of the [I/O](#) typedefs.

Definition at line 136 of file iosfwd.

**2.14.2.14 typedef basic\_filebuf<wchar\_t> std::wfilebuf**

One of the [I/O](#) typedefs.

Definition at line 152 of file iosfwd.

**2.14.2.15 typedef basic\_fstream<wchar\_t> std::wfstream**

One of the [I/O](#) typedefs.

Definition at line 155 of file iosfwd.

**2.14.2.16 typedef basic\_ifstream<wchar\_t> std::wifstream**

One of the [I/O](#) typedefs.

Definition at line 153 of file iosfwd.

**2.14.2.17 typedef basic\_ios<wchar\_t> std::wios**

One of the [I/O](#) typedefs.

Definition at line 143 of file iosfwd.

**2.14.2.18 typedef basic\_iostream<wchar\_t> std::wiostream**

One of the [I/O](#) typedefs.

Definition at line 147 of file iosfwd.

**2.14.2.19 typedef basic\_istream<wchar\_t> std::wistream**

One of the [I/O](#) typedefs.

Definition at line 145 of file iosfwd.

**2.14.2.20 typedef basic\_istreamstream<wchar\_t> std::wistreamstream**

One of the [I/O](#) typedefs.

Definition at line 149 of file iosfwd.

**2.14.2.21 typedef basic\_ofstream<wchar\_t> std::wofstream**

One of the [I/O](#) typedefs.

Definition at line 154 of file iosfwd.

**2.14.2.22 typedef basic\_ostream<wchar\_t> std::wostream**

One of the [I/O](#) typedefs.

Definition at line 146 of file iosfwd.

**2.14.2.23 typedef basic\_ostringstream<wchar\_t> std::wostringstream**

One of the [I/O](#) typedefs.

Definition at line 150 of file iosfwd.

**2.14.2.24 typedef basic\_streambuf<wchar\_t> std::wstreambuf**

One of the [I/O](#) typedefs.

Definition at line 144 of file iosfwd.

**2.14.2.25 typedef basic\_stringbuf<wchar\_t> std::wstringbuf**

One of the [I/O](#) typedefs.

Definition at line 148 of file iosfwd.

**2.14.2.26 typedef basic\_stringstream<wchar\_t> std::wstringstream**

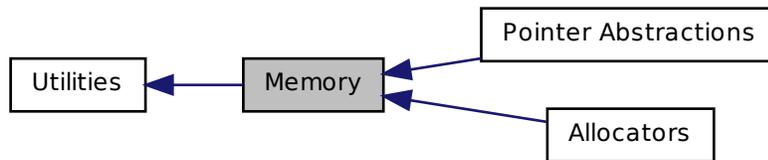
One of the [I/O](#) typedefs.

Definition at line 151 of file iosfwd.

---

## 2.15 Memory

Collaboration diagram for Memory:



### Modules

- [Pointer Abstractions](#)
- [Allocators](#)

#### 2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

## 2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



## Classes

- struct `std::default_delete< _Tp >`  
*Primary template, `default_delete`.*
- struct `std::default_delete< _Tp[ ]>`  
*Specialization, `default_delete`.*
- class `std::enable_shared_from_this< _Tp >`  
*Base class allowing use of member function `shared_from_this`.*
- struct `std::owner_less< shared_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `shared_ptr`.*
- struct `std::owner_less< weak_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `weak_ptr`.*
- class `std::shared_ptr< _Tp >`  
*A smart pointer with reference-counted copy semantics.*
- class `std::unique_ptr< _Tp, _Tp_Deleter >`  
*20.7.12.2 `unique_ptr` for single objects.*
- class `std::unique_ptr< _Tp[ ], _Tp_Deleter >`  
*20.7.12.3 `unique_ptr` for array objects with a runtime length*
- class `std::weak_ptr< _Tp >`  
*A smart pointer with weak semantics.*

## Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp > std::allocate_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`

- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _Tp > std::make\_shared (_Args &&... __args)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`  
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator!= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`  
`Tp2 > &__b)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic\_ostream< _Ch, _Tr > & std::operator<< (std::basic\_ostream< _Ch,`  
`_Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator== (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`  
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const`  
`unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp_Deleter >`  
`void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _-`  
`Tp_Deleter > &__y)`
- `template<typename _Tp >`  
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
- `template<typename _Tp >`  
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`

## 2.16.1 Detailed Description

Smart pointers, etc.

## 2.16.2 Function Documentation

**2.16.2.1** `template<typename _Tp , typename _Alloc , typename... _Args>  
shared_ptr<_Tp> std::allocate_shared ( _Alloc __a, _Args &&...  
__args ) [inline]`

Create an object that is owned by a [shared\\_ptr](#).

### Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

### Returns

A [shared\\_ptr](#) that owns the newly created object.

### Exceptions

*An* exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared\\_ptr](#) and the new object.

Definition at line 452 of file `shared_ptr.h`.

**2.16.2.2** `template<typename _Del , typename _Tp , _Lock_policy _Lp>  
_Del* std::get_deleter ( const __shared_ptr<_Tp, _Lp> & __p )  
[inline]`

2.2.3.10 [shared\\_ptr](#) `get_deleter` (experimental)

Definition at line 74 of file `shared_ptr.h`.

**2.16.2.3** `template<typename _Tp , typename... _Args> shared_ptr<_Tp>  
std::make_shared ( _Args &&... __args ) [inline]`

Create an object that is owned by a [shared\\_ptr](#).

### Parameters

`__args` Arguments for the `_Tp` object's constructor.

**Returns**

A [shared\\_ptr](#) that owns the newly created object.

**Exceptions**

[std::bad\\_alloc](#), or an exception thrown from the constructor of *\_Tp*.

Definition at line 467 of file `shared_ptr.h`.

```
2.16.2.4 template<typename _Ch , typename _Tr , typename _Tp
, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>&
std::operator<<( std::basic_ostream<_Ch, _Tr> & __os, const
__shared_ptr<_Tp, _Lp> & __p ) [inline]
```

2.2.3.7 [shared\\_ptr](#) I/O

Definition at line 64 of file `shared_ptr.h`.

## 2.17 Mutexes

Collaboration diagram for Mutexes:

**Classes**

- struct [std::adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [std::defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [std::lock\\_guard<\\_Mutex>](#)  
*Scoped lock idiom.*

- class `std::mutex`  
*mutex*
- struct `std::once_flag`  
*once\_flag*
- class `std::recursive_mutex`  
*recursive\_mutex*
- class `std::recursive_timed_mutex`  
*recursive\_timed\_mutex*
- class `std::timed_mutex`  
*timed\_mutex*
- struct `std::try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `std::unique_lock< _Mutex >`  
*unique\_lock*

## Functions

- `mutex & std::__get_once_mutex ()`
- `void std::__once_proxy ()`
- `void std::__set_once_functor_lock_ptr (unique_lock< mutex > *)`
- `template<typename _Callable, typename... _Args>`  
`void std::call_once (once_flag &__once, _Callable __f, _Args &&...__args)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void std::lock (_L1 &, _L2 &, _L3 &...)`
- `template<typename _Mutex >`  
`void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`  
`int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`

## Variables

- `function< void()> std::__once_functor`
- `const adopt_lock_t std::adopt_lock`
- `const defer_lock_t std::defer_lock`
- `const try_to_lock_t std::try_to_lock`

### 2.17.1 Detailed Description

Classes for mutex support.

### 2.17.2 Function Documentation

**2.17.2.1** `template<typename _Callable , typename... _Args> void  
std::call_once ( once_flag & __once, _Callable __f, _Args &&...  
__args )`

`call_once`

Definition at line 722 of file mutex.

**2.17.2.2** `template<typename _L1 , typename _L2 , typename... _L3> void  
std::lock ( _L1 & , _L2 & , _L3 & ... )`

`lock`

**2.17.2.3** `template<typename _Lock1 , typename _Lock2 , typename... _Lock3>  
int std::try_lock ( _Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3  
)`

Generic `try_lock`.

#### Parameters

`__l1` Meets Mutex requirements ([try\\_lock\(\)](#) may throw).

`__l2` Meets Mutex requirements ([try\\_lock\(\)](#) may throw).

`__l3` Meets Mutex requirements ([try\\_lock\(\)](#) may throw).

#### Returns

Returns -1 if all [try\\_lock\(\)](#) calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

#### Postcondition

Either all arguments are locked, or none will be.

Sequentially calls [try\\_lock\(\)](#) on each argument.

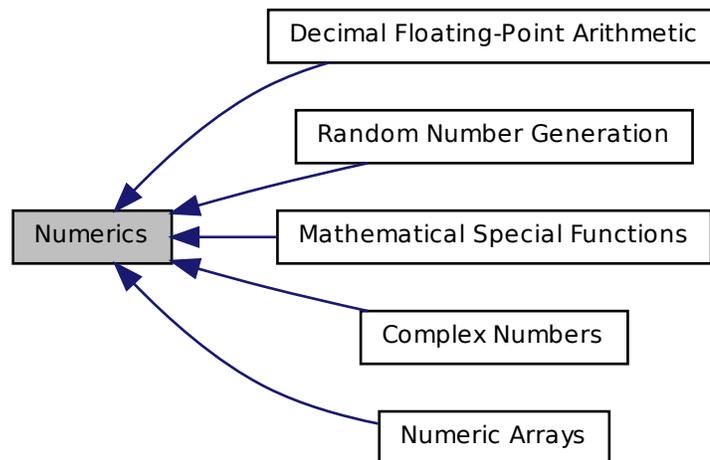
Definition at line 664 of file mutex.

References `std::try_lock()`.

Referenced by `std::try_lock()`.

## 2.18 Numerics

Collaboration diagram for Numerics:



### Modules

- [Complex Numbers](#)
- [Numeric Arrays](#)
- [Mathematical Special Functions](#)
- [Decimal Floating-Point Arithmetic](#)
- [Random Number Generation](#)

### 2.18.1 Detailed Description

Components for performing numeric operations. Includes support for for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

## 2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



### Classes

- struct `std::ratio< _Num, _Den >`  
*Provides compile-time rational arithmetic.*
- struct `std::ratio_add< _R1, _R2 >`  
*ratio\_add*
- struct `std::ratio_divide< _R1, _R2 >`  
*ratio\_divide*
- struct `std::ratio_equal< _R1, _R2 >`  
*ratio\_equal*
- struct `std::ratio_greater< _R1, _R2 >`  
*ratio\_greater*
- struct `std::ratio_greater_equal< _R1, _R2 >`  
*ratio\_greater\_equal*
- struct `std::ratio_less< _R1, _R2 >`  
*ratio\_less*
- struct `std::ratio_less_equal< _R1, _R2 >`  
*ratio\_less\_equal*
- struct `std::ratio_multiply< _R1, _R2 >`  
*ratio\_multiply*

- struct `std::ratio_not_equal<_R1, _R2 >`  
*ratio\_not\_equal*
- struct `std::ratio_subtract<_R1, _R2 >`  
*ratio\_subtract*

## Typedefs

- typedef `ratio< 1, 1000000000000000000 >` **std::atto**
- typedef `ratio< 1, 100 >` **std::centi**
- typedef `ratio< 10, 1 >` **std::deca**
- typedef `ratio< 1, 10 >` **std::deci**
- typedef `ratio< 1000000000000000000, 1 >` **std::exa**
- typedef `ratio< 1, 1000000000000000 >` **std::femto**
- typedef `ratio< 1000000000, 1 >` **std::giga**
- typedef `ratio< 100, 1 >` **std::hecto**
- typedef `ratio< 1000, 1 >` **std::kilo**
- typedef `ratio< 1000000, 1 >` **std::mega**
- typedef `ratio< 1, 1000000 >` **std::micro**
- typedef `ratio< 1, 1000 >` **std::milli**
- typedef `ratio< 1, 1000000000 >` **std::nano**
- typedef `ratio< 1000000000000000000, 1 >` **std::peta**
- typedef `ratio< 1, 1000000000000 >` **std::pico**
- typedef `ratio< 1000000000000, 1 >` **std::tera**

## Variables

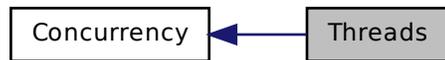
- static const `intmax_t` **std::ratio::den**
- static const `intmax_t` **std::ratio::num**

### 2.19.1 Detailed Description

Compile time representation of finite rational numbers.

## 2.20 Threads

Collaboration diagram for Threads:



### Classes

- struct `std::hash< thread::id >`  
*std::hash specialization for thread::id.*
- class `std::thread`  
*thread*

### Namespaces

- namespace `std::this_thread`

### Functions

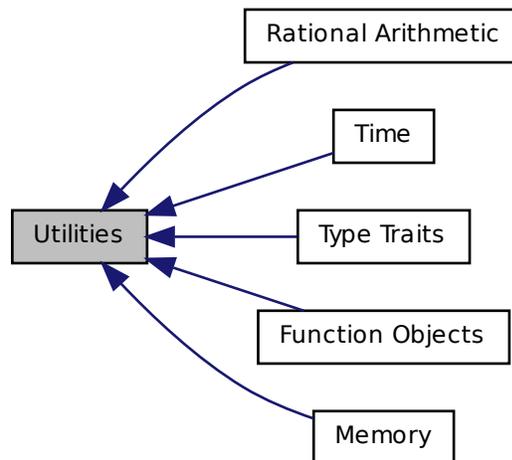
- bool `std::operator!=` (thread::id \_\_x, thread::id \_\_y)
- template<class \_CharT, class \_Traits >  
basic\_ostream< \_CharT, \_Traits > & `std::operator<<` (basic\_ostream< \_-  
CharT, \_Traits > &\_\_out, thread::id \_\_id)
- bool `std::operator<=` (thread::id \_\_x, thread::id \_\_y)
- bool `std::operator>` (thread::id \_\_x, thread::id \_\_y)
- bool `std::operator>=` (thread::id \_\_x, thread::id \_\_y)
- void `std::swap` (thread &\_\_x, thread &\_\_y)

#### 2.20.1 Detailed Description

Classes for thread support.

## 2.21 Utilities

Collaboration diagram for Utilities:



### Modules

- [Time](#)
- [Memory](#)
- [Rational Arithmetic](#)
- [Type Traits](#)
- [Function Objects](#)

#### 2.21.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

## 2.22 Numeric Arrays

Collaboration diagram for Numeric Arrays:



### Classes

- class `std::gslice`  
*Class defining multi-dimensional subset of an array.*
- class `std::gslice_array< _Tp >`  
*Reference to multi-dimensional subset of an array.*
- class `std::indirect_array< _Tp >`  
*Reference to arbitrary subset of an array.*
- class `std::mask_array< _Tp >`  
*Reference to selected subset of an array.*
- class `std::slice`  
*Class defining one-dimensional subset of an array.*
- class `std::slice_array< _Tp >`  
*Reference to one-dimensional subset of an array.*
- class `std::valarray< _Tp >`  
*Smart array designed to support numeric processing.*

### Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`

- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

## Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t, const valarray< size_t > &, const valarray< size_t > &)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array::gslice_array (const gslice_array &)`
- `std::indirect_array::indirect_array (const indirect_array &)`
- `std::mask_array::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t, size_t, size_t)`
- `std::slice_array::slice_array (const slice_array &)`
- `std::valarray::valarray (const mask_array< _Tp > &)`
- `template<class _Dom > std::valarray::valarray (const _Expr< _Dom, _Tp > &_e)`
- `std::valarray::valarray (const indirect_array< _Tp > &)`
- `std::valarray::valarray (initializer_list< _Tp >)`
- `std::valarray::valarray ()`
- `std::valarray::valarray (size_t)`
- `std::valarray::valarray (const _Tp &, size_t)`
- `template<typename _Tp> std::valarray::valarray (const _Tp *__restrict__ __p, size_t __n)`
- `std::valarray::valarray (const valarray &)`
- `std::valarray::valarray (const slice_array< _Tp > &)`
- `std::valarray::valarray (const gslice_array< _Tp > &)`
- `std::gslice::~gslice ()`
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(_Tp)) const`
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(const _Tp &)) const`
- `valarray< _Tp > std::valarray::cshift (int) const`
- `_Tp std::valarray::max () const`
- `_Tp std::valarray::min () const`

- `_UnaryOp< __logical_not >::_Rt std::valarray::operator! () const`
- `template<typename _Tp >  
_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-  
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const  
valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >  
_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-  
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _  
Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >  
_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-  
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const  
valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename  
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _  
Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename  
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _  
Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >  
_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename  
__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,  
const valarray< _Tp > &__v)`
- `void std::gslice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >  
void std::gslice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator%= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator%= (const valarray< _Tp > &)`
- `template<class _Dom >  
valarray< _Tp > & std::valarray::operator%= (const _Expr< _Dom, _Tp >  
&)`
- `template<typename _Tp >  
_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-  
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const  
valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >  
_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-  
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp  
&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >  
_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-  
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const  
valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _-`  
`Tp &_t, const valarray< _Tp > &_v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &_v, const _Tp &_t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &_v, const valarray< _Tp > &_w)`
- `void std::gslice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator&= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator&= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator&= (const _Expr< _Dom, _Tp >`  
`&)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _-`  
`Tp > &_v, const _Tp &_t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &_t,`  
`const valarray< _Tp > &_v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _-`  
`Tp > &_v, const valarray< _Tp > &_w)`
- `void std::gslice_array::operator* = (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator* = (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator* = (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator* = (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator* = (const _Expr< _Dom, _Tp >`  
`&)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _-`  
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`  
`&_v, const _Tp &_t)`

- `_UnaryOp< __unary_plus >::_Rt std::valarray::operator+ () const`
- `template<typename _Tp >  
_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _-  
_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >  
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _-  
_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const  
valarray< _Tp > &__v)`
- `void std::gslice_array::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >  
void std::gslice_array::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator+= (const valarray< _Tp > &)`
- `template<class _Dom >  
valarray< _Tp > & std::valarray::operator+= (const _Expr< _Dom, _Tp >  
&)`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _-  
_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >  
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _-  
_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >  
&__v, const _Tp &__t)`
- `_UnaryOp< __negate >::_Rt std::valarray::operator- () const`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _-  
_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const  
valarray< _Tp > &__v)`
- `void std::gslice_array::operator-= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator-= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator-= (const _Tp &)`
- `template<class _Dom >  
void std::gslice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >  
valarray< _Tp > & std::valarray::operator-= (const _Expr< _Dom, _Tp >  
&)`
- `template<typename _Tp >  
_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _-  
_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >  
&__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `void std::gslice_array::operator/= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator/= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator/= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator/= (const _Tp &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator/= (const _Expr< _Dom, _Tp >`  
`&)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `void std::gslice_array::operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator<<= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator<<= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator<<= (const valarray< _Tp > &)`

- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator<<= (const _Expr< _Dom, _Tp`  
`> &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator= (const _Expr< _Dom, _Tp > &)`
  
- `void std::slice_array::operator= (const valarray< _Tp > &) const`
- `void std::slice_array::operator= (const _Tp &) const`
- `template<class _Ex >`  
`void std::mask_array::operator= (const _Expr< _Ex, _Tp > &__e) const`
- `void std::mask_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const valarray< _Tp > &)`
- `gslice_array & std::gslice_array::operator= (const gslice_array &)`
- `void std::gslice_array::operator= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator= (const slice_array< _Tp > &)`
- `void std::gslice_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const indirect_array< _Tp > &)`
- `gslice & std::gslice::operator= (const gslice &)`
- `indirect_array & std::indirect_array::operator= (const indirect_array &)`
- `void std::indirect_array::operator= (const valarray< _Tp > &) const`
- `slice_array & std::slice_array::operator= (const slice_array &)`
- `mask_array & std::mask_array::operator= (const mask_array &)`
- `void std::mask_array::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::slice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `valarray & std::valarray::operator= (initializer_list< _Tp >)`
- `valarray< _Tp > & std::valarray::operator= (const gslice_array< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator= (const mask_array< _Tp > &)`

- `template<class _Dom >`  
`void std::indirect_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `void std::indirect_array::operator= (const _Tp &) const`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`

- `__fun< __shift_right, _Tp >::result_type > std::operator>>` (const valarray< \_Tp > &\_\_v, const \_Tp &\_\_t)
- `template<typename _Tp >`  
`__Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>>` (const \_Tp &\_\_t,  
const valarray< \_Tp > &\_\_v)
- `template<class _Dom >`  
`void std::gslice_array::operator>>=` (const \_Expr< \_Dom, \_Tp > &) const
- `valarray< _Tp > & std::valarray::operator>>=` (const valarray< \_Tp > &)
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator>>=` (const \_Expr< \_Dom, \_Tp  
`> &)`
- `void std::gslice_array::operator>>=` (const valarray< \_Tp > &) const
- `valarray< _Tp > & std::valarray::operator>>=` (const \_Tp &)
- `indirect_array< _Tp > std::valarray::operator[]` (const valarray< size\_t > &)
- `slice_array< _Tp > std::valarray::operator[]` (slice)
- `mask_array< _Tp > std::valarray::operator[]` (const valarray< bool > &)
- `valarray< _Tp > std::valarray::operator[]` (const valarray< bool > &) const
- `const _Tp & std::valarray::operator[]` (size\_t) const
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (const  
`gslice &) const`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (slice)  
`const`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (const  
`valarray< size_t > &) const`
- `_Tp & std::valarray::operator[]` (size\_t)
- `gslice_array< _Tp > std::valarray::operator[]` (const gslice &)
- `template<typename _Tp >`  
`__Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`__Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`__Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const \_Tp  
`&__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator^=` (const \_Expr< \_Dom, \_Tp >  
`&)`
- `valarray< _Tp > & std::valarray::operator^=` (const \_Tp &)

- `template<class _Dom >`  
`void std::gslice_array::operator^=(const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator^=(const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator^=(const valarray< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator|(const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator|(const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator|(const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray::operator|=(const _Expr< _Dom, _Tp >`  
`&)`
- `valarray< _Tp > & std::valarray::operator|=(const _Tp &)`
- `void std::gslice_array::operator|=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array::operator|=(const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator|=(const valarray< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator||(const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator||(const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator||(const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `_UnaryOp< __bitwise_not >::_Rt std::valarray::operator~() const`
- `void std::valarray::resize(size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray::shift(int) const`
- `size_t std::valarray::size() const`
- `size_t std::slice::size() const`
- `valarray< size_t > std::gslice::size() const`
- `size_t std::slice::start() const`
- `size_t std::gslice::start() const`

- `valarray< size_t > std::gslice::stride () const`
- `size_t std::slice::stride () const`
- `_Tp std::valarray::sum () const`

### 2.22.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

### 2.22.2 Function Documentation

#### 2.22.2.1 `std::gslice::gslice ( ) [inline, inherited]`

Construct an empty slice.

Definition at line 147 of file `gslice.h`.

#### 2.22.2.2 `std::gslice::gslice ( size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s ) [inline, inherited]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

#### Parameters

- o* Offset in array of first element.
- l* Array of dimension lengths.
- s* Array of dimension strides between array elements.

Definition at line 151 of file `gslice.h`.

#### 2.22.2.3 `std::gslice::gslice ( const gslice & __g ) [inline, inherited]`

Copy constructor.

Definition at line 156 of file `gslice.h`.

#### 2.22.2.4 `template<typename _Tp > std::gslice_array< _Tp >::gslice_array ( const gslice_array< _Tp > & __a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file `gslice_array.h`.

**2.22.2.5** `template<typename _Tp > std::indirect_array< _Tp >::indirect_array ( const indirect_array< _Tp > & __a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file indirect\_array.h.

**2.22.2.6** `template<typename _Tp > std::mask_array< _Tp >::mask_array ( const mask_array< _Tp > & a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 138 of file mask\_array.h.

**2.22.2.7** `std::slice::slice ( ) [inline, inherited]`

Construct an empty slice.

Definition at line 89 of file slice\_array.h.

**2.22.2.8** `std::slice::slice ( size_t __o, size_t __d, size_t __s ) [inline, inherited]`

Construct a slice.

#### Parameters

*o* Offset in array of first element.

*d* Number of elements in slice.

*s* Stride between array elements.

Definition at line 93 of file slice\_array.h.

**2.22.2.9** `template<typename _Tp > std::slice_array< _Tp >::slice_array ( const slice_array< _Tp > & a ) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 206 of file slice\_array.h.

**2.22.2.10** `template<typename _Tp> std::valarray< _Tp >::valarray ( const mask_array< _Tp > & __ma ) [inline, inherited]`

Construct an array with the same size and values in *ma*.

Definition at line 630 of file valarray.

**2.22.2.11** `template<typename _Tp> std::valarray<_Tp>::valarray ( const indirect_array<_Tp> & __ia ) [inline, inherited]`

Construct an array with the same size and values in *ia*.

Definition at line 639 of file valarray.

**2.22.2.12** `template<typename _Tp> std::valarray<_Tp>::valarray ( initializer_list<_Tp> __l ) [inline, inherited]`

Construct an array with an [initializer\\_list](#) of values.

Definition at line 649 of file valarray.

**2.22.2.13** `template<typename _Tp> std::valarray<_Tp>::valarray ( ) [inline, inherited]`

Construct an empty array.

Definition at line 578 of file valarray.

**2.22.2.14** `template<typename _Tp> std::valarray<_Tp>::valarray ( size_t __n ) [inline, explicit, inherited]`

Construct an array with *n* elements.

Definition at line 582 of file valarray.

**2.22.2.15** `template<typename _Tp> std::valarray<_Tp>::valarray ( const _Tp & __t, size_t __n ) [inline, inherited]`

Construct an array with *n* elements initialized to *t*.

Definition at line 588 of file valarray.

**2.22.2.16** `template<typename _Tp> std::valarray<_Tp>::valarray ( const valarray<_Tp> & __v ) [inline, inherited]`

Copy constructor.

Definition at line 603 of file valarray.

**2.22.2.17** `template<typename _Tp> std::valarray<_Tp>::valarray ( const slice_array<_Tp> & __sa ) [inline, inherited]`

Construct an array with the same size and values in *sa*.

Definition at line 610 of file valarray.

**2.22.2.18** `template<typename _Tp> std::valarray<_Tp>::valarray ( const gslice_array<_Tp> & __ga ) [inline, inherited]`

Construct an array with the same size and values in *ga*.

Definition at line 619 of file valarray.

**2.22.2.19** `std::gslice::~~gslice ( ) [inline, inherited]`

Destructor.

Definition at line 161 of file gslice.h.

**2.22.2.20** `template<class _Tp> _Expr<_ValFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply ( _Tp func_Tp ) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying *func* to the corresponding element of this array. The new array has the same size as this one.

#### Parameters

*func* Function of *Tp* returning *Tp* to apply.

#### Returns

New valarray with transformed elements.

Definition at line 972 of file valarray.

**2.22.2.21** `template<class _Tp> _Expr<_RefFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply ( _Tp funcconst_Tp & ) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying *func* to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

*func* Function of const Tp& returning Tp to apply.

**Returns**

New valarray with transformed elements.

Definition at line 980 of file valarray.

**2.22.2.22** `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::cshift  
( int __n ) const [inline, inherited]`

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is  $(i - n) \% \text{size}()$ . The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

**Parameters**

*n* Number of element positions to rotate.

**Returns**

New valarray with elements in shifted positions.

Definition at line 898 of file valarray.

**2.22.2.23** `template<typename _Tp> _Tp std::valarray<_Tp>::max ( )  
const [inline, inherited]`

Return the maximum element using operator<().

Definition at line 964 of file valarray.

References std::max\_element().

**2.22.2.24** `template<typename _Tp> _Tp std::valarray<_Tp>::min ( )  
const [inline, inherited]`

Return the minimum element using operator<().

Definition at line 956 of file valarray.

References std::min\_element().

**2.22.2.25** `template<typename _Tp > valarray< _Tp >::template _UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator! ( ) const [inline, inherited]`

Return a new valarray by applying unary ! to each element.

Definition at line 999 of file valarray.

**2.22.2.26** `template<typename _Tp > void std::gslice_array< _Tp >::operator%=( const valarray< _Tp > & __v ) const [inline, inherited]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 201 of file gslice\_array.h.

**2.22.2.27** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator%=( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to *e % t*.

Definition at line 1026 of file valarray.

**2.22.2.28** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator%=( const valarray< _Tp > & __v ) [inline, inherited]`

Modulo elements of array by corresponding elements of *v*.

Definition at line 1026 of file valarray.

**2.22.2.29** `template<typename _Tp > void std::gslice_array< _Tp >::operator&=( const valarray< _Tp > & __v ) const [inline, inherited]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice\_array.h.

**2.22.2.30** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator&=( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to *e & t*.

Definition at line 1028 of file valarray.

**2.22.2.31** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator&= ( const valarray<_Tp> & __v ) [inline, inherited]`

Logical and corresponding elements of *v* with elements of array.

Definition at line 1028 of file valarray.

**2.22.2.32** `template<typename _Tp> void std::gslice_array<_Tp>::operator*=( const valarray<_Tp> & __v ) const [inline, inherited]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 199 of file gslice\_array.h.

**2.22.2.33** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator*=( const _Tp & __t ) [inline, inherited]`

Multiply each element of array by *t*.

Definition at line 1024 of file valarray.

**2.22.2.34** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator*=( const valarray<_Tp> & __v ) [inline, inherited]`

Multiply elements of array by corresponding elements of *v*.

Definition at line 1024 of file valarray.

**2.22.2.35** `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__unary_plus>::Rt std::valarray<_Tp>::operator+( ) const [inline, inherited]`

Return a new valarray by applying unary + to each element.

Definition at line 996 of file valarray.

**2.22.2.36** `template<typename _Tp> void std::gslice_array<_Tp>::operator+=( const valarray<_Tp> & __v ) const [inline, inherited]`

Add corresponding elements of *v* to slice elements.

Definition at line 202 of file `gslice_array.h`.

**2.22.2.37** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator+=( const _Tp & __t ) [inline, inherited]`

Add  $t$  to each element of array.

Definition at line 1022 of file `valarray`.

**2.22.2.38** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator+=( const valarray<_Tp> & __v ) [inline, inherited]`

Add corresponding elements of  $v$  to elements of array.

Definition at line 1022 of file `valarray`.

**2.22.2.39** `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__negate>::_Rt std::valarray<_Tp>::operator-( ) const [inline, inherited]`

Return a new `valarray` by applying unary `-` to each element.

Definition at line 997 of file `valarray`.

**2.22.2.40** `template<typename _Tp> void std::gslice_array<_Tp>::operator-= ( const valarray<_Tp> & __v ) const [inline, inherited]`

Subtract corresponding elements of  $v$  from slice elements.

Definition at line 203 of file `gslice_array.h`.

**2.22.2.41** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator-= ( const valarray<_Tp> & __v ) [inline, inherited]`

Subtract corresponding elements of  $v$  from elements of array.

Definition at line 1023 of file `valarray`.

**2.22.2.42** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator-=( const _Tp & __t ) [inline, inherited]`

Subtract  $t$  to each element of array.

Definition at line 1023 of file valarray.

**2.22.2.43** `template<typename _Tp > void std::gslice_array< _Tp >::operator/=( const valarray< _Tp > & __v ) const [inline, inherited]`

Divide slice elements by corresponding elements of  $v$ .

Definition at line 200 of file gslice\_array.h.

**2.22.2.44** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator/=( const valarray< _Tp > & __v ) [inline, inherited]`

Divide elements of array by corresponding elements of  $v$ .

Definition at line 1025 of file valarray.

**2.22.2.45** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator/=( const _Tp & __t ) [inline, inherited]`

Divide each element of array by  $t$ .

Definition at line 1025 of file valarray.

**2.22.2.46** `template<typename _Tp > void std::gslice_array< _Tp >::operator<<=( const valarray< _Tp > & __v ) const [inline, inherited]`

Left shift slice elements by corresponding elements of  $v$ .

Definition at line 207 of file gslice\_array.h.

**2.22.2.47** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator<<=( const _Tp & __t ) [inline, inherited]`

Left shift each element  $e$  of array by  $t$  bits.

Definition at line 1030 of file valarray.

**2.22.2.48** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator<<=( const valarray< _Tp > & __v ) [inline, inherited]`

Left shift elements of array by corresponding elements of *v*.

Definition at line 1030 of file valarray.

**2.22.2.49** `template<typename _Tp > void std::slice_array< _Tp >::operator=( const valarray< _Tp > & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 228 of file slice\_array.h.

**2.22.2.50** `template<typename _Tp > void std::slice_array< _Tp >::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 223 of file slice\_array.h.

**2.22.2.51** `template<typename _Tp > void std::mask_array< _Tp >::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file mask\_array.h.

**2.22.2.52** `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp >::operator=( const valarray< _Tp > & __v ) [inline, inherited]`

Assign elements to an array.

Assign elements of array to values in *v*. Results are undefined if *v* does not have the same size as this array.

### Parameters

*v* Valarray to get values from.

Definition at line 670 of file valarray.

**2.22.2.53** `template<typename _Tp > gslice_array< _Tp > & std::gslice_array< _Tp >::operator=( const gslice_array< _Tp > & __a ) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 147 of file `gslice_array.h`.

References `std::valarray< _Tp >::size()`.

**2.22.2.54** `template<typename _Tp > void std::gslice_array< _Tp >::operator=( const valarray< _Tp > & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 165 of file `gslice_array.h`.

References `std::valarray< _Tp >::size()`.

**2.22.2.55** `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp >::operator=( const _Tp & __t ) [inline, inherited]`

Assign elements to a value.

Assign all elements of array to *t*.

#### Parameters

*t* Value for elements.

Definition at line 718 of file `valarray`.

**2.22.2.56** `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp >::operator=( const slice_array< _Tp > & __sa ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

#### Parameters

*sa* Array slice to get values from.

Definition at line 726 of file `valarray`.

**2.22.2.57** `template<typename _Tp > void std::gslice_array<_Tp >::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file `gslice_array.h`.

References `std::valarray<_Tp >::size()`.

**2.22.2.58** `template<typename _Tp> valarray<_Tp > & std::valarray<_Tp >::operator=( const indirect_array<_Tp > & __ia ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

#### Parameters

*ia* Array slice to get values from.

Definition at line 756 of file `valarray`.

**2.22.2.59** `gslice & std::gslice::operator=( const gslice & __g ) [inline, inherited]`

Assignment operator.

Definition at line 168 of file `gslice.h`.

**2.22.2.60** `template<typename _Tp > indirect_array<_Tp > & std::indirect_array<_Tp >::operator=( const indirect_array<_Tp > & __a ) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 153 of file `indirect_array.h`.

**2.22.2.61** `template<typename _Tp > void std::indirect_array<_Tp >::operator=( const valarray<_Tp > & __v ) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 167 of file `indirect_array.h`.

```
2.22.2.62 template<typename _Tp > slice_array< _Tp > & std::slice_array<
    _Tp >::operator= ( const slice_array< _Tp > & __a ) [inline,
    inherited]
```

Assignment operator. Assigns slice elements to corresponding *///* elements of *a*.

Definition at line 214 of file slice\_array.h.

```
2.22.2.63 template<typename _Tp > mask_array< _Tp > &
    std::mask_array< _Tp >::operator= ( const mask_array< _Tp > &
    __a ) [inline, inherited]
```

Assignment operator. Assigns elements to corresponding elements *///* of *a*.

Definition at line 148 of file mask\_array.h.

```
2.22.2.64 template<typename _Tp > valarray< _Tp > & std::valarray<
    _Tp >::operator= ( initializer_list< _Tp > __l ) [inline,
    inherited]
```

Assign elements to an [initializer\\_list](#).

Assign elements of array to values in *l*. Results are undefined if *l* does not have the same size as this array.

#### Parameters

*l* [initializer\\_list](#) to get values from.

Definition at line 694 of file valarray.

```
2.22.2.65 template<typename _Tp > valarray< _Tp > & std::valarray< _Tp
    >::operator= ( const gsllice_array< _Tp > & __ga ) [inline,
    inherited]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

#### Parameters

*ga* Array slice to get values from.

Definition at line 736 of file valarray.

References `std::valarray< _Tp >::size()`.

**2.22.2.66** `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=( const mask_array<_Tp> & __ma ) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

#### Parameters

*ma* Array slice to get values from.

Definition at line 746 of file valarray.

**2.22.2.67** `template<typename _Tp> void std::indirect_array<_Tp>::operator=( const _Tp & __t ) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 162 of file indirect\_array.h.

**2.22.2.68** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>=( const valarray<_Tp> & __v ) [inline, inherited]`

Right shift elements of array by corresponding elements of *v*.

Definition at line 1031 of file valarray.

**2.22.2.69** `template<typename _Tp> void std::gslice_array<_Tp>::operator>>=( const valarray<_Tp> & __v ) const [inline, inherited]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 208 of file gslice\_array.h.

**2.22.2.70** `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>=( const _Tp & __t ) [inline, inherited]`

Right shift each element *e* of array by *t* bits.

Definition at line 1031 of file valarray.

**2.22.2.71** `template<typename _Tp > indirect_array< _Tp > std::valarray< _Tp >::operator[] ( const valarray< size_t > & __i ) [inline, inherited]`

Return a reference to an array subset.

Returns an [indirect\\_array](#) referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned [indirect\\_array](#) refers to these elements.

#### Parameters

*i* The valarray element index list.

#### Returns

Indirect\_array referencing elements in *i*.

Definition at line 836 of file valarray.

References `std::valarray< _Tp >::size()`.

**2.22.2.72** `template<typename _Tp > slice_array< _Tp > std::valarray< _Tp >::operator[] ( slice __s ) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

#### See also

[slice](#).

#### Parameters

*s* The source slice.

#### Returns

New valarray containing elements in *s*.

Definition at line 783 of file valarray.

**2.22.2.73** `template<typename _Tp > mask_array< _Tp > std::valarray< _Tp >::operator[] ( const valarray< bool > & __m ) [inline, inherited]`

Return a reference to an array subset.

Returns a new [mask\\_array](#) referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

#### Parameters

*m* The valarray bitmask.

#### Returns

New valarray containing elements indicated by *m*.

Definition at line 817 of file valarray.

References `std::valarray<_Tp>::size()`.

**2.22.2.74** `template<typename _Tp > valarray<_Tp > std::valarray<_Tp >::operator[] ( const valarray<bool > & __m ) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

#### Parameters

*m* The valarray bitmask.

#### Returns

New valarray containing elements indicated by *m*.

Definition at line 805 of file valarray.

References `std::valarray<_Tp>::size()`.

**2.22.2.75** `template<typename _Tp > _Expr<_GClos<_ValArray, _Tp >, _Tp > std::valarray<_Tp >::operator[] ( const gslice & __gs ) const [inline, inherited]`

Return an array subset.

Returns a [slice\\_array](#) referencing the elements of the array indicated by the slice argument.

**See also**

[gslice](#).

**Parameters**

*s* The source slice.

**Returns**

Slice\_array referencing elements indicated by *s*.

Definition at line 788 of file valarray.

**2.22.2.76** `template<typename _Tp > _Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] ( slice __s ) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

[slice](#).

**Parameters**

*s* The source slice.

**Returns**

New valarray containing elements in *s*.

Definition at line 775 of file valarray.

**2.22.2.77** `template<typename _Tp > _Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] ( const valarray< size_t > & __i ) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

**Parameters**

*i* The valarray element index list.

**Returns**

New valarray containing elements in *s*.

Definition at line 828 of file valarray.

**2.22.2.78** `template<typename _Tp > _Tp & std::valarray< _Tp >::operator[] ( size_t __i ) [inline, inherited]`

Return a reference to the *i*'th array element.

**Parameters**

*i* Index of element to return.

**Returns**

Reference to the *i*'th element.

Definition at line 552 of file valarray.

**2.22.2.79** `template<typename _Tp > gslice_array< _Tp > std::valarray< _Tp >::operator[] ( const gslice & __gs ) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the `gslice` argument. The new valarray has the same size as the input `gslice`.

**See also**

[gslice](#).

**Parameters**

*s* The source `gslice`.

**Returns**

New valarray containing elements in *s*.

Definition at line 797 of file valarray.

**2.22.2.80** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator^=( const _Tp & __t ) [inline, inherited]`

Set each element *e* of array to  $e \wedge t$ .

Definition at line 1027 of file valarray.

**2.22.2.81** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator^=( const valarray< _Tp > & __v ) const [inline,  
inherited]`

Logical xor slice elements with corresponding elements of  $v$ .

Definition at line 204 of file `gslice_array.h`.

**2.22.2.82** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator^=( const valarray< _Tp > & __v ) [inline,  
inherited]`

Logical xor corresponding elements of  $v$  with elements of array.

Definition at line 1027 of file `valarray`.

**2.22.2.83** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator|=( const _Tp & __t ) [inline, inherited]`

Set each element  $e$  of array to  $e | t$ .

Definition at line 1029 of file `valarray`.

**2.22.2.84** `template<typename _Tp > void std::gslice_array< _Tp  
>::operator|=( const valarray< _Tp > & __v ) const [inline,  
inherited]`

Logical or slice elements with corresponding elements of  $v$ .

Definition at line 206 of file `gslice_array.h`.

**2.22.2.85** `template<class _Tp> valarray< _Tp > & std::valarray< _Tp  
>::operator|=( const valarray< _Tp > & __v ) [inline,  
inherited]`

Logical or corresponding elements of  $v$  with elements of array.

Definition at line 1029 of file `valarray`.

**2.22.2.86** `template<typename _Tp > valarray< _Tp >::template _UnaryOp<  
__bitwise_not >::Rt std::valarray< _Tp >::operator~ ( ) const  
[inline, inherited]`

Return a new valarray by applying unary  $\sim$  to each element.

Definition at line 998 of file valarray.

**2.22.2.87** `template<class _Tp> void std::valarray<_Tp>::resize ( size_t  
__size, _Tp __c = _Tp() ) [inline, inherited]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

#### Parameters

- size* New array size.
- c* New value for all elements.

Definition at line 939 of file valarray.

**2.22.2.88** `template<class _Tp > valarray<_Tp > std::valarray<_Tp >::shift  
( int __n ) const [inline, inherited]`

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

#### Parameters

- n* Number of element positions to shift.

#### Returns

- New valarray with elements in shifted positions.

Definition at line 857 of file valarray.

**2.22.2.89** `template<class _Tp > size_t std::valarray<_Tp >::size ( ) const  
[inline, inherited]`

Return the number of elements in array.

Definition at line 844 of file valarray.

Referenced by `std::valarray<_Tp >::operator=()`, `std::gslice_array<_Tp >::operator=()`, and `std::valarray<_Tp >::operator[]()`.

**2.22.2.90** `size_t std::slice::size ( ) const [inline, inherited]`

Return size of slice.

Definition at line 101 of file slice\_array.h.

**2.22.2.91** `valarray< size_t > std::gslice::size ( ) const [inline, inherited]`

Return array of sizes of slice dimensions.

Definition at line 137 of file gslice.h.

**2.22.2.92** `size_t std::slice::start ( ) const [inline, inherited]`

Return array offset of first slice element.

Definition at line 97 of file slice\_array.h.

**2.22.2.93** `size_t std::gslice::start ( ) const [inline, inherited]`

Return array offset of first slice element.

Definition at line 133 of file gslice.h.

**2.22.2.94** `valarray< size_t > std::gslice::stride ( ) const [inline, inherited]`

Return array of array strides for each dimension.

Definition at line 141 of file gslice.h.

**2.22.2.95** `size_t std::slice::stride ( ) const [inline, inherited]`

Return array stride of slice.

Definition at line 105 of file slice\_array.h.

**2.22.2.96** `template<class _Tp > _Tp std::valarray< _Tp >::sum ( ) const [inline, inherited]`

Return the sum of all elements in the array.

Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

Definition at line 849 of file valarray.

## 2.23 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_laguerre` (unsigned int `__n`, unsigned int `__m`, `_Tp __x`)
- `float std::tr1::assoc\_laguerref` (unsigned int `__n`, unsigned int `__m`, float `__x`)
- `long double std::tr1::assoc\_laguerrel` (unsigned int `__n`, unsigned int `__m`, long double `__x`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre` (unsigned int `__l`, unsigned int `__m`, `_Tp __x`)
- `float std::tr1::assoc\_legendref` (unsigned int `__l`, unsigned int `__m`, float `__x`)
- `long double std::tr1::assoc\_legendrel` (unsigned int `__l`, unsigned int `__m`, long double `__x`)
- `template<typename _Tpx , typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (`_Tpx __x`, `_Tpy __y`)
- `float std::tr1::betaf` (float `__x`, float `__y`)
- `long double std::tr1::betal` (long double `__x`, long double `__y`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1` (`_Tp __k`)
- `float std::tr1::comp\_ellint\_1f` (float `__k`)
- `long double std::tr1::comp\_ellint\_1l` (long double `__k`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2` (`_Tp __k`)
- `float std::tr1::comp\_ellint\_2f` (float `__k`)

- long double **std::tr1::comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **std::tr1::comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **std::tr1::comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **std::tr1::comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **std::tr1::conf\_hyperg** (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **std::tr1::conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)

- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint\_3 (_Tp`  
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint\_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint\_3l (long double __k, long double __nu, long double`  
`__phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`  
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`  
`(_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double`  
`__c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::laguerref (unsigned int __n, float __x)`
- `long double std::tr1::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::legendref (unsigned int __n, float __x)`
- `long double std::tr1::legendrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann\_zeta (_Tp __x)`
- `float std::tr1::riemann\_zetaf (float __x)`
- `long double std::tr1::riemann\_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_bessel (unsigned int __n,`  
`_Tp __x)`
- `float std::tr1::sph\_besself (unsigned int __n, float __x)`
- `long double std::tr1::sph\_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_legendre (unsigned int _-`  
`l, unsigned int __m, _Tp __theta)`

- float `std::tr1::sph_legendref` (unsigned int `__l`, unsigned int `__m`, float `__theta`)
- long double `std::tr1::sph_legendrel` (unsigned int `__l`, unsigned int `__m`, long double `__theta`)
- `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann` (unsigned int `__n`, `_Tp __x`)
- float `std::tr1::sph_neumannf` (unsigned int `__n`, float `__x`)
- long double `std::tr1::sph_neumannl` (unsigned int `__n`, long double `__x`)

### 2.23.1 Detailed Description

A collection of advanced mathematical special functions.

### 2.23.2 Function Documentation

**2.23.2.1** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_laguerre ( unsigned int __n, unsigned int __m, _Tp __x ) [inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 123 of file `tr1/cmath`.

**2.23.2.2** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_legendre ( unsigned int __l, unsigned int __m, _Tp __x ) [inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 140 of file `tr1/cmath`.

**2.23.2.3** `template<typename _Tpx , typename _Tpy > __gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta ( _Tpx __x, _Tpy __y ) [inline]`

5.2.1.3 Beta functions.

Definition at line 157 of file `tr1/cmath`.

**2.23.2.4** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::comp_ellint_1( _Tp __k ) [inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 174 of file tr1/cmath.

**2.23.2.5** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::comp_ellint_2( _Tp __k ) [inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 191 of file tr1/cmath.

**2.23.2.6** `template<typename _Tp , typename _Tpn > __gnu_cxx::__promote_  
2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3( _Tp __k, _Tpn  
__nu ) [inline]`

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 208 of file tr1/cmath.

**2.23.2.7** `template<typename _Tpa , typename _Tpc , typename _Tp  
> __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type  
std::tr1::conf_hyperg( _Tpa __a, _Tpc __c, _Tp __x ) [inline]`

5.2.1.7 Confluent hypergeometric functions.

Definition at line 225 of file tr1/cmath.

**2.23.2.8** `template<typename _Tpnu , typename _Tp >  
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j  
( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 242 of file tr1/cmath.

**2.23.2.9** `template<typename _Tpnu , typename _Tp >  
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j  
( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 259 of file tr1/cmath.

**2.23.2.10** `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`  
`std::tr1::cyl_bessel_k ( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 276 of file tr1/cmath.

**2.23.2.11** `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`  
`std::tr1::cyl_neumann ( _Tpnu __nu, _Tp __x ) [inline]`

5.2.1.11 Cylindrical Neumann functions.

Definition at line 293 of file tr1/cmath.

**2.23.2.12** `template<typename _Tp , typename _Tpp >`  
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type` `std::tr1::ellint_1 (`  
`_Tp __k, _Tpp __phi ) [inline]`

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 310 of file tr1/cmath.

**2.23.2.13** `template<typename _Tp , typename _Tpp >`  
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type` `std::tr1::ellint_2 (`  
`_Tp __k, _Tpp __phi ) [inline]`

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 327 of file tr1/cmath.

**2.23.2.14** `template<typename _Tp , typename _Tpn , typename _Tpp`  
`> __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type`  
`std::tr1::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi ) [inline]`

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 344 of file tr1/cmath.

**2.23.2.15** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type`  
`std::tr1::expint ( _Tp __x ) [inline]`

5.2.1.15 Exponential integrals.

Definition at line 361 of file tr1/cmath.

**2.23.2.16** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::hermite ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.16 Hermite polynomials.

Definition at line 378 of file tr1/cmath.

**2.23.2.17** `template<typename _Tpa , typename _Tpb , typename _Tpc ,  
typename _Tp > __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc,  
_Tp>::__type std::tr1::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c,  
_Tp __x ) [inline]`

5.2.1.17 Hypergeometric functions.

Definition at line 395 of file tr1/cmath.

**2.23.2.18** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::laguerre ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.18 Laguerre polynomials.

Definition at line 412 of file tr1/cmath.

**2.23.2.19** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::legendre ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.19 Legendre polynomials.

Definition at line 429 of file tr1/cmath.

**2.23.2.20** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::riemann_zeta ( _Tp __x ) [inline]`

5.2.1.20 Riemann zeta function.

Definition at line 446 of file tr1/cmath.

**2.23.2.21** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_bessel ( unsigned int __n, _Tp __x ) [inline]`

5.2.1.21 Spherical Bessel functions.

Definition at line 463 of file tr1/cmath.

**2.23.2.22** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_legendre ( unsigned int _l, unsigned int _m, _Tp  
_theta ) [inline]`

5.2.1.22 Spherical associated Legendre functions.

Definition at line 480 of file tr1/cmath.

**2.23.2.23** `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type  
std::tr1::sph_neumann ( unsigned int _n, _Tp _x ) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 497 of file tr1/cmath.

## 2.24 Type Traits

Collaboration diagram for Type Traits:



### Classes

- struct `std::add_lvalue_reference< _Tp >`  
*add\_lvalue\_reference*
- struct `std::add_rvalue_reference< _Tp >`  
*add\_rvalue\_reference*
- struct `std::aligned_storage< _Len, _Align >`  
*Alignment type.*
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`  
*conditional*

- class `std::decay< _Tp >`  
*decay*
- struct `std::enable_if< bool, _Tp >`  
*enable\_if*
- struct `std::has_nothrow_assign< _Tp >`  
*has\_nothrow\_assign*
- struct `std::has_nothrow_copy_constructor< _Tp >`  
*has\_nothrow\_copy\_constructor*
- struct `std::has_nothrow_default_constructor< _Tp >`  
*has\_nothrow\_default\_constructor*
- struct `std::has_trivial_assign< _Tp >`  
*has\_trivial\_assign*
- struct `std::has_trivial_copy_constructor< _Tp >`  
*has\_trivial\_copy\_constructor*
- struct `std::has_trivial_default_constructor< _Tp >`  
*has\_trivial\_default\_constructor*
- struct `std::has_trivial_destructor< _Tp >`  
*has\_trivial\_destructor*
- struct `std::is_base_of< _Base, _Derived >`  
*is\_base\_of*
- struct `std::is_constructible< _Tp, _Args >`  
*is\_constructible*
- struct `std::is_convertible< _From, _To >`  
*is\_convertible*
- struct `std::is_explicitly_convertible< _From, _To >`  
*is\_explicitly\_convertible*
- struct `std::is_lvalue_reference< typename >`  
*is\_lvalue\_reference*

- struct `std::is_pod< _Tp >`  
*is\_pod*
- struct `std::is_reference< _Tp >`  
*is\_reference*
- struct `std::is_rvalue_reference< typename >`  
*is\_rvalue\_reference*
- struct `std::is_signed< _Tp >`  
*is\_signed*
- struct `std::is_standard_layout< _Tp >`  
*is\_standard\_layout*
- struct `std::is_trivial< _Tp >`  
*is\_trivial*
- struct `std::is_unsigned< _Tp >`  
*is\_unsigned*
- struct `std::make_signed< _Tp >`  
*make\_signed*
- struct `std::make_unsigned< _Tp >`  
*make\_unsigned*
- struct `std::remove_reference< _Tp >`  
*remove\_reference*
- struct `std::tr1::__is_member_pointer_helper< _Tp >`  
*is\_member\_pointer*
- struct `std::tr1::add_const< _Tp >`  
*add\_const*
- struct `std::tr1::add_cv< _Tp >`  
*add\_cv*
- struct `std::tr1::add_pointer< _Tp >`  
*add\_pointer*

- struct `std::tr1::add_volatile< _Tp >`  
*add\_volatile*
- struct `std::tr1::alignment_of< _Tp >`  
*alignment\_of*
- struct `std::tr1::extent< typename, _Uint >`  
*extent*
- struct `std::tr1::has_virtual_destructor< _Tp >`  
*has\_virtual\_destructor*
- struct `std::tr1::integral_constant< _Tp, __v >`  
*integral\_constant*
- struct `std::tr1::is_abstract< _Tp >`  
*is\_abstract*
- struct `std::tr1::is_arithmetic< _Tp >`  
*is\_arithmetic*
- struct `std::tr1::is_array< typename >`  
*is\_array*
- struct `std::tr1::is_class< _Tp >`  
*is\_class*
- struct `std::tr1::is_compound< _Tp >`  
*is\_compound*
- struct `std::tr1::is_const< typename >`  
*is\_const*
- struct `std::tr1::is_empty< _Tp >`  
*is\_empty*
- struct `std::tr1::is_enum< _Tp >`  
*is\_enum*
- struct `std::tr1::is_floating_point< _Tp >`  
*is\_floating\_point*

- struct `std::tr1::is_function< typename >`  
*is\_function*
- struct `std::tr1::is_fundamental< _Tp >`  
*is\_fundamental*
- struct `std::tr1::is_integral< _Tp >`  
*is\_integral*
- struct `std::tr1::is_member_function_pointer< _Tp >`  
*is\_member\_function\_pointer*
- struct `std::tr1::is_member_object_pointer< _Tp >`  
*is\_member\_object\_pointer*
- struct `std::tr1::is_object< _Tp >`  
*is\_object*
- struct `std::tr1::is_pointer< _Tp >`  
*is\_pointer*
- struct `std::tr1::is_polymorphic< _Tp >`  
*is\_polymorphic*
- struct `std::tr1::is_same< typename, typename >`  
*is\_same*
- struct `std::tr1::is_scalar< _Tp >`  
*is\_scalar*
- struct `std::tr1::is_union< _Tp >`  
*is\_union*
- struct `std::tr1::is_void< _Tp >`  
*is\_void*
- struct `std::tr1::is_volatile< typename >`  
*is\_volatile*
- struct `std::tr1::rank< typename >`  
*rank*

- struct `std::tr1::remove_all_extents< _Tp >`  
*remove\_all\_extents*
- struct `std::tr1::remove_const< _Tp >`  
*remove\_const*
- struct `std::tr1::remove_cv< _Tp >`  
*remove\_cv*
- struct `std::tr1::remove_extent< _Tp >`  
*remove\_extent*
- struct `std::tr1::remove_pointer< _Tp >`  
*remove\_pointer*
- struct `std::tr1::remove_volatile< _Tp >`  
*remove\_volatile*

## Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`

## Typedefs

- `typedef integral_constant< bool, false > std::tr1::false_type`
- `typedef integral_constant< bool, true > std::tr1::true_type`

## Functions

- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type std::declval ()`

## Variables

- `static const _Tp std::tr1::integral_constant::value`

### 2.24.1 Detailed Description

Compile time type transformation and information.

### 2.24.2 Typedef Documentation

#### 2.24.2.1 `typedef integral_constant<bool, false> std::tr1::false_type`

typedef for `false_type`

Definition at line 78 of file `tr1_impl/type_traits`.

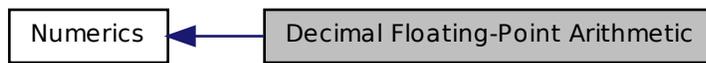
#### 2.24.2.2 `typedef integral_constant<bool, true> std::tr1::true_type`

typedef for `true_type`

Definition at line 75 of file `tr1_impl/type_traits`.

## 2.25 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



### Namespaces

- namespace [std::decimal](#)

### 2.25.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

## 2.26 Binder Classes

Collaboration diagram for Binder Classes:



### Classes

- class [std::binder1st< \\_Operation >](#)  
*One of the binder functors.*
- class [std::binder2nd< \\_Operation >](#)  
*One of the binder functors.*
- struct [std::is\\_bind\\_expression< \\_Tp >](#)  
*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].*
- struct [std::is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [std::is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [std::is\\_placeholder< \\_Tp >](#)  
*Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct [std::is\\_placeholder< \\_Placeholder< \\_Num > >](#)

### Namespaces

- namespace [std::placeholders](#)

## Functions

- `template<typename _Functor, typename... _ArgTypes>  
_Bind< typename _Maybe_wrap_member_pointer< _Functor >::type(_-  
ArgTypes...)> std::bind (_Functor __f, _ArgTypes... __args)`
- `template<typename _Operation, typename _Tp >  
binder1st< _Operation > std::binder1st (const _Operation &__fn, const _Tp &_  
_x)`
- `template<typename _Operation, typename _Tp >  
binder2nd< _Operation > std::binder2nd (const _Operation &__fn, const _Tp &_  
_x)`

## Variables

- `std::binder1st std::\_GLIBCXX\_DEPRECATED\_ATTR`

### 2.26.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B`'s `operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `binder1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>, 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `binder1st` had been used, the functor would perform `1.3 - x` instead.)

Creator-wrapper functions like `binder1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `binder1st(std::plus<int>, 5)`.)

These become more useful when combined with the composition functions.

## 2.26.2 Function Documentation

**2.26.2.1** `template<typename _Function , typename... _ArgTypes>  
_Bind<typename _Maybe_wrap_member_pointer<_  
_Function>::type(_ArgTypes...)> std::bind ( _Function __f,  
_ArgTypes... __args ) [inline]`

Function template for `std::bind`.

Definition at line 1384 of file `functional`.

**2.26.2.2** `template<typename _Operation , typename _Tp >  
binder1st<_Operation> std::bind1st ( const _Operation & __fn,  
const _Tp & __x ) [inline]`

One of the [binder functions](#).

Definition at line 125 of file `binders.h`.

**2.26.2.3** `template<typename _Operation , typename _Tp >  
binder2nd<_Operation> std::bind2nd ( const _Operation & __fn,  
const _Tp & __x ) [inline]`

One of the [binder functions](#).

Definition at line 160 of file `binders.h`.

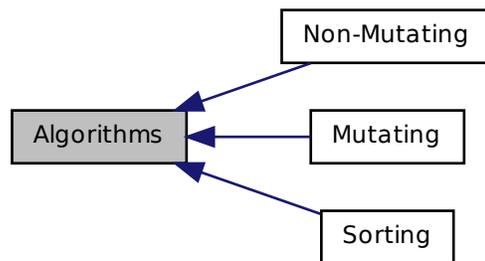
## 2.26.3 Variable Documentation

**2.26.3.1** `std::binder2nd std::_GLIBCXX_DEPRECATED_ATTR`

One of the [binder functions](#).

## 2.27 Algorithms

Collaboration diagram for Algorithms:



### Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

#### 2.27.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

## 2.28 Mutating

Collaboration diagram for Mutating:



### Functions

- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator`  
`__result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`  
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`  
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`  
`__gen)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`

- `template<typename _Tp >`  
`std::remove_reference< _Tp >::type && std::move ( _Tp &&__t)`
- `template<typename _BI1 , typename _BI2 >`  
`_BI2 std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::partition ( _ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , type-`  
`name _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy ( _InputIterator`  
`__first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`  
`out_false, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::partition_point ( _ForwardIterator __first, _ForwardIterator`  
`__last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::random_shuffle ( _RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _RandomNumberGenerator >`  
`void std::random_shuffle ( _RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator , typename _Tp >`  
`_ForwardIterator std::remove ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Tp >`  
`_OutputIterator std::remove_copy ( _InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator std::remove_copy_if ( _InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`_ForwardIterator std::remove_if ( _ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Tp >`  
`void std::replace ( _ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _-`  
`Tp >`  
`_OutputIterator std::replace_copy_if ( _InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator , typename _Predicate , typename _Tp >`  
`void std::replace_if ( _ForwardIterator __first, _ForwardIterator __last, _-`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`  
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __-`  
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _UniformRandomNumberGenerator &__g)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _-`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`  
`op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`  
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred)`

## 2.28.1 Function Documentation

**2.28.1.1** `template<typename _II, typename _OI > _OI std::copy ( _II __first, _II __last, _OI __result ) [inline]`

Copies the range [first,last) into result.

### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

### Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 464 of file `stl_algobase.h`.

**2.28.1.2** `template<typename _BI1, typename _BI2 > _BI2 std::copy_backward ( _BI1 __first, _BI1 __last, _BI2 __result ) [inline]`

Copies the range [first,last) into result.

### Parameters

*first* A bidirectional iterator.

*last* A bidirectional iterator.

*result* A bidirectional iterator.

### Returns

result - (first - last)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are

passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use copy instead. Note that the start of the output range may overlap [first,last).

Definition at line 633 of file stl\_algobase.h.

**2.28.1.3** `template<typename _InputIterator , typename _OutputIterator ,  
typename _Predicate > _OutputIterator std::copy_if ( _InputIterator  
__first, _InputIterator __last, _OutputIterator __result, _Predicate  
__pred )`

Copy the elements of a sequence for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) for which `pred` returns true to the range beginning at `result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 949 of file stl\_algo.h.

**2.28.1.4** `template<typename _InputIterator , typename _Size , typename  
_OutputIterator > _OutputIterator std::copy_n ( _InputIterator  
__first, _Size __n, _OutputIterator __result ) [inline]`

Copies the range [first,first+n) into [result,result+n).

#### Parameters

*first* An input iterator.

*n* The number of elements to copy.

*result* An output iterator.

**Returns**

result+n.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1006 of file `stl_algo.h`.

References `std::__iterator_category()`.

**2.28.1.5** `template<typename _ForwardIterator , typename _Tp > void std::fill  
( _ForwardIterator __first, _ForwardIterator __last, const _Tp &  
__value ) [inline]`

Fills the range `[first,last)` with copies of `value`.

**Parameters**

*first* A forward iterator.

*last* A forward iterator.

*value* A reference-to-const of arbitrary type.

**Returns**

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 735 of file `stl_algobase.h`.

**2.28.1.6** `template<typename _OI , typename _Size , typename _Tp > _OI  
std::fill_n( _OI __first, _Size __n, const _Tp & __value )  
[inline]`

Fills the range `[first,first+n)` with copies of `value`.

**Parameters**

*first* An output iterator.

*n* The count of copies to perform.

*value* A reference-to-const of arbitrary type.

**Returns**

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 793 of file `stl_algobase.h`.

**2.28.1.7** `template<typename _ForwardIterator , typename _Generator > void  
std::generate ( _ForwardIterator __first, _ForwardIterator __last,  
_Generator __gen )`

Assign the result of a function object to each value in a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*gen* A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

#### Returns

`generate()` returns no value.

Performs the assignment `*i = gen ()` for each `i` in the range `[first,last)`.

Definition at line 4809 of file `stl_algo.h`.

**2.28.1.8** `template<typename _OutputIterator , typename _Size , typename  
_Generator > _OutputIterator std::generate_n ( _OutputIterator  
__first, _Size __n, _Generator __gen )`

Assign the result of a function object to each value in a sequence.

#### Parameters

*first* A forward iterator.

*n* The length of the sequence.

*gen* A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

#### Returns

The end of the sequence, `first+n`

Performs the assignment `*i = gen()` for each `i` in the range `[first,first+n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 4840 of file `stl_algo.h`.

**2.28.1.9** `template<typename _InputIterator , typename _Predicate > bool  
std::is_partitioned ( _InputIterator __first, _InputIterator __last,  
_Predicate __pred ) [inline]`

Checks whether the sequence is partitioned.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

True if the range `[first,last)` is partitioned by `pred`, i.e. if all elements that satisfy `pred` appear before those that do not.

Definition at line 801 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

**2.28.1.10** `template<typename _ForwardIterator1 , typename  
_ForwardIterator2 > void std::iter_swap ( _ForwardIterator1 __a,  
_ForwardIterator2 __b ) [inline]`

Swaps the contents of two iterators.

#### Parameters

*a* An iterator.

*b* Another iterator.

#### Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 117 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

**2.28.1.11** `template<typename _II, typename _OI > _OI std::move ( _II __first, _II __last, _OI __result ) [inline]`

Moves the range `[first,last)` into `result`.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

#### Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 497 of file `stl_algobase.h`.

**2.28.1.12** `template<typename _Tp > std::remove_reference<_Tp>::type&& std::move ( _Tp && __t ) [inline]`

Move a value.

#### Parameters

*\_\_t* A thing of arbitrary type.

#### Returns

Same, moved.

Definition at line 81 of file `move.h`.

```
2.28.1.13 template<typename _BI1 , typename _BI2 > _BI2
std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result )
[inline]
```

Moves the range [first,last) into result.

#### Parameters

*first* A bidirectional iterator.  
*last* A bidirectional iterator.  
*result* A bidirectional iterator.

#### Returns

result - (first - last)

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use `move` instead. Note that the start of the output range may overlap [first,last).

Definition at line 669 of file `stl_algobase.h`.

```
2.28.1.14 template<typename _ForwardIterator , typename _Predicate
> _ForwardIterator std::partition ( _ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence.

#### Parameters

*first* A forward iterator.  
*last* A forward iterator.  
*pred* A predicate functor.

#### Returns

An iterator `middle` such that `pred(i)` is true for each iterator `i` in the range [first,middle) and false for each `i` in the range [middle,last).

`pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5011 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__partition()`.

**2.28.1.15** `template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate > pair<_OutputIterator1, _OutputIterator2> std::partition_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred )`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*out\_true* An output iterator.  
*out\_false* An output iterator.  
*pred* A predicate.

#### Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range [first,last) for which `pred` returns true to the range beginning at `out_true` and each element for which `pred` returns false to `out_false`.

Definition at line 1035 of file `stl_algo.h`.

**2.28.1.16** `template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::partition_point ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred )`

Find the partition point of a partitioned range.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*pred* A predicate.

#### Returns

An iterator `mid` such that `all_of(first, mid, pred)` and `none_of(mid, last, pred)` are both true.

Definition at line 819 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

```
2.28.1.17 template<typename _RandomAccessIterator > void  
std::random_shuffle ( _RandomAccessIterator __first,  
                      _RandomAccessIterator __last ) [inline]
```

Randomly shuffle the elements of a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

Nothing.

Reorder the elements in the range [first,last) using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 4947 of file stl\_algo.h.

References std::iter\_swap().

```
2.28.1.18 template<typename _RandomAccessIterator , typename  
                  _RandomNumberGenerator > void std::random_shuffle (  
                  _RandomAccessIterator __first, _RandomAccessIterator __last,  
                  _RandomNumberGenerator && __rand )
```

Shuffle the elements of a sequence using a random number generator.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*rand* The RNG functor or function.

#### Returns

Nothing.

Reorders the elements in the range [first,last) using *rand* to provide a random distribution. Calling *rand*(N) for a positive integer N should return a randomly chosen integer from the range [0,N).

Definition at line 4975 of file stl\_algo.h.

References std::iter\_swap().

**2.28.1.19** `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator std::remove ( _ForwardIterator __first,  
_ForwardIterator __last, const _Tp & __value )`

Remove elements from a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*value* The value to be removed.

#### Returns

An iterator designating the end of the resulting sequence.

All elements equal to `value` are removed from the range `[first,last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1084 of file `stl_algo.h`.

**2.28.1.20** `template<typename _InputIterator, typename _OutputIterator,  
typename _Tp > _OutputIterator std::remove_copy ( _InputIterator  
__first, _InputIterator __last, _OutputIterator __result, const _Tp  
& __value )`

Copy a sequence, removing elements of a given value.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*value* The value to be removed.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` not equal to `value` to the range beginning at `result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 872 of file `stl_algo.h`.

```
2.28.1.21  template<typename _InputIterator , typename _OutputIterator
            , typename _Predicate > _OutputIterator std::remove_copy_if (
            _InputIterator __first, _InputIterator __last, _OutputIterator
            __result, _Predicate __pred )
```

Copy a sequence, removing elements for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) for which `pred` returns false to the range beginning at `result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 910 of file `stl_algo.h`.

```
2.28.1.22  template<typename _ForwardIterator , typename _Predicate >
            _ForwardIterator std::remove_if ( _ForwardIterator __first,
            _ForwardIterator __last, _Predicate __pred )
```

Remove elements from a sequence using a predicate.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate.

#### Returns

An iterator designating the end of the resulting sequence.

All elements for which `pred` returns true are removed from the range [first,last).

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1127 of file `stl_algo.h`.

**2.28.1.23** `template<typename _ForwardIterator, typename _Tp > void  
std::replace ( _ForwardIterator __first, _ForwardIterator __last,  
const _Tp & __old_value, const _Tp & __new_value )`

Replace each occurrence of one value in a sequence with another value.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*old\_value* The value to be replaced.

*new\_value* The replacement value.

#### Returns

`replace()` returns no value.

For each iterator `i` in the range `[first,last)` if `*i == old_value` then the assignment `*i = new_value` is performed.

Definition at line 4745 of file `stl_algo.h`.

**2.28.1.24** `template<typename _InputIterator, typename _OutputIterator  
, typename _Predicate, typename _Tp > _OutputIterator  
std::replace_copy_if ( _InputIterator __first, _InputIterator  
__last, _OutputIterator __result, _Predicate __pred, const _Tp &  
__new_value )`

Copy a sequence, replacing each value for which a predicate returns true with another value.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*pred* A predicate.

*new\_value* The replacement value.

**Returns**

The end of the output sequence, `result+(last-first)`.

Copies each element in the range `[first,last)` to the range `[result,result+(last-first))` replacing elements for which `pred` returns true with `new_value`.

Definition at line 3786 of file `stl_algo.h`.

```
2.28.1.25 template<typename _ForwardIterator , typename _Predicate ,  
                typename _Tp > void std::replace_if ( _ForwardIterator __first,  
                _ForwardIterator __last, _Predicate __pred, const _Tp &  
                __new_value )
```

Replace each value in a sequence for which a predicate returns true with another value.

**Parameters**

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate.

*new\_value* The replacement value.

**Returns**

`replace_if()` returns no value.

For each iterator `i` in the range `[first,last)` if `pred(*i)` is true then the assignment `*i = new_value` is performed.

Definition at line 4777 of file `stl_algo.h`.

```
2.28.1.26 template<typename _BidirectionalIterator > void std::reverse (  
                _BidirectionalIterator __first, _BidirectionalIterator __last )  
                [inline]
```

Reverse a sequence.

**Parameters**

*first* A bidirectional iterator.

*last* A bidirectional iterator.

**Returns**

`reverse()` returns no value.

Reverses the order of the elements in the range  $[first, last)$ , so that the first element becomes the last etc. For every  $i$  such that  $0 \leq i < (last - first) / 2$ , `reverse()` swaps  $*(first + i)$  and  $*(last - (i + 1))$ .

Definition at line 1435 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

**2.28.1.27** `template<typename _BidirectionalIterator, typename  
_OutputIterator > _OutputIterator std::reverse_copy (  
_BidirectionalIterator __first, _BidirectionalIterator __last,  
_OutputIterator __result )`

Copy a sequence, reversing its elements.

#### Parameters

*first* A bidirectional iterator.

*last* A bidirectional iterator.

*result* An output iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range  $[first, last)$  to the range  $[result, result + (last - first))$  such that the order of the elements is reversed. For every  $i$  such that  $0 \leq i < (last - first)$ , `reverse_copy()` performs the assignment  $*(result + (last - first) - i) = *(first + i)$ . The ranges  $[first, last)$  and  $[result, result + (last - first))$  must not overlap.

Definition at line 1462 of file `stl_algo.h`.

**2.28.1.28** `template<typename _ForwardIterator > void std::rotate  
( _ForwardIterator __first, _ForwardIterator __middle,  
_ForwardIterator __last ) [inline]`

Rotate the elements of a sequence.

#### Parameters

*first* A forward iterator.

*middle* A forward iterator.

*last* A forward iterator.

**Returns**

Nothing.

Rotates the elements of the range `[first,last)` by `(middle-first)` positions so that the element at `middle` is moved to `first`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

This effectively swaps the ranges `[first,middle)` and `[middle,last)`.

Performs `*(first+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1666 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__stable_partition_adaptive()`.

```
2.28.1.29 template<typename _ForwardIterator , typename _OutputIterator
> _OutputIterator std::rotate_copy ( _ForwardIterator
__first, _ForwardIterator __middle, _ForwardIterator __last,
_OutputIterator __result )
```

Copy a sequence, rotating its elements.

**Parameters**

*first* A forward iterator.

*middle* A forward iterator.

*last* A forward iterator.

*result* An output iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[first,last)` to the range beginning at

**Returns**

, rotating the copied elements by `(middle-first)` positions so that the element at `middle` is moved to `result`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

Performs `*(result+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1700 of file `stl_algo.h`.

**2.28.1.30** `template<typename _RandomAccessIterator , typename  
_UniformRandomNumberGenerator > void std::shuffle (   
_RandomAccessIterator __first, _RandomAccessIterator __last,  
_UniformRandomNumberGenerator & __g )`

Shuffle the elements of a sequence using a uniform random number generator.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*g* A UniformRandomNumberGenerator (26.5.1.3).

#### Returns

Nothing.

Reorders the elements in the range [first,last) using *g* to provide random numbers.

Definition at line 4135 of file `stl_algo.h`.

References `std::iter_swap()`.

**2.28.1.31** `template<typename _ForwardIterator , typename _Predicate >  
_ForwardIterator std::stable_partition ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred )`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*pred* A predicate functor.

#### Returns

An iterator `middle` such that `pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `pred(x)==pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1858 of file `stl_algo.h`.

References `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::Temporary_buffer<_ForwardIterator, _Tp >::requested_size()`, and `std::Temporary_buffer<_ForwardIterator, _Tp >::size()`.

**2.28.1.32** `template<typename _Tp > void std::swap ( _Tp & __a, _Tp & __b ) [inline]`

Swaps two values.

#### Parameters

*\_\_a* A thing of arbitrary type.

*\_\_b* Another thing of arbitrary type.

#### Returns

Nothing.

Definition at line 106 of file `move.h`.

**2.28.1.33** `template<typename _ForwardIterator1 , typename _ForwardIterator2 > _ForwardIterator2 std::swap_ranges ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2 )`

Swap the elements of two sequences.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

#### Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 158 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

**2.28.1.34** `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation> _OutputIterator std::transform ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op )`

Perform an operation on corresponding elements of two sequences.

#### Parameters

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*result* An output iterator.  
*binary\_op* A binary operator.

#### Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(result+N)=binary_op(*(first1+N),*(first2+N))` for each `N` in the range `[0,last1-first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4713 of file `stl_algo.h`.

**2.28.1.35** `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation> _OutputIterator std::transform ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op )`

Perform an operation on a sequence.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*result* An output iterator.  
*unary\_op* A unary operator.

#### Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates  $*(result+N)=unary\_op(*(first+N))$  for each  $N$  in the range  $[0, last-first)$ .

`unary_op` must not alter its argument.

Definition at line 4677 of file `stl_algo.h`.

**2.28.1.36** `template<typename _ForwardIterator , typename _BinaryPredicate  
> _ForwardIterator std::unique ( _ForwardIterator __first,  
_ForwardIterator __last, _BinaryPredicate __binary_pred )`

Remove consecutive values from a sequence using a predicate.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

*binary\_pred* A binary predicate.

#### Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1207 of file `stl_algo.h`.

**2.28.1.37** `template<typename _ForwardIterator > _ForwardIterator  
std::unique ( _ForwardIterator __first, _ForwardIterator __last )`

Remove consecutive duplicate values from a sequence.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is

unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1167 of file `stl_algo.h`.

**2.28.1.38** `template<typename _InputIterator , typename _OutputIterator  
> _OutputIterator std::unique_copy ( _InputIterator __first,  
_InputIterator __last, _OutputIterator __result ) [inline]`

Copy a sequence, removing consecutive duplicate values.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

#### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4876 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

**2.28.1.39** `template<typename _InputIterator , typename _OutputIterator ,  
typename _BinaryPredicate > _OutputIterator std::unique_copy (   
_InputIterator __first, _InputIterator __last, _OutputIterator  
__result, _BinaryPredicate __binary_pred ) [inline]`

Copy a sequence, removing consecutive values using a predicate.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*binary\_pred* A binary predicate.

### Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) to the range beginning at *result*, except that only the first element is copied from groups of consecutive elements for which *binary\_pred* returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require Copy-Constructible and Assignable?

Definition at line 4916 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

## 2.29 Non-Mutating

Collaboration diagram for Non-Mutating:



### Functions

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __`  
`first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator`  
`__first, _InputIterator __last, _Predicate __pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _`  
`BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`  
`&__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _`  
`BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function`  
`__f)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`  
`pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2)`

- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

## 2.29.1 Function Documentation

### 2.29.1.1 `template<typename _ForwardIterator > _ForwardIterator std::adjacent_find ( _ForwardIterator __first, _ForwardIterator __last )`

Find two adjacent values in a sequence that are equal.

#### Parameters

*first* A forward iterator.

*last* A forward iterator.

#### Returns

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that *\*i* == *\*(i+1)*, or *last* if no such iterator exists.

Definition at line 4324 of file `stl_algo.h`.

### 2.29.1.2 `template<typename _ForwardIterator, typename _BinaryPredicate > _ForwardIterator std::adjacent_find ( _ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred )`

Find two adjacent values in a sequence using a predicate.

**Parameters**

*first* A forward iterator.

*last* A forward iterator.

*binary\_pred* A binary predicate.

**Returns**

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that `binary_pred(*i,*i+1)` is true, or *last* if no such iterator exists.

Definition at line 4356 of file `stl_algo.h`.

```
2.29.1.3 template<typename _InputIterator , typename _Predicate > bool
std::all_of ( _InputIterator __first, _InputIterator __last, _Predicate
__pred ) [inline]
```

Checks that a predicate is true for all the elements of a sequence.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

**Returns**

True if the check is true, false otherwise.

Returns true if *pred* is true for each element in the range [*first*,*last*), and false otherwise.

Definition at line 728 of file `stl_algo.h`.

References `std::find_if_not()`.

```
2.29.1.4 template<typename _InputIterator , typename _Predicate >
bool std::any_of ( _InputIterator __first, _InputIterator __last,
_Predicate __pred ) [inline]
```

Checks that a predicate is false for at least an element of a sequence.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

**Returns**

True if the check is true, false otherwise.

Returns true if an element exists in the range [first,last) such that *pred* is true, and false otherwise.

Definition at line 762 of file `stl_algo.h`.

References `std::none_of()`.

**2.29.1.5** `template<typename _InputIterator , typename _Tp >  
iterator_traits<_InputIterator>::difference_type std::count (  
_InputIterator __first, _InputIterator __last, const _Tp & __value )`

Count the number of copies of a value in a sequence.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*value* The value to be counted.

**Returns**

The number of iterators *i* in the range [first,last) for which `*i == value`

Definition at line 4388 of file `stl_algo.h`.

**2.29.1.6** `template<typename _InputIterator , typename _Predicate >  
iterator_traits<_InputIterator>::difference_type std::count_if (  
_InputIterator __first, _InputIterator __last, _Predicate __pred )`

Count the elements of a sequence for which a predicate is true.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

**Returns**

The number of iterators *i* in the range [first,last) for which `pred(*i)` is true.

Definition at line 4413 of file `stl_algo.h`.

**2.29.1.7** `template<typename _II1, typename _II2 > bool std::equal ( _II1  
_first1, _II1 _last1, _II2 _first2 ) [inline]`

Tests a range for element-wise equality.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

#### Returns

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1030 of file `stl_algobase.h`.

Referenced by `std::operator==()`.

**2.29.1.8** `template<typename _IIter1, typename _IIter2, typename  
_BinaryPredicate > bool std::equal ( _IIter1 _first1, _IIter1 _last1,  
_IIter2 _first2, _BinaryPredicate _binary_pred ) [inline]`

Tests a range for element-wise equality.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

*binary\_pred* A binary predicate [functor](#).

#### Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1062 of file `stl_algobase.h`.

```

2.29.1.9 template<typename _InputIterator , typename _Tp > _InputIterator
std::find ( _InputIterator __first, _InputIterator __last, const _Tp &
__val ) [inline]

```

Find the first occurrence of a value in a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*val* The value to find.

#### Returns

The first iterator *i* in the range [*first*,*last*) such that *\*i == val*, or *last* if no such iterator exists.

Definition at line 4200 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

```

2.29.1.10 template<typename _ForwardIterator1 , typename
_ForwardIterator2 , typename _BinaryPredicate >
_ForwardIterator1 std::find_end ( _ForwardIterator1 __first1,
_ForwardIterator1 __last1, _ForwardIterator2 __first2,
_ForwardIterator2 __last2, _BinaryPredicate __comp )
[inline]

```

Find last matching subsequence in a sequence using a predicate.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of sequence to match.

*last2* End of sequence to match.

*comp* The predicate to use.

#### Returns

The last iterator *i* in the range [*first1*,*last1*-(*last2*-*first2*)) such that `predicate(*(i+N), (first2+N))` is true for each *N* in the range [0,*last2*-*first2*), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 694 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.29.1.11 template<typename _ForwardIterator1 , typename
           _ForwardIterator2 > _ForwardIterator1 std::find_end (
           _ForwardIterator1 __first1, _ForwardIterator1 __last1,
           _ForwardIterator2 __first2, _ForwardIterator2 __last2 )
           [inline]
```

Find last matching subsequence in a sequence.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of sequence to match.

*last2* End of sequence to match.

#### Returns

The last iterator `i` in the range `[first1,last1-(last2-first2))` such that `*(i+N) == *(first2+N)` for each `N` in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 647 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.29.1.12 template<typename _InputIterator , typename _ForwardIterator ,
            typename _BinaryPredicate > _InputIterator std::find_first_of (
            _InputIterator __first1, _InputIterator __last1, _ForwardIterator
            __first2, _ForwardIterator __last2, _BinaryPredicate __comp )
```

Find element from a set in a sequence using a predicate.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of match candidates.

*last2* End of match candidates.

*comp* Predicate to use.

#### Returns

The first iterator *i* in the range [first1,last1) such that `comp(*i, *(i2))` is true and *i2* is an iterator in [first2,last2), or *last1* if no such iterator exists.

Searches the range [first1,last1) for an element that is equal to some element in the range [first2,last2). If found, returns an iterator in the range [first1,last1), otherwise returns *last1*.

Definition at line 4293 of file `stl_algo.h`.

```
2.29.1.13 template<typename _InputIterator , typename _ForwardIterator
            > _InputIterator std::find_first_of ( _InputIterator
            __first1, _InputIterator __last1, _ForwardIterator __first2,
            _ForwardIterator __last2 )
```

Find element from a set in a sequence.

#### Parameters

*first1* Start of range to search.

*last1* End of range to search.

*first2* Start of match candidates.

*last2* End of match candidates.

#### Returns

The first iterator *i* in the range [first1,last1) such that `*i == *(i2)` such that *i2* is an iterator in [first2,last2), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for an element that is equal to some element in the range `[first2,last2)`. If found, returns an iterator in the range `[first1,last1)`, otherwise returns `last1`.

Definition at line 4253 of file `stl_algo.h`.

**2.29.1.14** `template<typename _InputIterator , typename _Predicate >`  
`__InputIterator std::find_if ( _InputIterator __first, _InputIterator`  
`__last, _Predicate __pred ) [inline]`

Find the first element in a sequence for which a predicate is true.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is true, or `last` if no such iterator exists.

Definition at line 4224 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

**2.29.1.15** `template<typename _InputIterator , typename _Predicate`  
`> __InputIterator std::find_if_not ( _InputIterator __first,`  
`__InputIterator __last, _Predicate __pred ) [inline]`

Find the first element in a sequence for which a predicate is false.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is false, or `last` if no such iterator exists.

Definition at line 777 of file `stl_algo.h`.

References `std::__find_if_not()`, and `std::__iterator_category()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

```
2.29.1.16 template<typename _InputIterator , typename _Function >
           _Function std::for_each ( _InputIterator __first, _InputIterator
           __last, _Function __f )
```

Apply a function to every element of a sequence.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*f* A unary function object.

#### Returns

$f$  (std::move( $f$ ) in C++0x).

Applies the function object  $f$  to each element in the range [first,last).  $f$  must not modify the order of the sequence. If  $f$  has a return value it is ignored.

Definition at line 4179 of file stl\_algo.h.

```
2.29.1.17 template<typename _InputIterator1 , typename _InputIterator2
           , typename _BinaryPredicate > pair<_InputIterator1,
           _InputIterator2> std::mismatch ( _InputIterator1 __first1,
           _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate
           __binary_pred )
```

Finds the places in ranges which don't match.

#### Parameters

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*binary\_pred* A binary predicate [functor](#).

#### Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the *binary\_pred* parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1205 of file stl\_algobase.h.

```
2.29.1.18 template<typename _InputIterator1 , typename _InputIterator2
> pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
    __first2 )
```

Finds the places in ranges which don't match.

#### Parameters

*first1* An input iterator.

*last1* An input iterator.

*first2* An input iterator.

#### Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using == and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1167 of file stl\_algobase.h.

```
2.29.1.19 template<typename _InputIterator , typename _Predicate > bool
std::none_of ( _InputIterator __first, _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is false for all the elements of a sequence.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*pred* A predicate.

#### Returns

True if the check is true, false otherwise.

Returns true if *pred* is false for each element in the range [first,last), and false otherwise.

Definition at line 745 of file stl\_algo.h.

Referenced by std::any\_of(), and std::is\_partitioned().

```
2.29.1.20 template<typename _ForwardIterator1 , typename
        _ForwardIterator2 > _ForwardIterator1 std::search (
        _ForwardIterator1 __first1, _ForwardIterator1 __last1,
        _ForwardIterator2 __first2, _ForwardIterator2 __last2 )
```

Search a sequence for a matching sub-sequence.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

*last2* A forward iterator.

#### Returns

The first iterator *i* in the range  $[first1, last1 - (last2 - first2))$  such that  $*(i+N) == *(first2+N)$  for each *N* in the range  $[0, last2 - first2)$ , or *last1* if no such iterator exists.

Searches the range  $[first1, last1)$  for a sub-sequence that compares equal value-by-value with the sequence given by  $[first2, last2)$  and returns an iterator to the first element of the sub-sequence, or *last1* if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range  $[first1, last1)$  it must start at a position less than  $last1 - (last2 - first2)$  where  $last2 - first2$  is the length of the sub-sequence. This means that the returned iterator *i* will be in the range  $[first1, last1 - (last2 - first2))$

Definition at line 4453 of file `stl_algo.h`.

```
2.29.1.21 template<typename _ForwardIterator1 , typename
        _ForwardIterator2 , typename _BinaryPredicate >
        _ForwardIterator1 std::search ( _ForwardIterator1 __first1,
        _ForwardIterator1 __last1, _ForwardIterator2 __first2,
        _ForwardIterator2 __last2, _BinaryPredicate __predicate )
```

Search a sequence for a matching sub-sequence using a predicate.

#### Parameters

*first1* A forward iterator.

*last1* A forward iterator.

*first2* A forward iterator.

*last2* A forward iterator.

*predicate* A binary predicate.

### Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `predicate(*(i+N),*(first2+N))` is true for each *N* in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)`, using `predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `last1` if no such iterator exists.

### See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4525 of file `stl_algo.h`.

**2.29.1.22** `template<typename _ForwardIterator , typename _Integer ,  
typename _Tp > _ForwardIterator std::search_n ( _ForwardIterator  
__first, _ForwardIterator __last, _Integer __count, const _Tp &  
__val )`

Search a sequence for a number of consecutive values.

### Parameters

*first* A forward iterator.

*last* A forward iterator.

*count* The number of consecutive values.

*val* The value to find.

### Returns

The first iterator *i* in the range `[first,last-count)` such that `*(i+N) == val` for each *N* in the range `[0,count)`, or `last` if no such iterator exists.

Searches the range `[first,last)` for `count` consecutive elements equal to `val`.

Definition at line 4598 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

```
2.29.1.23 template<typename _ForwardIterator , typename _Integer ,
            typename _Tp , typename _BinaryPredicate > _ForwardIterator
            std::search_n ( _ForwardIterator __first, _ForwardIterator
                __last, _Integer __count, const _Tp & __val, _BinaryPredicate
                __binary_pred )
```

Search a sequence for a number of consecutive values using a predicate.

### Parameters

*first* A forward iterator.

*last* A forward iterator.

*count* The number of consecutive values.

*val* The value to find.

*binary\_pred* A binary predicate.

### Returns

The first iterator *i* in the range `[first,last-count)` such that `binary_pred(*(i+N),val)` is true for each *N* in the range `[0,count)`, or *last* if no such iterator exists.

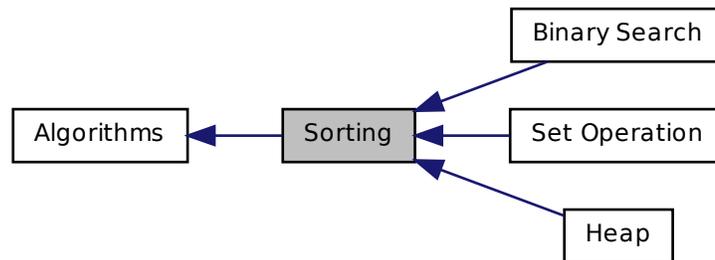
Searches the range `[first,last)` for `count` consecutive elements for which the predicate returns true.

Definition at line 4635 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

## 2.30 Sorting

Collaboration diagram for Sorting:



### Modules

- [Set Operation](#)
- [Binary Search](#)
- [Heap](#)

### Functions

- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _`  
`Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b)`

- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_-`  
`ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_-`  
`ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __-`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`  
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __-`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`  
`RandomAccessIterator __result_last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`

### 2.30.1 Function Documentation

**2.30.1.1** `template<typename _BidirectionalIterator > void std::inplace_merge`  
`( _BidirectionalIterator __first, _BidirectionalIterator __middle,`  
`_BidirectionalIterator __last )`

Merges two sorted ranges in place.

#### Parameters

- first* An iterator.
- middle* Another iterator.
- last* Another iterator.

#### Returns

Nothing.

Merges two sorted and consecutive ranges, `[first,middle)` and `[middle,last)`, and puts the result in `[first,last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(last-first)-1$  comparisons. Otherwise an  $N\log N$  algorithm is used, where  $N$  is `distance(first,last)`.

Definition at line 3061 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

**2.30.1.2** `template<typename _BidirectionalIterator, typename _Compare`  
`> void std::inplace_merge ( _BidirectionalIterator __first,`  
`_BidirectionalIterator __middle, _BidirectionalIterator __last,`  
`_Compare __comp )`

Merges two sorted ranges in place.

**Parameters**

- first* An iterator.
- middle* Another iterator.
- last* Another iterator.
- comp* A functor to use for comparisons.

**Returns**

Nothing.

Merges two sorted and consecutive ranges, [first,middle) and [middle,last), and puts the result in [first,last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (last-first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(first,last).

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3116 of file stl\_algo.h.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

### 2.30.1.3 `template<typename _ForwardIterator > bool std::is_sorted (` `_FowardIterator __first, _ForwardIterator __last ) [inline]`

Determines whether the elements of a sequence are sorted.

**Parameters**

- first* An iterator.
- last* Another iterator.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3816 of file stl\_algo.h.

References `std::is_sorted_until()`.

**2.30.1.4** `template<typename _ForwardIterator, typename _Compare > bool  
std::is_sorted ( _ForwardIterator __first, _ForwardIterator __last,  
_Compare __comp ) [inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

#### Parameters

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

True if the elements are sorted, false otherwise.

Definition at line 3830 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**2.30.1.5** `template<typename _ForwardIterator > _ForwardIterator  
std::is_sorted_until ( _ForwardIterator __first, _ForwardIterator  
__last )`

Determines the end of a sorted sequence.

#### Parameters

*first* An iterator.

*last* Another iterator.

#### Returns

An iterator pointing to the last iterator *i* in `[first, last)` for which the range `[first, i)` is sorted.

Definition at line 3844 of file `stl_algo.h`.

**2.30.1.6** `template<typename _ForwardIterator, typename _Compare >  
_ForwardIterator std::is_sorted_until ( _ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp )`

Determines the end of a sorted sequence using comparison functor.

**Parameters**

*first* An iterator.  
*last* Another iterator.  
*comp* A comparison functor.

**Returns**

An iterator pointing to the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is sorted.

Definition at line 3873 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

**2.30.1.7** `template<typename _II1 , typename _II2 > bool  
std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2  
__first2, _II2 __last2 ) [inline]`

Performs **dictionary** comparison on ranges.

**Parameters**

*first1* An input iterator.  
*last1* An input iterator.  
*first2* An input iterator.  
*last2* An input iterator.

**Returns**

A boolean true or false.

*Returns true if the sequence of elements defined by the range [*first1*,*last1*) is lexicographically less than the sequence of elements defined by the range [*first2*,*last2*). Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1093 of file `stl_algobase.h`.

**2.30.1.8** `template<typename _II1 , typename _II2 , typename _Compare >  
bool std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2  
__first2, _II2 __last2, _Compare __comp )`

Performs **dictionary** comparison on ranges.

**Parameters**

- first1* An input iterator.
- last1* An input iterator.
- first2* An input iterator.
- last2* An input iterator.
- comp* A [comparison functor](#).

**Returns**

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1127 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

**2.30.1.9** `template<typename _Tp> const _Tp & std::max ( const _Tp & __a,  
const _Tp & __b ) [inline]`

This does what you think it does.

**Parameters**

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

**Returns**

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 209 of file `stl_algobase.h`.

Referenced by `std::_Deque_base< _Tp, _Alloc >::_M_initialize_map()`, and `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

**2.30.1.10** `template<typename _Tp, typename _Compare> const _Tp &  
std::max ( const _Tp & __a, const _Tp & __b, _Compare __comp  
) [inline]`

This does what you think it does.

**Parameters**

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.
- comp* A [comparison functor](#).

**Returns**

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 253 of file `stl_algobase.h`.

**2.30.1.11** `template<typename _ForwardIterator > _ForwardIterator  
std::max_element ( _ForwardIterator __first, _ForwardIterator  
__last )`

Return the maximum element in a range.

**Parameters**

- first* Start of range.
- last* End of range.

**Returns**

Iterator referencing the first instance of the largest value.

Definition at line 6034 of file `stl_algo.h`.

**2.30.1.12** `template<typename _ForwardIterator , typename _Compare >  
_ForwardIterator std::max_element ( _ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp )`

Return the maximum element in a range using comparison functor.

**Parameters**

- first* Start of range.
- last* End of range.
- comp* Comparison functor.

**Returns**

Iterator referencing the first instance of the largest value according to *comp*.

Definition at line 6062 of file stl\_algo.h.

Referenced by std::valarray< \_Tp >::max().

```
2.30.1.13 template<typename _InputIterator1 , typename _InputIterator2
, typename _OutputIterator > _OutputIterator std::merge (
    _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
    __first2, _InputIterator2 __last2, _OutputIterator __result )
```

Merges two sorted ranges.

#### Parameters

*first1* An iterator.

*first2* Another iterator.

*last1* Another iterator.

*last2* Another iterator.

*result* An iterator pointing to the end of the merged range.

#### Returns

An iterator pointing to the first element *not less than val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5277 of file stl\_algo.h.

```
2.30.1.14 template<typename _InputIterator1 , typename _InputIterator2 ,
typename _OutputIterator , typename _Compare > _OutputIterator
std::merge ( _InputIterator1 __first1, _InputIterator1 __last1,
    _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator
    __result, _Compare __comp )
```

Merges two sorted ranges.

#### Parameters

*first1* An iterator.

*first2* Another iterator.

*last1* Another iterator.

*last2* Another iterator.

*result* An iterator pointing to the end of the merged range.

*comp* A functor to use for comparisons.

### Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5340 of file stl\_algo.h.

**2.30.1.15** `template<typename _Tp , typename _Compare > const _Tp & std::min ( const _Tp & __a, const _Tp & __b, _Compare __comp ) [inline]`

This does what you think it does.

### Parameters

*a* A thing of arbitrary type.

*b* Another thing of arbitrary type.

*comp* A [comparison functor](#).

### Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 232 of file stl\_algobase.h.

**2.30.1.16** `template<typename _Tp > const _Tp & std::min ( const _Tp & __a, const _Tp & __b ) [inline]`

This does what you think it does.

**Parameters**

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

**Returns**

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 186 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_profile::__report()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<char >::compare()`, `std::generate_canonical()`, `__gnu_cxx::random_sample_n()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`, `std::basic_string<_CharT, _Traits, _Alloc >::rfind()`, `std::basic_filebuf<_CharT, _Traits >::underflow()`, `std::basic_streambuf<_CharT, _Traits >::xsgetn()`, and `std::basic_streambuf<_CharT, _Traits >::xsputn()`.

**2.30.1.17** `template<typename _ForwardIterator > _ForwardIterator  
std::min_element ( _ForwardIterator __first, _ForwardIterator  
__last )`

Return the minimum element in a range.

**Parameters**

- first* Start of range.
- last* End of range.

**Returns**

Iterator referencing the first instance of the smallest value.

Definition at line 5978 of file `stl_algo.h`.

**2.30.1.18** `template<typename _ForwardIterator, typename _Compare >  
_ForwardIterator std::min_element ( _ForwardIterator __first,  
_ForwardIterator __last, _Compare __comp )`

Return the minimum element in a range using comparison functor.

**Parameters**

*first* Start of range.  
*last* End of range.  
*comp* Comparison functor.

**Returns**

Iterator referencing the first instance of the smallest value according to *comp*.

Definition at line 6006 of file `stl_algo.h`.

Referenced by `std::valarray<_Tp>::min()`.

**2.30.1.19** `template<typename _Tp, typename _Compare> pair< const _Tp &, const _Tp & > std::minmax ( const _Tp & __a, const _Tp & __b, _Compare __comp ) [inline]`

Determines min and max at once as an ordered pair.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.  
*comp* A [comparison functor](#).

**Returns**

A pair(*b*, *a*) if *b* is smaller than *a*, pair(*a*, *b*) otherwise.

Definition at line 3921 of file `stl_algo.h`.

**2.30.1.20** `template<typename _Tp> pair< const _Tp &, const _Tp & > std::minmax ( const _Tp & __a, const _Tp & __b ) [inline]`

Determines min and max at once as an ordered pair.

**Parameters**

*a* A thing of arbitrary type.  
*b* Another thing of arbitrary type.

**Returns**

A pair(*b*, *a*) if *b* is smaller than *a*, pair(*a*, *b*) otherwise.

Definition at line 3902 of file `stl_algo.h`.

```
2.30.1.21  template<typename _ForwardIterator > pair<_ForwardIterator,
           _ForwardIterator> std::minmax_element ( _ForwardIterator
           __first, _ForwardIterator __last )
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

#### Parameters

*first* Start of range.

*last* End of range.

#### Returns

make\_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 3940 of file stl\_algo.h.

```
2.30.1.22  template<typename _ForwardIterator , typename _Compare >
           pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
           _ForwardIterator __first, _ForwardIterator __last, _Compare
           __comp )
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

#### Parameters

*first* Start of range.

*last* End of range.

*comp* Comparison functor.

#### Returns

make\_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 4016 of file stl\_algo.h.

```
2.30.1.23  template<typename _BidirectionalIterator > bool
           std::next_permutation ( _BidirectionalIterator __first,
           _BidirectionalIterator __last )
```

Permute range into the next *dictionary* ordering.

**Parameters**

*first* Start of range.

*last* End of range.

**Returns**

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3518 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

**2.30.1.24** `template<typename _BidirectionalIterator , typename _Compare  
> bool std::next_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last, _Compare __comp )`

Permute range into the next *dictionary* ordering using comparison functor.

**Parameters**

*first* Start of range.

*last* End of range.

*comp* A comparison functor.

**Returns**

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range [first,last) as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3575 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

**2.30.1.25** `template<typename _RandomAccessIterator > void std::nth_element  
( _RandomAccessIterator __first, _RandomAccessIterator __nth,  
_RandomAccessIterator __last ) [inline]`

Sort a sequence just enough to find a particular position.

**Parameters**

*first* An iterator.

*nth* Another iterator.

*last* Another iterator.

**Returns**

Nothing.

Rearranges the elements in the range [first,last) so that \*nth is the same element that would have been in that position had the whole sequence been sorted. The elements either side of \*nth are not completely sorted, but for any iterator in the range [first,nth) and any iterator in the range [nth,last) it holds that \*j < \*i is false.

Definition at line 5122 of file stl\_algo.h.

References std::\_\_lg().

```
2.30.1.26 template<typename _RandomAccessIterator , typename _Compare  
> void std::nth_element ( _RandomAccessIterator __first,  
_RandomAccessIterator __nth, _RandomAccessIterator __last,  
_Compare __comp ) [inline]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

**Parameters**

*first* An iterator.

*nth* Another iterator.

*last* Another iterator.

*comp* A comparison functor.

**Returns**

Nothing.

Rearranges the elements in the range [first,last) so that \*nth is the same element that would have been in that position had the whole sequence been sorted. The elements either side of \*nth are not completely sorted, but for any iterator in the range [first,nth) and any iterator in the range [nth,last) it holds that comp(\*j, \*i) is false.

Definition at line 5161 of file stl\_algo.h.

References std::\_\_lg().

**2.30.1.27** `template<typename _RandomAccessIterator > void std::partial_sort  
( _RandomAccessIterator __first, _RandomAccessIterator  
__middle, _RandomAccessIterator __last ) [inline]`

Sort the smallest elements of a sequence.

#### Parameters

*first* An iterator.

*middle* Another iterator.

*last* Another iterator.

#### Returns

Nothing.

Sorts the smallest (middle-first) elements in the range [first,last) and moves them to the range [first,middle). The order of the remaining elements in the range [middle,last) is undefined. After the sort if *i* and *j* are iterators in the range [first,middle) such that *j* precedes *i* and *k* is an iterator in the range [middle,last) then *\*j*<*\*i* and *\*k*<*\*i* are both false.

Definition at line 5045 of file stl\_algo.h.

References std::\_\_heap\_select(), and std::sort\_heap().

**2.30.1.28** `template<typename _RandomAccessIterator , typename _Compare  
> void std::partial_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __middle, _RandomAccessIterator __last,  
_Compare __comp ) [inline]`

Sort the smallest elements of a sequence using a predicate for comparison.

#### Parameters

*first* An iterator.

*middle* Another iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

Nothing.

Sorts the smallest (middle-first) elements in the range [first,last) and moves them to the range [first,middle). The order of the remaining elements in the range [middle,last) is

undefined. After the sort if  $i$  and  $j$  are iterators in the range  $[first, middle)$  such that  $j$  precedes  $i$  and  $i$  is an iterator in the range  $[middle, last)$  then  $*comp(j, *i)$  and  $comp(*i, *j)$  are both false.

Definition at line 5084 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

**2.30.1.29** `template<typename _InputIterator , typename  
_RandomAccessIterator , typename _Compare >  
_RandomAccessIterator std::partial_sort_copy ( _InputIterator  
__first, _InputIterator __last, _RandomAccessIterator __result_first,  
_RandomAccessIterator __result_last, _Compare __comp )`

Copy the smallest elements of a sequence using a predicate for comparison.

#### Parameters

*first* An input iterator.

*last* Another input iterator.

*result\_first* A random-access iterator.

*result\_last* Another random-access iterator.

*comp* A comparison functor.

#### Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[first, last)$  to the range beginning at `result_first`, where the number of elements to be copied,  $N$ , is the smaller of  $(last - first)$  and  $(result\_last - result\_first)$ . After the sort if  $i$  and  $j$  are iterators in the range  $[result\_first, result\_first + N)$  such that  $j$  precedes  $i$  then `comp(*j, *i)` is false. The value returned is `result_first + N`.

Definition at line 2006 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

**2.30.1.30** `template<typename _InputIterator , typename  
_RandomAccessIterator > _RandomAccessIterator  
std::partial_sort_copy ( _InputIterator __first,  
_InputIterator __last, _RandomAccessIterator __result_first,  
_RandomAccessIterator __result_last )`

Copy the smallest elements of a sequence.

**Parameters**

*first* An iterator.  
*last* Another iterator.  
*result\_first* A random-access iterator.  
*result\_last* Another random-access iterator.

**Returns**

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range [first,last) to the range beginning at *result\_first*, where the number of elements to be copied, N, is the smaller of (last-first) and (result\_last-result\_first). After the sort if *i* and *j* are iterators in the range [result\_first,result\_first+N) such that *j* precedes *i* then *\*j*<*\*i* is false. The value returned is *result\_first*+N.

Definition at line 1940 of file stl\_algo.h.

References std::make\_heap(), and std::sort\_heap().

**2.30.1.31** `template<typename _BidirectionalIterator , typename _Compare  
> bool std::prev_permutation ( _BidirectionalIterator __first,  
_BidirectionalIterator __last, _Compare __comp )`

Permute range into the previous *dictionary* ordering using comparison functor.

**Parameters**

*first* Start of range.  
*last* End of range.  
*comp* A comparison functor.

**Returns**

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [first,last) as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3688 of file stl\_algo.h.

References std::iter\_swap(), and std::reverse().

```
2.30.1.32 template<typename _BidirectionalIterator > bool
std::prev_permutation ( _BidirectionalIterator __first,
_BidirectionalIterator __last )
```

Permute range into the previous *dictionary* ordering.

#### Parameters

*first* Start of range.

*last* End of range.

#### Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3631 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

```
2.30.1.33 template<typename _RandomAccessIterator , typename
_Compare > void std::sort ( _RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison.

#### Parameters

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

#### Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for every iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5235 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

---

**2.30.1.34** `template<typename _RandomAccessIterator > void std::sort (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Sort the elements of a sequence.

**Parameters**

*first* An iterator.

*last* Another iterator.

**Returns**

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5199 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

**2.30.1.35** `template<typename _RandomAccessIterator , typename _Compare`  
`> void std::stable_sort ( _RandomAccessIterator __first,`  
`_RandomAccessIterator __last, _Compare __comp ) [inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

**Parameters**

*first* An iterator.

*last* Another iterator.

*comp* A comparison functor.

**Returns**

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `comp(x,y)` is false and `comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5441 of file stl\_algo.h.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

**2.30.1.36** `template<typename _RandomAccessIterator > void std::stable_sort`  
`( _RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Sort the elements of a sequence, preserving the relative order of equivalent elements.

#### Parameters

*first* An iterator.

*last* Another iterator.

#### Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

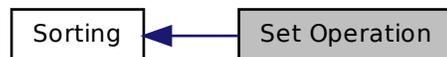
The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5399 of file stl\_algo.h.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

## 2.31 Set Operation

Collaboration diagram for Set Operation:



## Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`  
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`

### 2.31.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

### 2.31.2 Function Documentation

**2.31.2.1** `template<typename _InputIterator1 , typename _InputIterator2 >  
bool std::includes ( _InputIterator1 __first1, _InputIterator1 __last1,  
_InputIterator2 __first2, _InputIterator2 __last2 )`

Determines whether all elements of a sequence exists in a range.

#### Parameters

*first1* Start of search range.

*last1* End of search range.

*first2* Start of sequence

*last2* End of sequence.

#### Returns

True if each element in [first2,last2) is contained in order within [first1,last1). False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3414 of file stl\_algo.h.

**2.31.2.2** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _Compare > bool std::includes ( _InputIterator1 __first1,  
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _Compare __comp )`

Determines whether all elements of a sequence exists in a range using comparison.

#### Parameters

*first1* Start of search range.

*last1* End of search range.

*first2* Start of sequence

*last2* End of sequence.

*comp* Comparison function to use.

### Returns

True if each element in [first2,last2) is contained in order within [first1,last1) according to comp. False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3464 of file stl\_algo.h.

```
2.31.2.3 template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _OutputIterator , typename _Compare > _OutputIterator
            std::set_difference ( _InputIterator1 __first1, _InputIterator1
                                __last1, _InputIterator2 __first2, _InputIterator2 __last2,
                                _OutputIterator __result, _Compare __comp )
```

Return the difference of two sorted ranges using comparison functor.

### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

*comp* The comparison functor.

### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to *comp*, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5798 of file stl\_algo.h.

**2.31.2.4** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator > _OutputIterator std::set_difference (  
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the difference of two sorted ranges.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5737 of file `stl_algo.h`.

**2.31.2.5** `template<typename _InputIterator1 , typename _InputIterator2 ,  
typename _OutputIterator > _OutputIterator std::set_intersection (  
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2, _InputIterator2 __last2, _OutputIterator __result )`

Return the intersection of two sorted ranges.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5622 of file `stl_algo.h`.

```
2.31.2.6 template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _OutputIterator , typename _Compare > _OutputIterator
std::set_intersection ( _InputIterator1 __first1, _InputIterator1
__last1, _InputIterator2 __first2, _InputIterator2 __last2,
_OutputIterator __result, _Compare __comp )
```

Return the intersection of two sorted ranges using comparison functor.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

*comp* The comparison functor.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that iterator advances. If an element is contained in both ranges according to *comp*, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5679 of file `stl_algo.h`.

```
2.31.2.7 template<typename _InputIterator1 , typename
            _InputIterator2 , typename _OutputIterator > _OutputIterator
std::set_symmetric_difference ( _InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result )
```

Return the symmetric difference of two sorted ranges.

**Parameters**

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5856 of file `stl_algo.h`.

```
2.31.2.8 template<typename _InputIterator1 , typename _InputIterator2 ,  
            typename _OutputIterator , typename _Compare > _OutputIterator  
            std::set_symmetric_difference ( _InputIterator1 __first1,  
            _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
            __last2, _OutputIterator __result, _Compare __comp )
```

Return the symmetric difference of two sorted ranges using comparison functor.

**Parameters**

*first1* Start of first range.  
*last1* End of first range.  
*first2* Start of second range.  
*last2* End of second range.  
*comp* The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5922 of file `stl_algo.h`.

```
2.31.2.9  template<typename _InputIterator1 , typename _InputIterator2 ,
          typename _OutputIterator > _OutputIterator std::set_union (
            _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
            __first2, _InputIterator2 __last2, _OutputIterator __result )
```

Return the union of two sorted ranges.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5488 of file `stl_algo.h`.

```
2.31.2.10  template<typename _InputIterator1 , typename _InputIterator2 ,
           typename _OutputIterator , typename _Compare > _OutputIterator
std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1,
                _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator
                __result, _Compare __comp )
```

Return the union of two sorted ranges using a comparison functor.

#### Parameters

*first1* Start of first range.

*last1* End of first range.

*first2* Start of second range.

*last2* End of second range.

*comp* The comparison functor.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advanced. If an equivalent element according to *comp* is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5555 of file `stl_algo.h`.

## 2.32 Binary Search

Collaboration diagram for Binary Search:



### Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`

### 2.32.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by `equal_range` is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

### 2.32.2 Function Documentation

**2.32.2.1** `template<typename _ForwardIterator, typename _Tp > bool`  
`std::binary\_search ( _ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp & __val )`

Determines whether an element exists in a range.

#### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

#### Returns

True if *val* (or its equivalent) is in [*first*,*last* ].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2668 of file `stl_algo.h`.

References `std::lower_bound()`.

```
2.32.2.2 template<typename _ForwardIterator , typename _Tp , typename  
_Compare > bool std::binary_search ( _ForwardIterator __first,  
_ForwardIterator __last, const _Tp & __val, _Compare __comp )
```

Determines whether an element exists in a range.

#### Parameters

- first* An iterator.
- last* Another iterator.
- val* The search term.
- comp* A functor to use for comparisons.

#### Returns

True if *val* (or its equivalent) is in [*first,last* ].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2701 of file `stl_algo.h`.

References `std::lower_bound()`.

```
2.32.2.3 template<typename _ForwardIterator , typename _Tp , typename  
_Compare > pair<_ForwardIterator, _ForwardIterator>  
std::equal_range ( _ForwardIterator __first, _ForwardIterator  
_last, const _Tp & __val, _Compare __comp )
```

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

#### Parameters

- first* An iterator.
- last* Another iterator.

*val* The search term.

*comp* A functor to use for comparisons.

### Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val, comp),
               upper_bound(first, last, val, comp))
```

but does not actually call those functions.

Definition at line 2607 of file stl\_algo.h.

References std::advance(), std::distance(), std::lower\_bound(), and std::upper\_bound().

**2.32.2.4** `template<typename _ForwardIterator , typename _Tp >`  
`pair<_ForwardIterator, _ForwardIterator> std::equal_range (`  
 `_ForwardIterator __first, _ForwardIterator __last, const _Tp &`  
 `__val )`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

### Parameters

*first* An iterator.

*last* Another iterator.

*val* The search term.

### Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val),
               upper_bound(first, last, val))
```

but does not actually call those functions.

Definition at line 2545 of file stl\_algo.h.

References std::advance(), std::distance(), std::lower\_bound(), and std::upper\_bound().

```
2.32.2.5  template<typename _ForwardIterator , typename _Tp , typename
           _Compare > _ForwardIterator std::lower_bound ( _ForwardIterator
           __first, _ForwardIterator __last, const _Tp & __val, _Compare
           __comp )
```

Finds the first position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*val* The search term.  
*comp* A functor to use for comparisons.

#### Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2390 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

```
2.32.2.6  template<typename _ForwardIterator , typename _Tp >
           _ForwardIterator std::lower_bound ( _ForwardIterator __first,
           _ForwardIterator __last, const _Tp & __val )
```

Finds the first position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*val* The search term.

#### Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 954 of file `stl_algobase.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

**2.32.2.7** `template<typename _ForwardIterator, typename _Tp, typename _Compare > _ForwardIterator std::upper_bound ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp )`

Finds the last position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*val* The search term.  
*comp* A functor to use for comparisons.

#### Returns

An iterator pointing to the first element greater than *val*, or end() if no elements are greater than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2490 of file stl\_algo.h.

References std::advance(), and std::distance().

Referenced by std::\_\_merge\_adaptive(), std::\_\_merge\_without\_buffer(), and std::equal\_range().

**2.32.2.8** `template<typename _ForwardIterator, typename _Tp > _ForwardIterator std::upper_bound ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val )`

Finds the last position in which *val* could be inserted without changing the ordering.

#### Parameters

*first* An iterator.  
*last* Another iterator.  
*val* The search term.

#### Returns

An iterator pointing to the first element greater than *val*, or end() if no elements are greater than *val*.

Definition at line 2439 of file stl\_algo.h.

References std::advance(), and std::distance().

## 2.33 Allocators

Collaboration diagram for Allocators:



### Classes

- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).  
 Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class `__gnu_cxx::__pool_alloc< _Tp >`  
*Allocator using a memory pool with a single lock.*
- class `__gnu_cxx::_ExtPtr_allocator< _Tp >`  
*An example allocator which uses a non-standard pointer type.  
 This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.*
- class `__gnu_cxx::array_allocator< _Tp, _Array >`  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class `__gnu_cxx::bitmap_allocator< _Tp >`  
*Bitmap Allocator, primary template.*
- class `__gnu_cxx::debug_allocator< _Alloc >`  
*A meta-allocator with debugging bits, as per [20.4].  
 This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator `new`
  - all deallocation calls operator `delete`.

- class `__gnu_cxx::malloc_allocator< _Tp >`  
*An allocator that uses malloc.  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls `malloc`
  - all deallocation calls `free`.
- class `__gnu_cxx::new_allocator< _Tp >`  
*An allocator that uses global new, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator `new`
  - all deallocation calls operator `delete`.
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`  
*Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.  
Note: Deallocate not allowed to throw.*
- class `std::allocator< _Tp >`  
*The standard allocator, as per [20.4].  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*

### 2.33.1 Detailed Description

Classes encapsulating memory operations.

## 2.34 Atomics

### Classes

- struct `std::atomic< _Tp >`  
*atomic // 29.4.3, Generic atomic type, primary class template.*
- struct `std::atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `std::atomic< bool >`  
*Explicit specialization for bool.*
- struct `std::atomic< char >`

*Explicit specialization for char.*

- struct `std::atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `std::atomic< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `std::atomic< int >`  
*Explicit specialization for int.*
- struct `std::atomic< long >`  
*Explicit specialization for long.*
- struct `std::atomic< long long >`  
*Explicit specialization for long long.*
- struct `std::atomic< short >`  
*Explicit specialization for short.*
- struct `std::atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `std::atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `std::atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `std::atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::atomic< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `std::atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::atomic< void * >`  
*Explicit specialization for void\*.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Defines

- #define `_ATOMIC_CMPEXCHNG`(`_a`, `_e`, `_m`, `_x`)
- #define `_ATOMIC_LOAD`(`_a`, `_x`)
- #define `_ATOMIC_MODIFY`(`_a`, `_o`, `_m`, `_x`)
- #define `_ATOMIC_STORE`(`_a`, `_m`, `_x`)
- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base **std::atomic\_flag\_base**
- typedef \_\_atomic\_base< char > `atomic_char`
- typedef \_\_atomic\_base< char16\_t > `atomic_char16_t`
- typedef \_\_atomic\_base< char32\_t > `atomic_char32_t`
- typedef \_\_atomic\_base< int > `atomic_int`
- typedef `atomic_short` **std::atomic\_int\_fast16\_t**
- typedef `atomic_int` **std::atomic\_int\_fast32\_t**
- typedef `atomic_llong` **std::atomic\_int\_fast64\_t**
- typedef `atomic_schar` **std::atomic\_int\_fast8\_t**
- typedef `atomic_short` **std::atomic\_int\_least16\_t**
- typedef `atomic_int` **std::atomic\_int\_least32\_t**
- typedef `atomic_llong` **std::atomic\_int\_least64\_t**
- typedef `atomic_schar` **std::atomic\_int\_least8\_t**
- typedef `atomic_llong` **std::atomic\_intmax\_t**
- typedef `atomic_long` **std::atomic\_intptr\_t**
- typedef \_\_atomic\_base< long long > `atomic_llong`
- typedef \_\_atomic\_base< long > `atomic_long`
- typedef `atomic_long` **std::atomic\_ptrdiff\_t**
- typedef \_\_atomic\_base< signed char > `atomic_schar`
- typedef \_\_atomic\_base< short > `atomic_short`
- typedef `atomic_ulong` **std::atomic\_size\_t**
- typedef `atomic_long` **std::atomic\_ssize\_t**
- typedef \_\_atomic\_base< unsigned char > `atomic_uchar`
- typedef \_\_atomic\_base< unsigned int > `atomic_uint`
- typedef `atomic_ushort` **std::atomic\_uint\_fast16\_t**
- typedef `atomic_uint` **std::atomic\_uint\_fast32\_t**
- typedef `atomic_ullong` **std::atomic\_uint\_fast64\_t**
- typedef `atomic_uchar` **std::atomic\_uint\_fast8\_t**

- typedef `atomic_ushort` `std::atomic_uint_least16_t`
- typedef `atomic_uint` `std::atomic_uint_least32_t`
- typedef `atomic_ullong` `std::atomic_uint_least64_t`
- typedef `atomic_uchar` `std::atomic_uint_least8_t`
- typedef `atomic_ullong` `std::atomic_uintmax_t`
- typedef `atomic_ulong` `std::atomic_uintptr_t`
- typedef `__atomic_base< unsigned long long >` `atomic_ullong`
- typedef `__atomic_base< unsigned long >` `atomic_ulong`
- typedef `__atomic_base< unsigned short >` `atomic_ushort`
- typedef `__atomic_base< wchar_t >` `atomic_wchar_t`
- typedef enum `std::memory_order` `std::memory_order`

## Enumerations

- enum `std::memory_order` {  
`memory_order_relaxed`, `memory_order_consume`, `memory_order_-`  
`acquire`, `memory_order_release`,  
`memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- void `std::__atomic_flag_wait_explicit` (`__atomic_flag_base *`, `memory_order`)  
`_GLIBCXX_NOTHROW`
- `std::__attribute__` (`((__const__)) __atomic_flag_base *` `__atomic_flag_for_-`  
`address(const void *_z)` `_GLIBCXX_NOTHROW`
- `memory_order` `std::__calculate_memory_order` (`memory_order __m`)
- template<typename `_ITp`>  
bool `std::atomic_compare_exchange_strong` (`__atomic_base< _ITp > *__a`,  
`_ITp *__i1`, `_ITp __i2`)
- bool `std::atomic_compare_exchange_strong` (`atomic_address *__a`, `void **_-`  
`_v1`, `void *_v2`)
- bool `std::atomic_compare_exchange_strong` (`atomic_bool *__a`, `bool *__i1`,  
`bool __i2`)
- template<typename `_ITp`>  
bool `std::atomic_compare_exchange_strong_explicit` (`__atomic_base< _ITp`  
`> *__a`, `_ITp *__i1`, `_ITp __i2`, `memory_order __m1`, `memory_order __m2`)
- bool `std::atomic_compare_exchange_strong_explicit` (`atomic_address *__a`,  
`void **_v1`, `void *_v2`, `memory_order __m1`, `memory_order __m2`)
- bool `std::atomic_compare_exchange_strong_explicit` (`atomic_bool *__a`,  
`bool *__i1`, `bool __i2`, `memory_order __m1`, `memory_order __m2`)
- template<typename `_ITp`>  
bool `std::atomic_compare_exchange_weak` (`__atomic_base< _ITp > *__a`,  
`_ITp *__i1`, `_ITp __i2`)

- `bool std::atomic_compare_exchange_weak` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- `bool std::atomic_compare_exchange_weak` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- `bool std::atomic_compare_exchange_weak_explicit` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- `bool std::atomic_compare_exchange_weak_explicit` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- `void * std::atomic_exchange` (atomic\_address \*\_\_a, void \*\_\_v)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `bool std::atomic_exchange` (atomic\_bool \*\_\_a, bool \_\_i)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `void * std::atomic_exchange_explicit` (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- `bool std::atomic_exchange_explicit` (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- `void * std::atomic_fetch_add` (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `void * std::atomic_fetch_add_explicit` (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `void * std::atomic_fetch_sub` (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void std::atomic_flag_clear (__atomic_flag_base *__a)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `void std::atomic_flag_clear_explicit (__atomic_flag_base *, memory_order`  
`_GLIBCXX_NOTHROW)`
- `bool std::atomic_flag_test_and_set (__atomic_flag_base *__a)`
- `bool std::atomic_flag_test_and_set_explicit (__atomic_flag_base *,`  
`memory_order) _GLIBCXX_NOTHROW`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_`  
`order __m)`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a,`  
`memory_order __m)`
- `void * std::atomic_load_explicit (const atomic_address *__a, memory_order`  
`__m)`
- `bool std::atomic_load_explicit (const atomic_bool *__a, memory_order __m)`
- `void std::atomic_store (atomic_address *__a, void *__v)`
- `template<typename _ITp >`  
`void std::atomic_store (__atomic_base< _ITp > *__a, _ITp __i)`
- `void std::atomic_store (atomic_bool *__a, bool __i)`
- `void std::atomic_store_explicit (atomic_bool *__a, bool __i, memory_order _`  
`__m)`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `void std::atomic_store_explicit (atomic_address *__a, void *__v, memory_`  
`order __m)`

- `bool std::atomic< _Tp * >::compare_exchange_strong` (`_Tp * &`, `_Tp *`, `memory_order`, `memory_order`)
- `bool std::atomic< _Tp * >::compare_exchange_strong` (`_Tp * &`, `_Tp *`, `memory_order=memory_order_seq_cst`)
- `bool std::atomic< _Tp * >::compare_exchange_weak` (`_Tp * &`, `_Tp *`, `memory_order`, `memory_order`)
- `bool std::atomic< _Tp * >::compare_exchange_weak` (`_Tp * &`, `_Tp *`, `memory_order=memory_order_seq_cst`)
- `_Tp * std::atomic< _Tp * >::fetch_add` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp * std::atomic< _Tp * >::fetch_sub` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `template<typename _Tp > _Tp std::kill_dependency` (`_Tp __y`)

### 2.34.1 Detailed Description

Components for performing atomic operations.

### 2.34.2 Define Documentation

#### 2.34.2.1 `#define _GLIBCXX_ATOMIC_PROPERTY`

29.2 Lock-free Property

Definition at line 68 of file `atomic_base.h`.

### 2.34.3 Typedef Documentation

#### 2.34.3.1 `typedef __atomic_base<char> atomic_char`

`atomic_char`

Definition at line 71 of file `atomicfwd_cxx.h`.

#### 2.34.3.2 `typedef __atomic_base<char16_t> atomic_char16_t`

`atomic_char16_t`

Definition at line 107 of file `atomicfwd_cxx.h`.

**2.34.3.3 typedef \_\_atomic\_base<char32\_t> atomic\_char32\_t**

atomic\_char32\_t

Definition at line 110 of file atomicfwd\_cxx.h.

**2.34.3.4 typedef \_\_atomic\_base<int> atomic\_int**

atomic\_int

Definition at line 86 of file atomicfwd\_cxx.h.

**2.34.3.5 typedef \_\_atomic\_base<long long> atomic\_llong**

atomic\_llong

Definition at line 98 of file atomicfwd\_cxx.h.

**2.34.3.6 typedef \_\_atomic\_base<long> atomic\_long**

atomic\_long

Definition at line 92 of file atomicfwd\_cxx.h.

**2.34.3.7 typedef \_\_atomic\_base<signed char> atomic\_schar**

atomic\_schar

Definition at line 74 of file atomicfwd\_cxx.h.

**2.34.3.8 typedef \_\_atomic\_base<short> atomic\_short**

atomic\_short

Definition at line 80 of file atomicfwd\_cxx.h.

**2.34.3.9 typedef \_\_atomic\_base<unsigned char> atomic\_uchar**

atomic\_uchar

Definition at line 77 of file atomicfwd\_cxx.h.

**2.34.3.10 typedef \_\_atomic\_base<unsigned int> atomic\_uint**

atomic\_uint

Definition at line 89 of file atomicfwd\_cxx.h.

**2.34.3.11 typedef \_\_atomic\_base<unsigned long long> atomic\_ullong**

atomic\_ullong

Definition at line 101 of file atomicfwd\_cxx.h.

**2.34.3.12 typedef \_\_atomic\_base<unsigned long> atomic\_ulong**

atomic\_ulong

Definition at line 95 of file atomicfwd\_cxx.h.

**2.34.3.13 typedef \_\_atomic\_base<unsigned short> atomic\_ushort**

atomic\_ushort

Definition at line 83 of file atomicfwd\_cxx.h.

**2.34.3.14 typedef \_\_atomic\_base<wchar\_t> atomic\_wchar\_t**

atomic\_wchar\_t

Definition at line 104 of file atomicfwd\_cxx.h.

**2.34.3.15 typedef enum std::memory\_order std::memory\_order**

Enumeration for memory\_order.

**2.34.4 Enumeration Type Documentation****2.34.4.1 enum std::memory\_order**

Enumeration for memory\_order.

Definition at line 47 of file atomic\_base.h.

---

### 2.34.5 Function Documentation

**2.34.5.1** `template<typename _Tp > _Tp std::kill_dependency ( _Tp __y )`  
`[inline]`

`kill_dependency`

Definition at line 54 of file `atomic`.

## 2.35 Hashes

Collaboration diagram for Hashes:



### Classes

- struct `std::hash< _Tp >`  
*Primary class template hash.*
- struct `std::hash< _Tp * >`  
*Partial specializations for pointer types.*

### Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

### 2.35.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

## 2.36 Locales

### Classes

- class `std::codecvt<_InternT, _ExternT, _StateT >`  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class `std::ctype<_CharT >`  
*Primary class template ctype facet.*  
*This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class `std::ctype<char >`  
*The ctype<char> specialization.*  
*This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class `std::ctype<wchar_t >`  
*The ctype<wchar\_t> specialization.*  
*This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- class `std::locale`  
*Container class for localization functionality.*  
*The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.*
- class `std::locale::facet`  
*Localization functionality base class.*  
*The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*
- class `std::locale::id`  
*Facet ID class.*  
*The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.*
- class `std::messages<_CharT >`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- struct `std::messages_base`  
*Messages facet base class providing catalog typedef.*
- class `std::money_base`  
*Money format ordering data.  
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*
- class `std::money_get<_CharT, _InIter >`  
*Primary class template `money_get`.  
This facet encapsulates the code to parse and return a monetary amount from a string.*
- class `std::money_put<_CharT, _OutIter >`  
*Primary class template `money_put`.  
This facet encapsulates the code to format and output a monetary amount.*
- class `std::moneypunct<_CharT, _Intl >`  
*Primary class template `moneypunct`.  
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*
- class `std::num_get<_CharT, _InIter >`  
*Primary class template `num_get`.  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class `std::num_put<_CharT, _OutIter >`  
*Primary class template `num_put`.  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- class `std::numpunct<_CharT >`  
*Primary class template `numpunct`.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class `std::time_base`

*Time format ordering data.*

*This class provides an enum representing different orderings of time: day, month, and year.*

- class `std::time_get<_CharT, _InIter >`

*Primary class template `time_get`.*

*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*

- class `std::time_put<_CharT, _OutIter >`

*Primary class template `time_put`.*

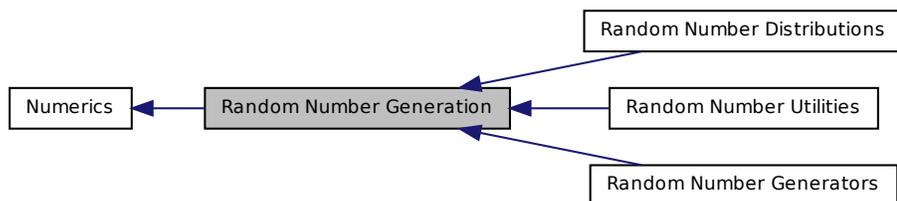
*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

### 2.36.1 Detailed Description

Classes and functions for internationalization and localization.

## 2.37 Random Number Generation

Collaboration diagram for Random Number Generation:



### Namespaces

- namespace `std::__detail`

## Modules

- [Random Number Generators](#)
- [Random Number Distributions](#)
- [Random Number Utilities](#)

## Functions

- `template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate\_canonical (_UniformRandomNumberGenerator & __g)`

### 2.37.1 Detailed Description

A facility for generating random numbers on selected distributions.

### 2.37.2 Function Documentation

**2.37.2.1** `template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate\_canonical ( _UniformRandomNumberGenerator & __g )`

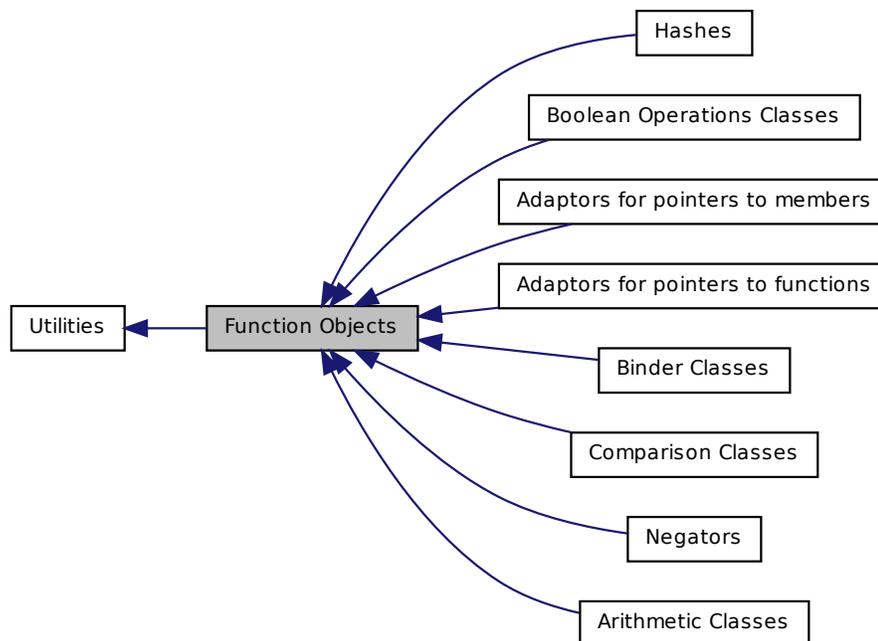
A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 2765 of file random.tcc.

References `std::log()`, and `std::min()`.

## 2.38 Function Objects

Collaboration diagram for Function Objects:



### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::function<_Res(_ArgTypes...)>`  
*Primary class template for `std::function`.  
 Polymorphic function wrapper.*
- class `std::reference_wrapper<_Tp >`  
*Primary class template for `reference_wrapper`.*
- struct `std::unary_function<_Arg, _Result >`

## Modules

- [Binder Classes](#)
- [Hashes](#)
- [Arithmetic Classes](#)
- [Comparison Classes](#)
- [Boolean Operations Classes](#)
- [Negators](#)
- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)

## Functions

- `template<typename _Tp, typename _Class >  
 _Mem_fn< _Tp _Class::* > std::mem\_fn (_Tp _Class::* __pm)`

### 2.38.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform(a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named [unary\\_function](#) and [binary\\_function](#). These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

## 2.38.2 Function Documentation

**2.38.2.1** `template<typename _Tp , typename _Class > _Mem_fn<_Tp  
_Class::*> std::mem_fn ( _Tp _Class::* __pm ) [inline]`

Returns a function object that forwards to the member pointer *pm*.

Definition at line 778 of file functional.

## 2.39 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



### Classes

- struct `std::divides<_Tp >`  
*One of the math functors.*
- struct `std::minus<_Tp >`  
*One of the math functors.*
- struct `std::modulus<_Tp >`  
*One of the math functors.*
- struct `std::multiplies<_Tp >`  
*One of the math functors.*
- struct `std::negate<_Tp >`  
*One of the math functors.*
- struct `std::plus<_Tp >`  
*One of the math functors.*

### 2.39.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

## 2.40 Comparison Classes

Collaboration diagram for Comparison Classes:



### Classes

- struct `std::equal_to< _Tp >`  
*One of the `comparison functors`.*
- struct `std::greater< _Tp >`  
*One of the `comparison functors`.*
- struct `std::greater_equal< _Tp >`  
*One of the `comparison functors`.*
- struct `std::less< _Tp >`  
*One of the `comparison functors`.*
- struct `std::less_equal< _Tp >`  
*One of the `comparison functors`.*
- struct `std::not_equal_to< _Tp >`  
*One of the `comparison functors`.*

### 2.40.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

## 2.41 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



### Classes

- struct `std::logical_and< _Tp >`  
*One of the Boolean operations functors.*
- struct `std::logical_not< _Tp >`  
*One of the Boolean operations functors.*
- struct `std::logical_or< _Tp >`  
*One of the Boolean operations functors.*

### 2.41.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

## 2.42 Negators

Collaboration diagram for Negators:



### Classes

- class `std::binary_negate< _Predicate >`  
*One of the [negation functors](#).*
- class `std::unary_negate< _Predicate >`  
*One of the [negation functors](#).*

### Functions

- `template<typename _Predicate >`  
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`

#### 2.42.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};
  
```

```
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (`i`) of `v` for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

## 2.42.2 Function Documentation

**2.42.2.1** `template<typename _Predicate > unary_negate<_Predicate>`  
`std::not1 ( const _Predicate & __pred ) [inline]`

One of the [negation functors](#).

Definition at line 364 of file `stl_function.h`.

**2.42.2.2** `template<typename _Predicate > binary_negate<_Predicate>`  
`std::not2 ( const _Predicate & __pred ) [inline]`

One of the [negation functors](#).

Definition at line 389 of file `stl_function.h`.

## 2.43 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



### Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`  
*One of the adaptors for function pointers.*

- class `std::pointer_to_unary_function<_Arg, _Result>`

*One of the [adaptors for function pointers](#).*

## Functions

- `template<typename _Arg, typename _Result>`  
`pointer_to_unary_function<_Arg, _Result>` `std::ptr_fun` (`_Result(*__x)(_`  
`Arg)`)
- `template<typename _Arg1, typename _Arg2, typename _Result>`  
`pointer_to_binary_function<_Arg1, _Arg2, _Result>` `std::ptr_fun` (`_`  
`Result(*__x)(_Arg1, _Arg2)`)

### 2.43.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes [unary\\_function](#) and [binary\\_function](#)). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor [pointer\\_to\\_unary\\_function](#) is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor [pointer\\_to\\_binary\\_function](#) does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

### 2.43.2 Function Documentation

- 2.43.2.1** `template<typename _Arg, typename _Result>`  
`pointer_to_unary_function<_Arg, _Result>` `std::ptr_fun` (`_`  
`Result(*)(_Arg) __x )` [`inline`]

One of the [adaptors for function pointers](#).

Definition at line 437 of file `stl_function.h`.

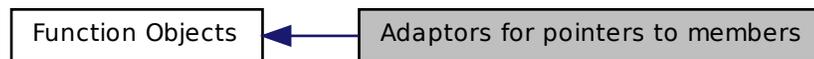
**2.43.2.2** `template<typename _Arg1, typename _Arg2, typename _Result >  
 pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (  
 _Result(*)(_Arg1, _Arg2) _x ) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 463 of file `stl_function.h`.

## 2.44 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



### Classes

- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_t< _Ret, _Tp >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`  
*One of the [adaptors for member /// pointers](#).*

- class `std::mem_fun_ref_t<_Ret, _Tp>`  
*One of the adaptors for member /// pointers.*
- class `std::mem_fun_t<_Ret, _Tp>`  
*One of the adaptors for member /// pointers.*

## Functions

- `template<typename _Ret, typename _Tp>`  
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*_f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*_f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*_f)(_-Arg))`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*_f)())`

### 2.44.1 Detailed Description

There are a total of  $8 = 2^3$  function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

## 2.45 Heap

Collaboration diagram for Heap:



## Functions

- `template<typename _RandomAccessIterator >`  
`bool std::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is\_heap\_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is\_heap\_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`

## 2.45.1 Function Documentation

**2.45.1.1** `template<typename _RandomAccessIterator > bool std::is_heap (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`  
`[inline]`

Determines whether a range is a heap.

### Parameters

*first* Start of range.

*last* End of range.

### Returns

True if range is a heap, false otherwise.

Definition at line 558 of file stl\_heap.h.

References std::is\_heap\_until().

**2.45.1.2** `template<typename _RandomAccessIterator , typename`  
`_Compare > bool std::is_heap ( _RandomAccessIterator __first,`  
`_RandomAccessIterator __last, _Compare __comp ) [inline]`

Determines whether a range is a heap using comparison functor.

### Parameters

*first* Start of range.

*last* End of range.

*comp* Comparison functor to use.

### Returns

True if range is a heap, false otherwise.

Definition at line 571 of file stl\_heap.h.

References std::is\_heap\_until().

**2.45.1.3** `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until ( _RandomAccessIterator`  
`__first, _RandomAccessIterator __last ) [inline]`

Search the end of a heap.

**Parameters**

*first* Start of range.

*last* End of range.

**Returns**

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap.

Definition at line 510 of file `stl_heap.h`.

References `std::distance()`.

```
2.45.1.4 template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::is_heap_until ( _RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp )
[inline]
```

Search the end of a heap using comparison functor.

**Parameters**

*first* Start of range.

*last* End of range.

*comp* Comparison functor to use.

**Returns**

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap. Comparisons are made using *comp*.

Definition at line 536 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

```
2.45.1.5 template<typename _RandomAccessIterator > void std::make_heap (
_RandomAccessIterator __first, _RandomAccessIterator __last )
```

Construct a heap over a range.

**Parameters**

*first* Start of heap.

*last* End of heap.

This operation makes the elements in [first,last) into a heap.

Definition at line 373 of file stl\_heap.h.

```
2.45.1.6 template<typename _RandomAccessIterator , typename _Compare  
> void std::make_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )
```

Construct a heap over a range using comparison functor.

**Parameters**

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation makes the elements in [first,last) into a heap. Comparisons are made using comp.

Definition at line 413 of file stl\_heap.h.

Referenced by std::\_\_heap\_select(), std::partial\_sort\_copy(), and std::priority\_queue<\_Tp, \_Sequence, \_Compare >::priority\_queue().

```
2.45.1.7 template<typename _RandomAccessIterator > void std::pop_heap (   
_RandomAccessIterator __first, _RandomAccessIterator __last )  
[inline]
```

Pop an element off a heap.

**Parameters**

*first* Start of heap.

*last* End of heap.

This operation pops the top of the heap. The elements first and last-1 are swapped and [first,last-1) is made into a heap.

Definition at line 276 of file stl\_heap.h.

**2.45.1.8** `template<typename _RandomAccessIterator, typename _Compare> void std::pop_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp ) [inline]`

Pop an element off a heap using comparison functor.

#### Parameters

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation pops the top of the heap. The elements *first* and *last*-1 are swapped and [*first*,*last*-1) is made into a heap. Comparisons are made using *comp*.

Definition at line 350 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`.

**2.45.1.9** `template<typename _RandomAccessIterator, typename _Compare> void std::push_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp ) [inline]`

Push an element onto a heap using comparison functor.

#### Parameters

*first* Start of heap.

*last* End of heap + element.

*comp* Comparison functor.

This operation pushes the element at *last*-1 onto the valid heap over the range [*first*,*last*-1). After completion, [*first*,*last*) is a valid heap. Compare operations are performed using *comp*.

Definition at line 203 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

**2.45.1.10** `template<typename _RandomAccessIterator> void std::push_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last ) [inline]`

Push an element onto a heap.

**Parameters**

*first* Start of heap.

*last* End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [first,last-1). After completion, [first,last) is a valid heap.

Definition at line 154 of file stl\_heap.h.

**2.45.1.11** `template<typename _RandomAccessIterator, typename _Compare  
> void std::sort_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )`

Sort a heap using comparison functor.

**Parameters**

*first* Start of heap.

*last* End of heap.

*comp* Comparison functor to use.

This operation sorts the valid heap in the range [first,last). Comparisons are made using comp.

Definition at line 481 of file stl\_heap.h.

Referenced by std::partial\_sort(), and std::partial\_sort\_copy().

**2.45.1.12** `template<typename _RandomAccessIterator > void std::sort_heap (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last )`

Sort a heap.

**Parameters**

*first* Start of heap.

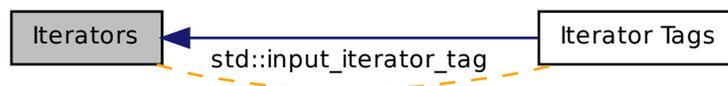
*last* End of heap.

This operation sorts the valid heap in the range [first,last).

Definition at line 452 of file stl\_heap.h.

## 2.46 Iterators

Collaboration diagram for Iterators:



### Classes

- class [std::back\\_insert\\_iterator< \\_Container >](#)  
*Turns assignment into insertion.*
- class [std::front\\_insert\\_iterator< \\_Container >](#)  
*Turns assignment into insertion.*
- struct [std::input\\_iterator\\_tag](#)  
*Marking input iterators.*
- class [std::insert\\_iterator< \\_Container >](#)  
*Turns assignment into insertion.*
- class [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >](#)  
*Provides input iterator semantics for streams.*
- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)  
*Provides input iterator semantics for streambufs.*
- struct [std::iterator< \\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)  
*Common iterator class.*
- struct [std::iterator\\_traits< \\_Iterator >](#)  
*Traits class for iterators.*
- struct [std::iterator\\_traits< \\_Tp \\* >](#)

*Partial specialization for pointer types.*

- struct `std::iterator_traits< const _Tp * >`  
*Partial specialization for const pointer types.*
- class `std::move_iterator< _Iterator >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`  
*Provides output iterator semantics for streams.*
- class `std::ostreambuf_iterator< _CharT, _Traits >`  
*Provides output iterator semantics for streambufs.*
- class `std::reverse_iterator< _Iterator >`

## Modules

- [Iterator Tags](#)

## Functions

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` `std::__copy_move_a2` (`_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` `std::__copy_move_a2` (`const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type` `std::__copy_move_a2` (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result`)
- `template<typename _Iter >`  
`iterator_traits< _Iter >::iterator_category` `std::__iterator_category` (`const _Iter &`)
- `template<typename _Container >`  
`back_insert_iterator< _Container >` `std::back_inserter` (`_Container & __x`)
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type` `std::copy` (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result`)

- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_`  
`iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT >`  
`__first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator >`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__`  
`__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _`  
`Iterator >::__difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator`  
`>::__difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::__difference_type std::operator- (const reverse_`  
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`  
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`

### 2.46.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

### 2.46.2 Function Documentation

#### 2.46.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category` `std::_iterator_category ( const _Iter & ) [inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 174 of file `stl_iterator_base_types.h`.

Referenced by `std::advance()`, `std::copy_n()`, `__gnu_cxx::copy_n()`, `std::distance()`, `__gnu_cxx::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::find_if_not()`, `std::partition()`, `std::reverse()`, `std::search_n()`, `std::uninitialized_copy_n()`, `__gnu_cxx::uninitialized_copy_n()`, and `std::unique_copy()`.

#### 2.46.2.2 `template<typename _Container > back_insert_iterator<_Container>` `std::back_inserter ( _Container & __x ) [inline]`

##### Parameters

`x` A container of arbitrary type.

##### Returns

An instance of `back_insert_iterator` working on `x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 471 of file `stl_iterator.h`.

**2.46.2.3** `template<typename _Container > front_insert_iterator<_Container>  
std::front_inserter ( _Container & __x ) [inline]`

#### Parameters

*x* A container of arbitrary type.

#### Returns

An instance of `front_insert_iterator` working on *x*.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 561 of file `stl_iterator.h`.

**2.46.2.4** `template<typename _Container , typename _Iterator >  
insert_iterator<_Container> std::inserter ( _Container & __x,  
_Iterator __i ) [inline]`

#### Parameters

*x* A container of arbitrary type.

#### Returns

An instance of `insert_iterator` working on *x*.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 675 of file `stl_iterator.h`.

**2.46.2.5** `template<class _Tp , class _CharT , class _Traits , class _Dist > bool  
std::operator!=( const istream_iterator<_Tp, _CharT, _Traits, _Dist  
> & __x, const istream_iterator<_Tp, _CharT, _Traits, _Dist > &  
__y ) [inline]`

Return false if *x* and *y* are both end or not end, or *x* and *y* are the same.

Definition at line 135 of file `stream_iterator.h`.

**2.46.2.6** `template<typename _Iterator > bool std::operator==( const reverse_iterator< _Iterator > & __x, const reverse_iterator< _Iterator > & __y ) [inline]`

#### Parameters

*x* A reverse\_iterator.

*y* A reverse\_iterator.

#### Returns

A simple bool.

Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 283 of file stl\_iterator.h.

References `std::reverse_iterator< _Iterator >::base()`.

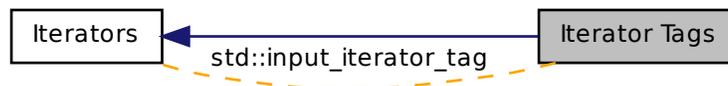
**2.46.2.7** `template<typename _Tp , typename _CharT , typename _Traits , typename _Dist > bool std::operator==( const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y ) [inline]`

Return true if *x* and *y* are both end or not end, or *x* and *y* are the same.

Definition at line 128 of file stream\_iterator.h.

## 2.47 Iterator Tags

Collaboration diagram for Iterator Tags:



## Classes

- struct `std::bidirectional_iterator_tag`  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct `std::forward_iterator_tag`  
*Forward iterators support a superset of input iterator operations.*
- struct `std::input_iterator_tag`  
*Marking input iterators.*
- struct `std::output_iterator_tag`  
*Marking output iterators.*
- struct `std::random_access_iterator_tag`  
*Random-access iterators support a superset of bidirectional /// iterator operations.*

### 2.47.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

## 2.48 Strings

Collaboration diagram for Strings:



## Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`

*Managing sequences of characters and character-like objects.*

## Typedefs

- typedef `basic_string<char>` `std::string`
- typedef `basic_string<char16_t>` `std::u16string`
- typedef `basic_string<char32_t>` `std::u32string`
- typedef `basic_string<wchar_t>` `std::wstring`

### 2.48.1 Typedef Documentation

#### 2.48.1.1 typedef `basic_string<char>` `std::string`

A string of `char`.

Definition at line 63 of file `stringfwd.h`.

#### 2.48.1.2 typedef `basic_string<char16_t>` `std::u16string`

A string of `char16_t`.

Definition at line 77 of file `stringfwd.h`.

#### 2.48.1.3 typedef `basic_string<char32_t>` `std::u32string`

A string of `char32_t`.

Definition at line 78 of file `stringfwd.h`.

#### 2.48.1.4 typedef `basic_string<wchar_t>` `std::wstring`

A string of `wchar_t`.

Definition at line 68 of file `stringfwd.h`.

## 2.49 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



### Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`  
*An abstract basic hash-based associative container.*
- class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`  
*An abstract basic tree-like (tree, trie) associative container.*
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete collision-chaining hash-based associative container.*
- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`  
*An abstract basic associative container.*
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete general-probing hash-based associative container.*
- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`  
*A list-update based associative container.*
- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`  
*A concrete basic tree-based associative container.*

- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

*A concrete basic trie-based associative container.*

## Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

### 2.49.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in `std` (except for some points where it differs by design).

For details, see: [http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb\\_ds/index.html](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

## 2.50 Random Number Generators

Collaboration diagram for Random Number Generators:



## Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`  
*A model of a linear congruential random number generator.*
- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.*

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` **`std::minstd_rand`**
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` **`std::minstd_rand0`**
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` **`std::mt19937`**
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` **`std::mt19937_64`**
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

## Functions

- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m>  
**`bool std::operator!=`**(const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_rhs)

- `template<typename _RandomNumberEngine, size_t __k>`  
`bool std::operator!= (const std::shuffle_order_engine< _-`  
`RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine<`  
`_RandomNumberEngine, __k > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`  
`UIntType __f>`  
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _-`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`  
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_-`  
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine<`  
`_RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`  
`, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__x)`

### 2.50.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.
-------------------

Table 2.1: Random Number Generator Requirements

## 2.50.2 Typedef Documentation

### 2.50.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1473 of file random.h.

### 2.50.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1467 of file random.h.

### 2.50.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1489 of file random.h.

### 2.50.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1501 of file random.h.

### 2.50.3 Function Documentation

**2.50.3.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool std::operator!=( const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs ) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 331 of file random.h.

**2.50.3.2** `template<typename _RandomNumberEngine, size_t __k> bool std::operator!=( const std::shuffle_order_engine< _RandomNumberEngine, __k > & __lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs ) [inline]`

Compares two `shuffle_order_engine` random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A `shuffle_order_engine` random number generator object.

`__rhs` Another `shuffle_order_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1456 of file random.h.

```

2.50.3.3 template<typename _UIntType, size_t __w, size_t __n, size_t __m,
size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f> bool std::operator!=( const std::mersenne_twister_engine<
_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,
__f > & __lhs, const std::mersenne_twister_engine< _UIntType, __w,
__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs )
[inline]

```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for inequality.

#### Parameters

**\_\_lhs** A % mersenne\_twister\_engine random number generator object.

**\_\_rhs** Another % mersenne\_twister\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 567 of file random.h.

```

2.50.3.4 template<typename _RandomNumberEngine, size_t
__w, typename _UIntType > bool std::operator!=( const
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType > & __lhs, const std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType > & __rhs ) [inline]

```

Compares two independent\_bits\_engine random number generator objects of the same type for inequality.

#### Parameters

**\_\_lhs** A independent\_bits\_engine random number generator object.

**\_\_rhs** Another independent\_bits\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1200 of file random.h.

```

2.50.3.5 template<typename _RandomNumberEngine , size_t __p, size_t
__r> bool std::operator!=( const std::discard_block_engine<
_RandomNumberEngine, __p, __r > & __lhs, const
std::discard_block_engine< _RandomNumberEngine, __p, __r > &
__rhs ) [inline]

```

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A `discard_block_engine` random number generator object.

`__rhs` Another `discard_block_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 998 of file `random.h`.

```

2.50.3.6 template<typename _UIntType , size_t __w, size_t __s, size_t __r> bool
std::operator!=( const std::subtract_with_carry_engine< _UIntType,
__w, __s, __r > & __lhs, const std::subtract_with_carry_engine<
_UIntType, __w, __s, __r > & __rhs ) [inline]

```

Compares two `% subtract_with_carry_engine` random number generator objects of the same type for inequality.

#### Parameters

`__lhs` A `% subtract_with_carry_engine` random number generator object.

`__rhs` Another `% subtract_with_carry_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 770 of file `random.h`.

```

2.50.3.7 template<typename _RandomNumberEngine , size_t __w,
typename _UIntType , typename _CharT , typename _Traits
> std::basic_ostream<_CharT, _Traits>& std::operator<<
( std::basic_ostream<_CharT, _Traits > & __os, const
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType > & __x )

```

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `independent_bits_engine` random number generator engine.

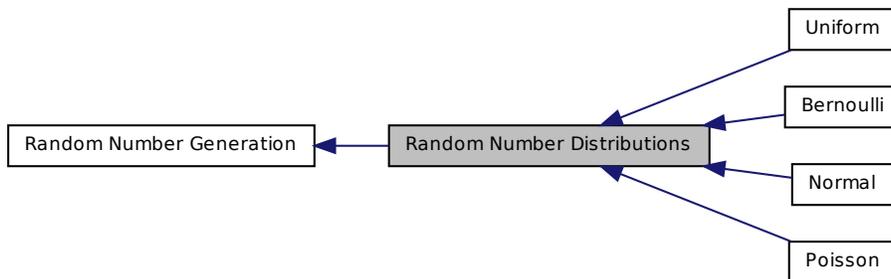
#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1219 of file `random.h`.

## 2.51 Random Number Distributions

Collaboration diagram for Random Number Distributions:



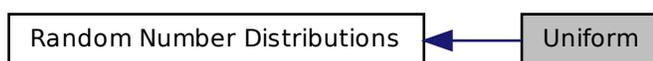
#### Modules

- [Uniform](#)

- [Normal](#)
- [Bernoulli](#)
- [Poisson](#)

## 2.52 Uniform

Collaboration diagram for Uniform:



### Classes

- class `std::uniform_int_distribution<_IntType>`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class `std::uniform_real_distribution<_RealType>`  
*Uniform continuous distribution for random numbers.*

### Functions

- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_int_distribution<_IntType > &__d1,`  
`const std::uniform_int_distribution<_IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution<_IntType > &__d1,`  
`const std::uniform_real_distribution<_IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits > & std::operator<< (std::basic_`  
`ostream<_CharT, _Traits > &, const std::uniform_real_distribution<_`  
`RealType > &)`

- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType >  
bool std::operator== (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator== (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`

## 2.52.1 Function Documentation

**2.52.1.1** `template<typename _IntType > bool std::operator!=( const std::uniform_int_distribution< _IntType > & __d1, const std::uniform_int_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1758 of file random.h.

**2.52.1.2** `template<typename _IntType > bool std::operator!=( const std::uniform_real_distribution< _IntType > & __d1, const std::uniform_real_distribution< _IntType > & __d2 ) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 1939 of file random.h.

**2.52.1.3** `template<typename _RealType, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::uniform_real_distribution< _RealType > & __x )`

Inserts a uniform\_real\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.

`__x` A `uniform_real_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 900 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

```
2.52.1.4 template<typename _IntType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
( std::basic_ostream< _CharT, _Traits > & __os, const
std::uniform_int_distribution< _IntType > & __x )
```

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

**Parameters**

`__os` An output stream.

`__x` A `uniform_int_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 858 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

```
2.52.1.5 template<typename _IntType > bool std::operator==( const
std::uniform_real_distribution< _IntType > & __d1, const
std::uniform_real_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform real distributions have the same parameters.

Definition at line 1929 of file `random.h`.

References `std::uniform_real_distribution< _RealType >::param()`.

```
2.52.1.6 template<typename _IntType > bool std::operator==( const
std::uniform_int_distribution< _IntType > & __d1, const
std::uniform_int_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform integer distributions have the same parameters.

Definition at line 1748 of file random.h.

References `std::uniform_int_distribution<_IntType>::param()`.

```
2.52.1.7 template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream<_CharT, _Traits > &
std::operator>> ( std::basic_istream<_CharT, _Traits > & __is,
std::uniform_real_distribution<_RealType > & __x )
```

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `uniform_real_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 924 of file random.tcc.

References `std::ios_base::flags()`, `std::uniform_real_distribution<_RealType>::param()`, and `std::skipws()`.

```
2.52.1.8 template<typename _IntType , typename _CharT , typename
_Traits > std::basic_istream<_CharT, _Traits > &
std::operator>> ( std::basic_istream<_CharT, _Traits > & __is,
std::uniform_int_distribution<_IntType > & __x )
```

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `uniform_int_distribution` random number generator engine.

#### Returns

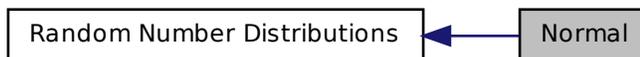
The input stream with `__x` extracted or in an error state.

Definition at line 879 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution<_IntType>::param()`, and `std::skipws()`.

## 2.53 Normal

Collaboration diagram for Normal:



### Classes

- class `std::cauchy_distribution<_RealType>`  
A *cauchy\_distribution* random number distribution.
- class `std::chi_squared_distribution<_RealType>`  
A *chi\_squared\_distribution* random number distribution.
- class `std::fisher_f_distribution<_RealType>`  
A *fisher\_f\_distribution* random number distribution.
- class `std::gamma_distribution<_RealType>`  
A *gamma* continuous distribution for random numbers.
- class `std::lognormal_distribution<_RealType>`  
A *lognormal\_distribution* random number distribution.
- class `std::normal_distribution<_RealType>`  
A *normal* continuous distribution for random numbers.
- class `std::student_t_distribution<_RealType>`  
A *student\_t\_distribution* random number distribution.

### Functions

- template<typename `_RealType`>  
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &\_\_d1,  
const `std::normal_distribution<_RealType>` &\_\_d2)

- `template<typename _RealType >`  
`bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1,`  
`const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`  
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1,`  
`const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`  
`&)`
- `template<typename _RealType >`  
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

### 2.53.1 Function Documentation

**2.53.1.1** `template<typename _RealType > bool std::operator!= ( const`  
`std::normal_distribution< _RealType > & __d1, const`  
`std::normal_distribution< _RealType > & __d2 ) [inline]`

Return true if two normal distributions are different.

Definition at line 2159 of file random.h.

**2.53.1.2** `template<typename _RealType > bool std::operator!= ( const`  
`std::lognormal_distribution< _RealType > & __d1, const`  
`std::lognormal_distribution< _RealType > & __d2 ) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2335 of file random.h.

**2.53.1.3** `template<typename _RealType > bool std::operator!=( const  
std::chi_squared_distribution< _RealType > & __d1, const  
std::chi_squared_distribution< _RealType > & __d2 ) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2692 of file random.h.

**2.53.1.4** `template<typename _RealType > bool std::operator!=( const  
std::fisher_f_distribution< _RealType > & __d1, const  
std::fisher_f_distribution< _RealType > & __d2 ) [inline]`

Return true if two Fisher f distributions are diferent.

Definition at line 3047 of file random.h.

**2.53.1.5** `template<typename _RealType > bool std::operator!=( const  
std::student_t_distribution< _RealType > & __d1, const  
std::student_t_distribution< _RealType > & __d2 ) [inline]`

Return true if two Student t distributions are different.

Definition at line 3221 of file random.h.

**2.53.1.6** `template<typename _RealType > bool std::operator!=(  
const std::cauchy_distribution< _RealType > & __d1, const  
std::cauchy_distribution< _RealType > & __d2 ) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 2831 of file random.h.

**2.53.1.7** `template<typename _RealType > bool std::operator!=( const  
std::gamma_distribution< _RealType > & __d1, const  
std::gamma_distribution< _RealType > & __d2 ) [inline]`

Return true if two gamma distributions are different.

Definition at line 2527 of file random.h.

**2.53.1.8** `template<typename _RealType, typename _CharT, typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::cauchy_distribution< _RealType > & __x )`

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `cauchy_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1822 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

**2.53.1.9** `template<typename _RealType > bool std::operator==(  
const std::cauchy_distribution< _RealType > & __d1, const  
std::cauchy_distribution< _RealType > & __d2 ) [inline]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 2821 of file `random.h`.

References `std::cauchy_distribution< _RealType >::param()`.

**2.53.1.10** `template<typename _RealType, typename _CharT,  
typename _Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>>( std::basic_istream< _CharT, _Traits > & __is,  
std::cauchy_distribution< _RealType > & __x )`

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `cauchy_distribution` random number generator engine.

#### Returns

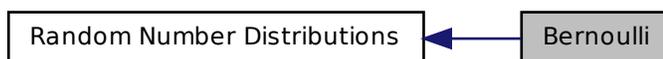
The input stream with `__x` extracted or in an error state.

Definition at line 1846 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.

## 2.54 Bernoulli

Collaboration diagram for Bernoulli:



### Classes

- class [std::bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- class [std::binomial\\_distribution< \\_IntType >](#)  
*A discrete binomial random number distribution.*
- class [std::geometric\\_distribution< \\_IntType >](#)  
*A discrete geometric random number distribution.*
- class [std::negative\\_binomial\\_distribution< \\_IntType >](#)  
*A negative\_binomial\_distribution random number distribution.*

### Functions

- `bool std::operator!= (const std::bernoulli\_distribution &__d1, const std::bernoulli\_distribution &__d2)`
- `template<typename _IntType > bool std::operator!= (const std::binomial\_distribution< _IntType > &__d1, const std::binomial\_distribution< _IntType > &__d2)`

- `template<typename _IntType >`  
`bool std::operator!= (const std::negative_binomial_distribution< _IntType >`  
`&__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1,`  
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`  
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`  
`&)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`  
`ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `bool std::operator== (const std::bernoulli_distribution &__d1, const`  
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`  
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

## 2.54.1 Function Documentation

### 2.54.1.1 `bool std::operator!= ( const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2 ) [inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3369 of file random.h.

### 2.54.1.2 `template<typename _IntType > bool std::operator!= ( const std::binomial_distribution< _IntType > & __d1, const std::binomial_distribution< _IntType > & __d2 ) [inline]`

Return true if two binomial distributions are different.

Definition at line 3606 of file random.h.

**2.54.1.3** `template<typename _IntType > bool std::operator!=( const  
std::negative_binomial_distribution< _IntType > & __d1, const  
std::negative_binomial_distribution< _IntType > & __d2 )  
[inline]`

Return true if two negative binomial distributions are different.

Definition at line 3952 of file random.h.

**2.54.1.4** `template<typename _IntType > bool std::operator!=( const  
std::geometric_distribution< _IntType > & __d1, const  
std::geometric_distribution< _IntType > & __d2 ) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 3748 of file random.h.

**2.54.1.5** `template<typename _IntType , typename _CharT , typename _Traits  
> std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::geometric_distribution< _IntType > & __x )`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `geometric_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 995 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.54.1.6** `template<typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<  
( std::basic_ostream< _CharT, _Traits > & __os, const  
std::bernoulli_distribution & __x )`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

- `__os` An output stream.
- `__x` A `bernoulli_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 945 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.54.1.7** `bool std::operator==( const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2 ) [inline]`

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3360 of file `random.h`.

References `std::bernoulli_distribution::param()`.

**2.54.1.8** `template<typename _IntType > bool std::operator==( const std::geometric_distribution< _IntType > & __d1, const std::geometric_distribution< _IntType > & __d2 ) [inline]`

Return true if two geometric distributions have the same parameters.

Definition at line 3738 of file `random.h`.

References `std::geometric_distribution< _IntType >::param()`.

**2.54.1.9** `template<typename _IntType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>>( std::basic_istream< _CharT, _Traits > & __is, std::geometric_distribution< _IntType > & __x )`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

- `__is` An input stream.
- `__x` A `geometric_distribution` random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

Definition at line 1019 of file random.tcc.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType >::param()`, and `std::skipws()`.

```
2.54.1.10 template<typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& std::operator>>
( std::basic_istream< _CharT, _Traits > & __is,
  std::bernoulli_distribution & __x )
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `bernoulli_distribution` random number generator engine.

#### Returns

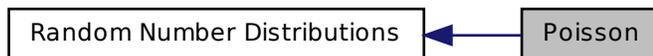
The input stream with `__x` extracted or in an error state.

Definition at line 3399 of file random.h.

References `std::bernoulli_distribution::param()`.

## 2.55 Poisson

Collaboration diagram for Poisson:



#### Classes

- class `std::discrete_distribution< _IntType >`

*A `discrete_distribution` random number distribution.*

- class `std::exponential_distribution<_RealType>`  
*An exponential continuous distribution for random numbers.*
- class `std::extreme_value_distribution<_RealType>`  
*A `extreme_value_distribution` random number distribution.*
- class `std::piecewise_constant_distribution<_RealType>`  
*A `piecewise_constant_distribution` random number distribution.*
- class `std::piecewise_linear_distribution<_RealType>`  
*A `piecewise_linear_distribution` random number distribution.*
- class `std::poisson_distribution<_IntType>`  
*A discrete Poisson random number distribution.*
- class `std::weibull_distribution<_RealType>`  
*A `weibull_distribution` random number distribution.*

## Functions

- `template<typename _IntType >`  
`bool std::operator!= (const std::poisson_distribution<_IntType> &__d1, const std::poisson_distribution<_IntType> &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution<_RealType> &__d1, const std::extreme_value_distribution<_RealType> &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution<_RealType> &__d1, const std::piecewise_constant_distribution<_RealType> &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution<_RealType> &__d1, const std::piecewise_linear_distribution<_RealType> &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::exponential_distribution<_RealType> &__d1, const std::exponential_distribution<_RealType> &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution<_RealType> &__d1, const std::weibull_distribution<_RealType> &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution<_IntType> &__d1, const std::discrete_distribution<_IntType> &__d2)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _RealType >`  
`bool std::operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`

### 2.55.1 Function Documentation

**2.55.1.1** `template<typename _IntType > bool std::operator!=( const  
std::poisson_distribution< _IntType > & __d1, const  
std::poisson_distribution< _IntType > & __d2 ) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4140 of file random.h.

**2.55.1.2** `template<typename _RealType > bool std::operator!=( const  
std::extreme_value_distribution< _RealType > & __d1, const  
std::extreme_value_distribution< _RealType > & __d2 ) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 4640 of file random.h.

**2.55.1.3** `template<typename _RealType > bool std::operator!=( const  
std::piecewise_constant_distribution< _RealType > & __d1, const  
std::piecewise_constant_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5070 of file random.h.

**2.55.1.4** `template<typename _RealType > bool std::operator!=( const  
std::piecewise_linear_distribution< _RealType > & __d1, const  
std::piecewise_linear_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 5281 of file random.h.

**2.55.1.5** `template<typename _RealType > bool std::operator!=( const  
std::exponential_distribution< _RealType > & __d1, const  
std::exponential_distribution< _RealType > & __d2 ) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4290 of file random.h.

**2.55.1.6** `template<typename _RealType > bool std::operator!=( const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2 ) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 4465 of file random.h.

**2.55.1.7** `template<typename _IntType > bool std::operator!=( const std::discrete_distribution< _IntType > & __d1, const std::discrete_distribution< _IntType > & __d2 ) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 4862 of file random.h.

**2.55.1.8** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT , _Traits > & std::operator<< ( std::basic_ostream< _CharT , _Traits > & __os, const std::weibull_distribution< _RealType > & __x )`

Inserts a weibull\_distribution random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A weibull\_distribution random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2075 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.55.1.9** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT , _Traits > & std::operator<< ( std::basic_ostream< _CharT , _Traits > & __os, const std::exponential_distribution< _RealType > & __x )`

Inserts a exponential\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.

`__x` A exponential\_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1555 of file random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

**2.55.1.10** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::extreme_value_distribution< _RealType > & __x )`

Inserts a extreme\_value\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.

`__x` A extreme\_value\_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2132 of file random.tcc.

References `std::left()`, and `std::scientific()`.

**2.55.1.11** `template<typename _RealType > bool std::operator==( const std::exponential_distribution< _RealType > & __d1, const std::exponential_distribution< _RealType > & __d2 ) [inline]`

Return true if two exponential distributions have the same parameters.

Definition at line 4280 of file random.h.

References `std::exponential_distribution< _RealType >::param()`.

**2.55.1.12** `template<typename _RealType > bool std::operator==( const  
std::piecewise_constant_distribution< _RealType > & __d1, const  
std::piecewise_constant_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5060 of file random.h.

References `std::piecewise_constant_distribution< _RealType >::param()`.

**2.55.1.13** `template<typename _RealType > bool std::operator==( const  
std::piecewise_linear_distribution< _RealType > & __d1, const  
std::piecewise_linear_distribution< _RealType > & __d2 )  
[inline]`

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5271 of file random.h.

References `std::piecewise_linear_distribution< _RealType >::param()`.

**2.55.1.14** `template<typename _IntType > bool std::operator==( const  
std::discrete_distribution< _IntType > & __d1, const  
std::discrete_distribution< _IntType > & __d2 ) [inline]`

Return true if two discrete distributions have the same parameters.

Definition at line 4852 of file random.h.

References `std::discrete_distribution< _IntType >::param()`.

**2.55.1.15** `template<typename _RealType > bool std::operator==( const  
std::extreme_value_distribution< _RealType > & __d1, const  
std::extreme_value_distribution< _RealType > & __d2 )  
[inline]`

Return true if two extreme value distributions have the same parameters.

Definition at line 4630 of file random.h.

References `std::extreme_value_distribution< _RealType >::param()`.

**2.55.1.16** `template<typename _RealType > bool std::operator==( const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2 ) [inline]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4455 of file random.h.

References `std::weibull_distribution< _RealType >::param()`.

**2.55.1.17** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>>( std::basic_istream< _CharT, _Traits > & __is, std::exponential_distribution< _RealType > & __x )`

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `exponential_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1578 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution< _RealType >::param()`, and `std::skipws()`.

**2.55.1.18** `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>>( std::basic_istream< _CharT, _Traits > & __is, std::weibull_distribution< _RealType > & __x )`

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `weibull_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2099 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution< _RealType >::param()`, and `std::skipws()`.

**2.55.1.19** `template<typename _RealType , typename _CharT ,  
typename _Traits > std::basic_istream< _CharT, _Traits > &  
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is,  
std::extreme_value_distribution< _RealType > & __x )`

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `extreme_value_distribution` random number generator engine.

#### Returns

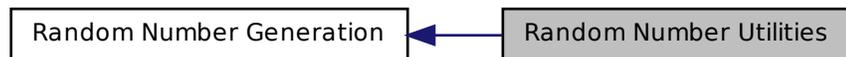
The input stream with `__x` extracted or in an error state.

Definition at line 2156 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution< _RealType >::param()`, and `std::skipws()`.

## 2.56 Random Number Utilities

Collaboration diagram for Random Number Utilities:



#### Classes

- class [std::seed\\_seq](#)

The [seed\\_seq](#) class generates sequences of seeds for random number generators.



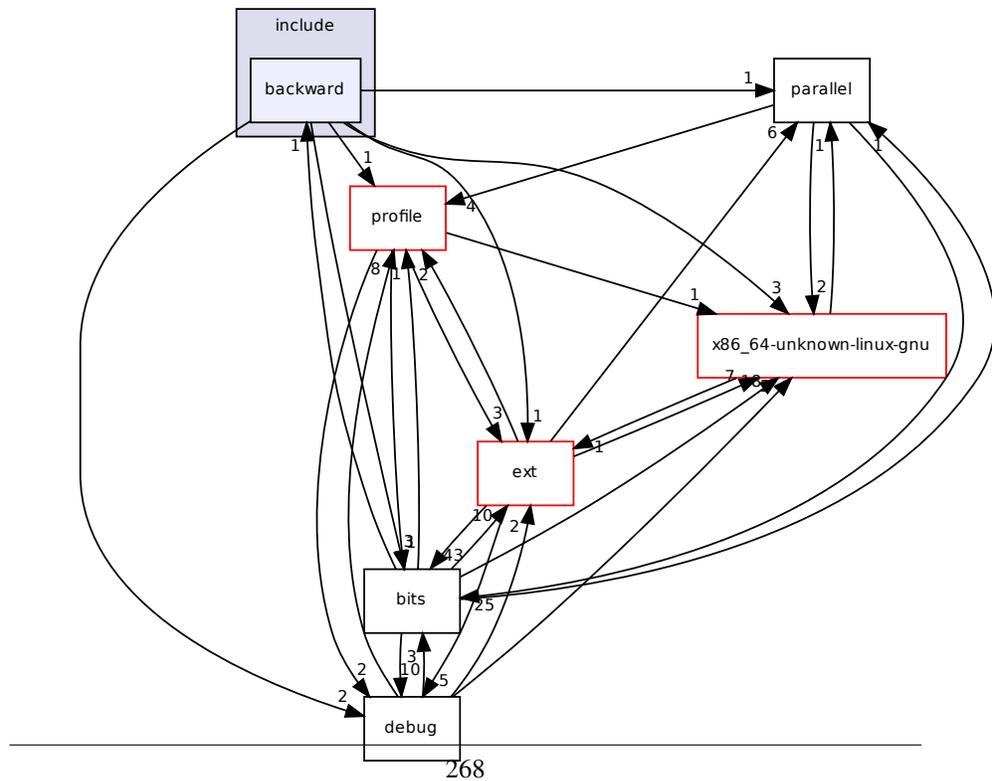


## Chapter 3

# Directory Documentation

### 3.1 include/backward/ Directory Reference

Directory dependency graph for include/backward/:

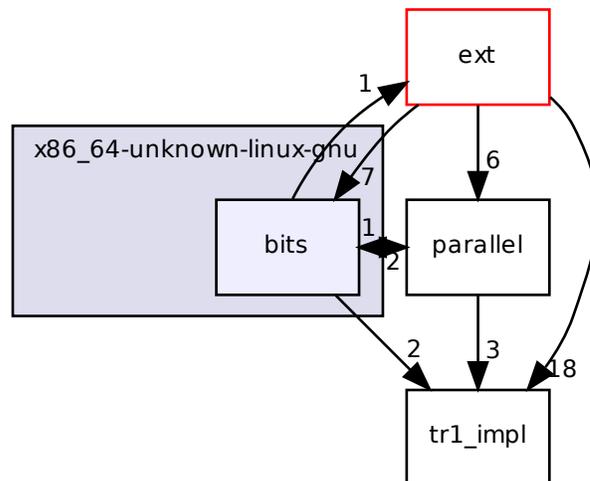


## Files

- file [auto\\_ptr.h](#)
- file [backward\\_warning.h](#)
- file [binders.h](#)
- file [hash\\_fun.h](#)
- file [hash\\_map](#)
- file [hash\\_set](#)
- file [backward/hashtable.h](#)
- file [strstream](#)

## 3.2 include/x86\_64-unknown-linux-gnu/bits/ Directory Reference

Directory dependency graph for include/x86\_64-unknown-linux-gnu/bits/:



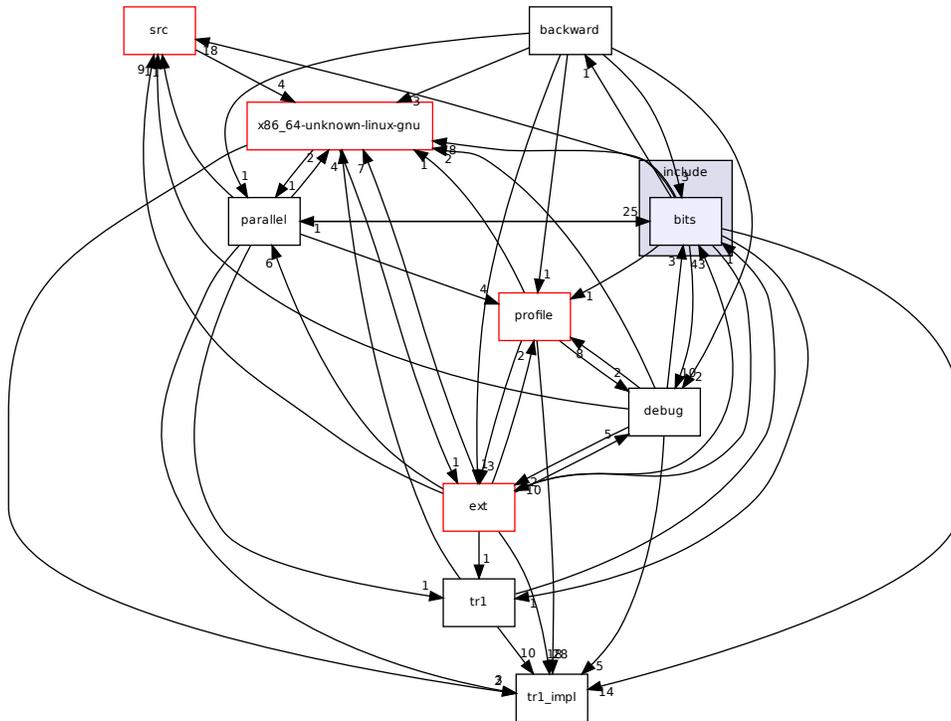
## Files

- file [atomic\\_word.h](#)

- file [basic\\_file.h](#)
- file [c++allocator.h](#)
- file [c++config.h](#)
- file [c++io.h](#)
- file [c++locale.h](#)
- file [c++locale\\_internal.h](#)
- file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#)
- file [cpu\\_defines.h](#)
- file [ctype\\_base.h](#)
- file [ctype\\_inline.h](#)
- file [ctype\\_noninline.h](#)
- file [cxxabi\\_tweaks.h](#)
- file [error\\_constants.h](#)
- file **gthr-default.h**
- file **gthr-posix.h**
- file **gthr-single.h**
- file **gthr-tpf.h**
- file **gthr.h**
- file [messages\\_members.h](#)
- file [os\\_defines.h](#)
- file [time\\_members.h](#)

### 3.3 include/bits/ Directory Reference

Directory dependency graph for include/bits/:



#### Files

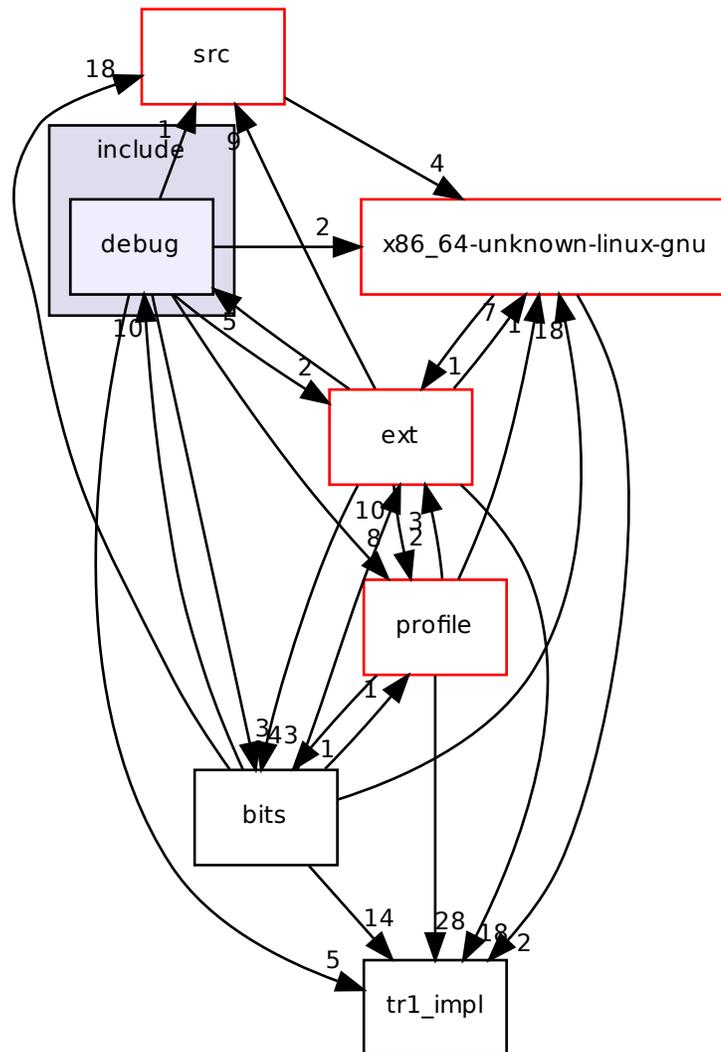
- file [bits/algorithmfwd.h](#)
- file [allocator.h](#)
- file [atomic\\_0.h](#)
- file [atomic\\_2.h](#)
- file [atomic\\_base.h](#)
- file [atomicfwd\\_c.h](#)
- file [atomicfwd\\_cxx.h](#)
- file [basic\\_ios.h](#)
- file [basic\\_ios.tcc](#)
- file [basic\\_string.h](#)

- file [basic\\_string.tcc](#)
- file [boost\\_concept\\_check.h](#)
- file [c++0x\\_warning.h](#)
- file [char\\_traits.h](#)
- file [cmath.tcc](#)
- file [codecvt.h](#)
- file [concept\\_check.h](#)
- file [cpp\\_type\\_traits.h](#)
- file [deque.tcc](#)
- file [forward\\_list.h](#)
- file [forward\\_list.tcc](#)
- file [fstream.tcc](#)
- file [functexcept.h](#)
- file [functional\\_hash.h](#)
- file [gslice.h](#)
- file [gslice\\_array.h](#)
- file [bits/hashtable.h](#)
- file [hashtable\\_policy.h](#)
- file [indirect\\_array.h](#)
- file [ios\\_base.h](#)
- file [istream.tcc](#)
- file [list.tcc](#)
- file [locale\\_classes.h](#)
- file [locale\\_classes.tcc](#)
- file [locale\\_facets.h](#)
- file [locale\\_facets.tcc](#)
- file [locale\\_facets\\_nonio.h](#)
- file [locale\\_facets\\_nonio.tcc](#)
- file [localefwd.h](#)
- file [mask\\_array.h](#)
- file [move.h](#)
- file [ostream.tcc](#)
- file [ostream\\_insert.h](#)
- file [postypes.h](#)
- file [random.h](#)
- file [random.tcc](#)
- file [shared\\_ptr.h](#)
- file [shared\\_ptr\\_base.h](#)
- file [slice\\_array.h](#)
- file [sstream.tcc](#)
- file [stl\\_algo.h](#)
- file [stl\\_algobase.h](#)
- file [stl\\_bvector.h](#)

- file [stl\\_construct.h](#)
- file [stl\\_deque.h](#)
- file [stl\\_function.h](#)
- file [stl\\_heap.h](#)
- file [stl\\_iterator.h](#)
- file [stl\\_iterator\\_base\\_funcs.h](#)
- file [stl\\_iterator\\_base\\_types.h](#)
- file [stl\\_list.h](#)
- file [stl\\_map.h](#)
- file [stl\\_multimap.h](#)
- file [stl\\_multiset.h](#)
- file [stl\\_numeric.h](#)
- file [stl\\_pair.h](#)
- file [stl\\_queue.h](#)
- file [stl\\_raw\\_storage\\_iter.h](#)
- file [stl\\_relops.h](#)
- file [stl\\_set.h](#)
- file [stl\\_stack.h](#)
- file [stl\\_tempbuf.h](#)
- file [stl\\_tree.h](#)
- file [stl\\_uninitialized.h](#)
- file [stl\\_vector.h](#)
- file [stream\\_iterator.h](#)
- file [streambuf.tcc](#)
- file [streambuf\\_iterator.h](#)
- file [stringfwd.h](#)
- file [unique\\_ptr.h](#)
- file [unordered\\_map.h](#)
- file [unordered\\_set.h](#)
- file [valarray\\_after.h](#)
- file [valarray\\_array.h](#)
- file [valarray\\_array.tcc](#)
- file [valarray\\_before.h](#)
- file [vector.tcc](#)

### 3.4 include/debug/ Directory Reference

Directory dependency graph for include/debug/:

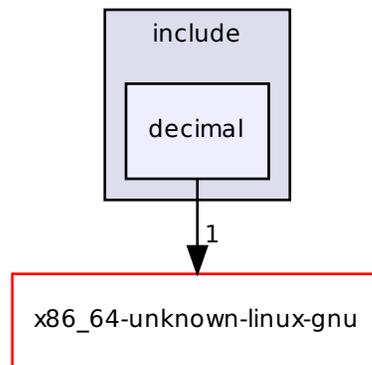


## Files

- file [debug/bitset](#)
- file [debug.h](#)
- file [debug/deque](#)
- file [formatter.h](#)
- file [functions.h](#)
- file [debug/list](#)
- file [macros.h](#)
- file [debug/map](#)
- file [debug/map.h](#)
- file [debug/multimap.h](#)
- file [debug/multiset.h](#)
- file [safe\\_base.h](#)
- file [safe\\_iterator.h](#)
- file [safe\\_iterator.tcc](#)
- file [safe\\_sequence.h](#)
- file [debug/set](#)
- file [debug/set.h](#)
- file [debug/string](#)
- file [debug/unordered\\_map](#)
- file [debug/unordered\\_set](#)
- file [debug/vector](#)

### 3.5 include/decimal/ Directory Reference

Directory dependency graph for include/decimal/:

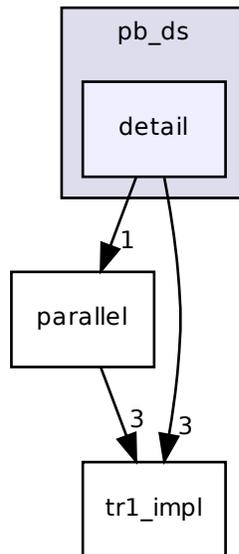


#### Files

- file [decimal](#)

### 3.6 include/ext/pb\_ds/detail/ Directory Reference

Directory dependency graph for include/ext/pb\_ds/detail/:

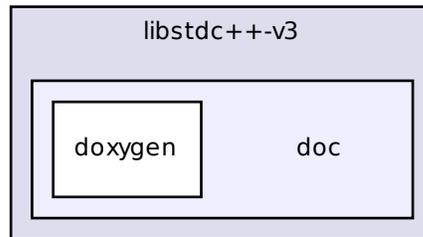


#### Files

- file [basic\\_types.hpp](#)
- file [cond\\_dealtor.hpp](#)
- file [constructors\\_destructor\\_fn\\_imps.hpp](#)
- file [container\\_base\\_dispatch.hpp](#)
- file [debug\\_map\\_base.hpp](#)
- file [priority\\_queue\\_base\\_dispatch.hpp](#)
- file [standard\\_policies.hpp](#)
- file [tree\\_trace\\_base.hpp](#)
- file [type\\_utils.hpp](#)
- file [types\\_traits.hpp](#)

### 3.7 /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/:

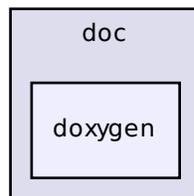


#### Directories

- directory [doxygen](#)

### 3.8 /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/doxygen/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/doxygen/:

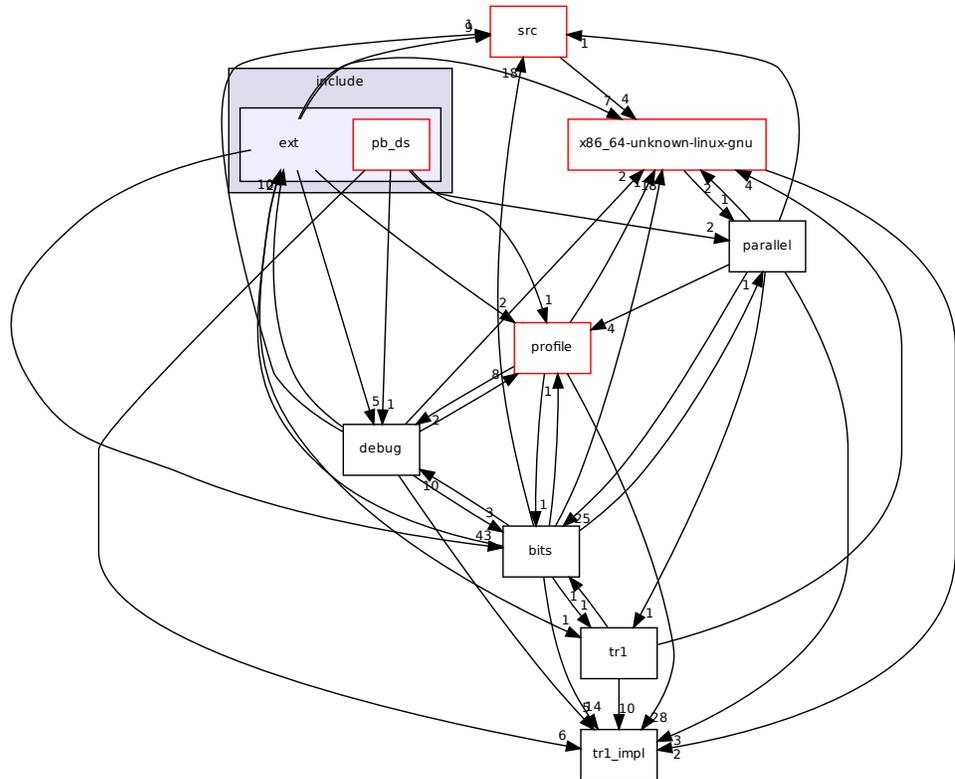


#### Files

- file `doxygroups.cc`

### 3.9 include/ext/ Directory Reference

Directory dependency graph for include/ext/:



#### Directories

- directory [pb\\_ds](#)

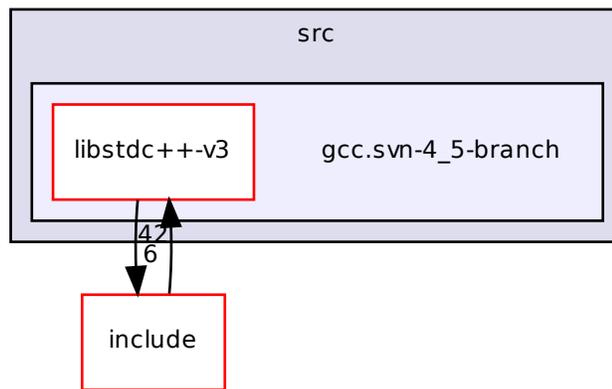
#### Files

- file [ext/algorithm](#)
- file [array\\_allocator.h](#)
- file [atomicity.h](#)
- file [bitmap\\_allocator.h](#)

- file **cast.h**
- file [codecvt\\_specializations.h](#)
- file [concurrency.h](#)
- file [debug\\_allocator.h](#)
- file [enc\\_filebuf.h](#)
- file [extptr\\_allocator.h](#)
- file [ext/functional](#)
- file [ext/iterator](#)
- file [malloc\\_allocator.h](#)
- file [ext/memory](#)
- file [mt\\_allocator.h](#)
- file [new\\_allocator.h](#)
- file [ext/numeric](#)
- file [numeric\\_traits.h](#)
- file [pod\\_char\\_traits.h](#)
- file [pointer.h](#)
- file [pool\\_allocator.h](#)
- file [rb\\_tree](#)
- file [rc\\_string\\_base.h](#)
- file [rope](#)
- file [ropeimpl.h](#)
- file [slist](#)
- file [sso\\_string\\_base.h](#)
- file [stdio\\_filebuf.h](#)
- file [stdio\\_sync\\_filebuf.h](#)
- file **string\_conversions.h**
- file [throw\\_allocator.h](#)
- file [type\\_traits.h](#)
- file [typelist.h](#)
- file [vstring.h](#)
- file [vstring.tcc](#)
- file [vstring\\_fwd.h](#)
- file [vstring\\_util.h](#)

### 3.10 /mnt/share/src/gcc.svn-4\_5-branch/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-4\_5-branch/:

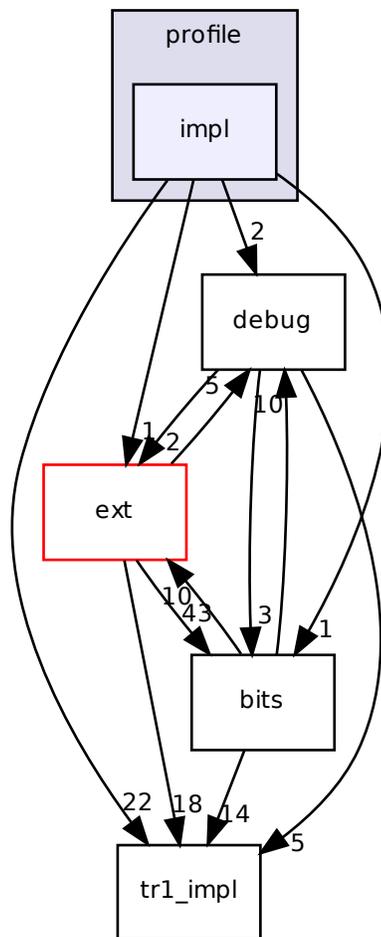


#### Directories

- directory [libstdc++-v3](#)

### 3.11 include/profile/impl/ Directory Reference

Directory dependency graph for include/profile/impl/:

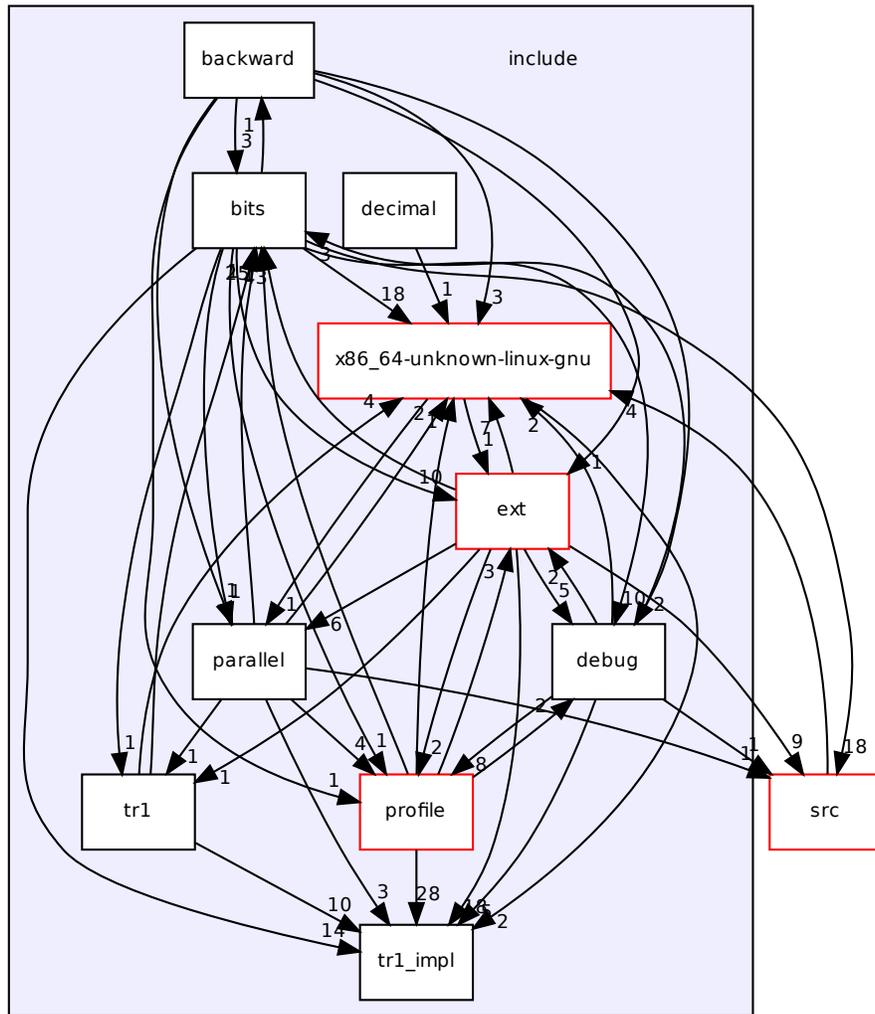


## Files

- file [profiler.h](#)
- file **profiler\_container\_size.h**
- file **profiler\_hash\_func.h**
- file [profiler\\_hashtable\\_size.h](#)
- file [profiler\\_list\\_to\\_slist.h](#)
- file [profiler\\_list\\_to\\_vector.h](#)
- file [profiler\\_map\\_to\\_unordered\\_map.h](#)
- file [profiler\\_node.h](#)
- file [profiler\\_state.h](#)
- file [profiler\\_trace.h](#)
- file [profiler\\_vector\\_size.h](#)
- file [profiler\\_vector\\_to\\_list.h](#)

### 3.12 include/ Directory Reference

Directory dependency graph for include/:



## Directories

- directory [backward](#)
- directory [bits](#)
- directory [debug](#)
- directory [decimal](#)
- directory [ext](#)
- directory [parallel](#)
- directory [profile](#)
- directory [tr1](#)
- directory [tr1\\_impl](#)
- directory [x86\\_64-unknown-linux-gnu](#)

## Files

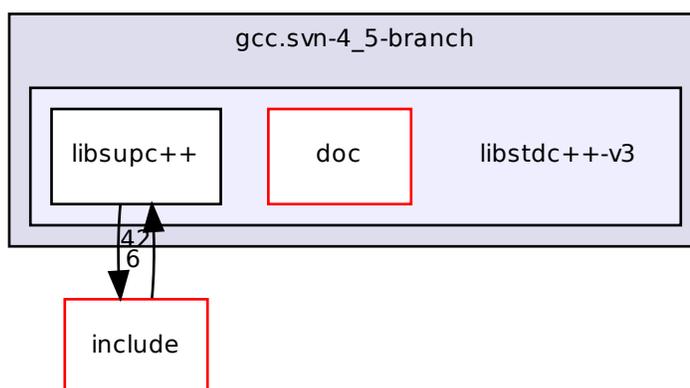
- file [algorithm](#)
- file [array](#)
- file [atomic](#)
- file [bitset](#)
- file [cassert](#)
- file [ccomplex](#)
- file [cctype](#)
- file [cerrno](#)
- file [cfenv](#)
- file [cfloat](#)
- file [chrono](#)
- file [cinttypes](#)
- file [ciso646](#)
- file [climits](#)
- file [clocale](#)
- file [cmath](#)
- file [complex](#)
- file [complex.h](#)
- file [condition\\_variable](#)
- file [csetjmp](#)
- file [csignal](#)
- file [cstdarg](#)
- file [cstdbool](#)
- file [cstddef](#)
- file [cstdint](#)
- file [cstdio](#)
- file [cstdlib](#)

- file [cstring](#)
- file [ctgmath](#)
- file [ctime](#)
- file [cwchar](#)
- file [cwctype](#)
- file [deque](#)
- file [fenv.h](#)
- file [fstream](#)
- file [functional](#)
- file [future](#)
- file **[gstdint.h](#)**
- file [iomanip](#)
- file [ios](#)
- file [iosfwd](#)
- file [iostream](#)
- file [istream](#)
- file [iterator](#)
- file [limits](#)
- file [list](#)
- file [locale](#)
- file [map](#)
- file [memory](#)
- file [mutex](#)
- file [numeric](#)
- file [ostream](#)
- file [queue](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [sstream](#)
- file [stack](#)
- file [stdatomic.h](#)
- file [stdexcept](#)
- file [streambuf](#)
- file [string](#)
- file [system\\_error](#)
- file [tgmath.h](#)
- file [thread](#)
- file [tuple](#)
- file [type\\_traits](#)
- file [unordered\\_map](#)
- file [unordered\\_set](#)

- file [utility](#)
- file [valarray](#)
- file [vector](#)

### 3.13 /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/:

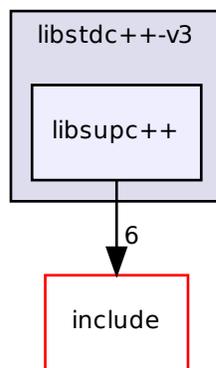


#### Directories

- directory [doc](#)
- directory [libsupc++](#)

### 3.14 /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/libsupc++/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/libsupc++/:

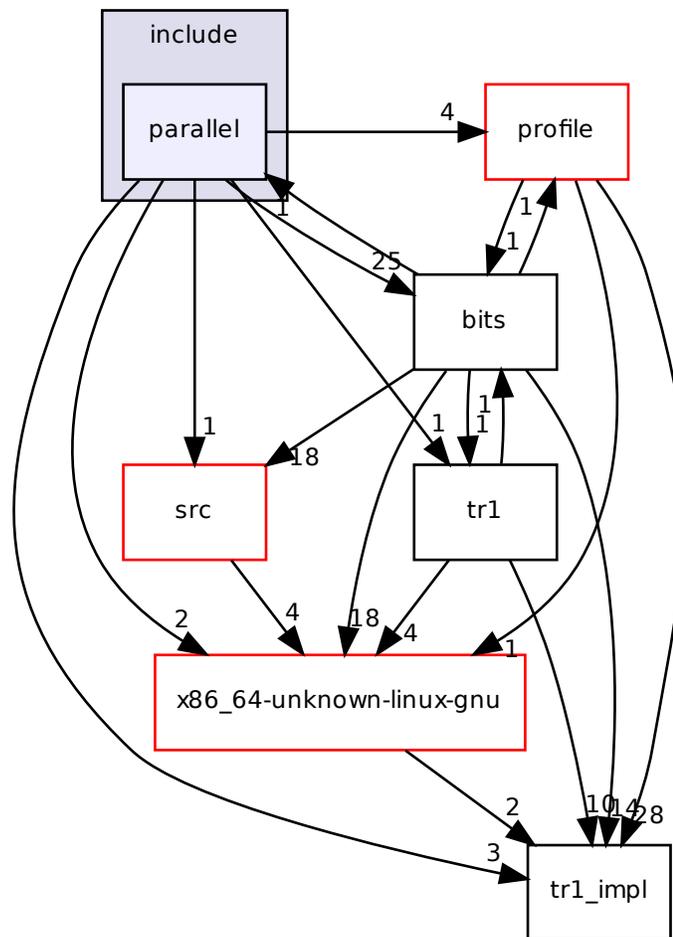


#### Files

- file [cxxabi-forced.h](#)
- file [cxxabi.h](#)
- file [exception](#)
- file [exception\\_ptr.h](#)
- file [initializer\\_list](#)
- file [nested\\_exception.h](#)
- file [new](#)
- file [typeinfo](#)

### 3.15 include/parallel/ Directory Reference

Directory dependency graph for include/parallel/:



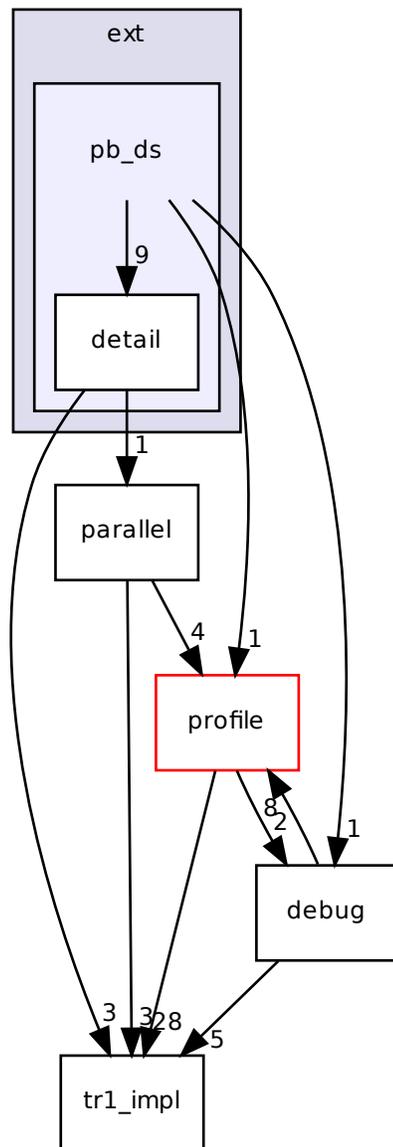
## Files

- file [algo.h](#)
- file [algbase.h](#)
- file [parallel/algorithm](#)
- file [parallel/algorithmfwd.h](#)
- file [balanced\\_quicksort.h](#)
- file [parallel/base.h](#)
- file [basic\\_iterator.h](#)
- file [checkers.h](#)
- file [parallel/compatibility.h](#)
- file [comptime\\_settings.h](#)
- file [equally\\_split.h](#)
- file [features.h](#)
- file [find.h](#)
- file [find\\_selectors.h](#)
- file [for\\_each.h](#)
- file [for\\_each\\_selectors.h](#)
- file [iterator.h](#)
- file [list\\_partition.h](#)
- file [losertree.h](#)
- file [merge.h](#)
- file [multiseq\\_selection.h](#)
- file [multiway\\_merge.h](#)
- file [multiway\\_mergesort.h](#)
- file [parallel/numeric](#)
- file [numericfwd.h](#)
- file [omp\\_loop.h](#)
- file [omp\\_loop\\_static.h](#)
- file [par\\_loop.h](#)
- file [parallel.h](#)
- file [partial\\_sum.h](#)
- file [partition.h](#)
- file [queue.h](#)
- file [quicksort.h](#)
- file [random\\_number.h](#)
- file [random\\_shuffle.h](#)
- file [search.h](#)
- file [set\\_operations.h](#)
- file [settings.h](#)
- file [sort.h](#)
- file [tags.h](#)
- file [types.h](#)
- file [unique\\_copy.h](#)
- file [workstealing.h](#)



### 3.16 include/ext/pb\_ds/ Directory Reference

Directory dependency graph for include/ext/pb\_ds/:



## Directories

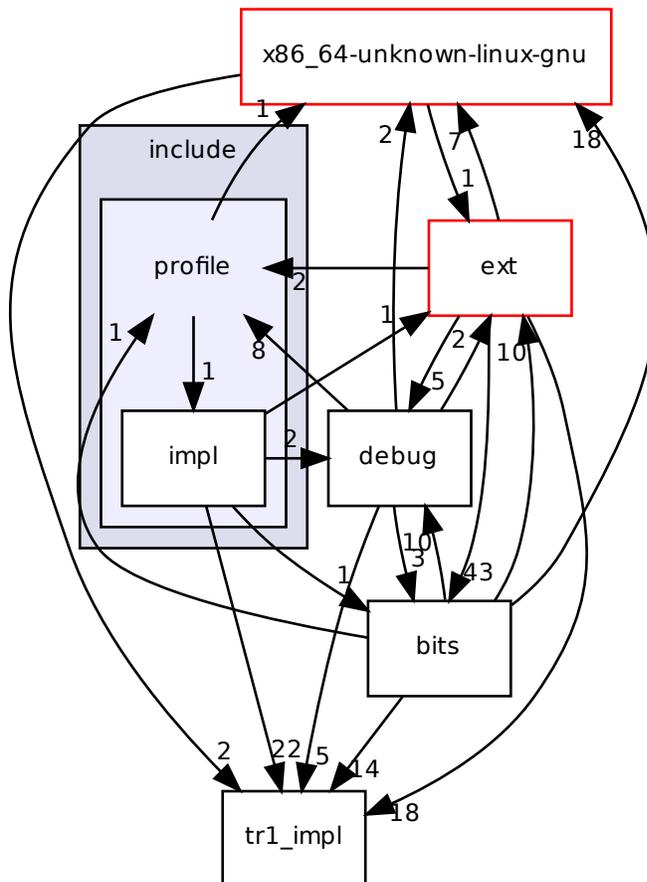
- directory [detail](#)

## Files

- file [assoc\\_container.hpp](#)
- file [exception.hpp](#)
- file [hash\\_policy.hpp](#)
- file [list\\_update\\_policy.hpp](#)
- file [priority\\_queue.hpp](#)
- file [tag\\_and\\_trait.hpp](#)
- file [tree\\_policy.hpp](#)
- file [trie\\_policy.hpp](#)

### 3.17 include/profile/ Directory Reference

Directory dependency graph for include/profile/:



#### Directories

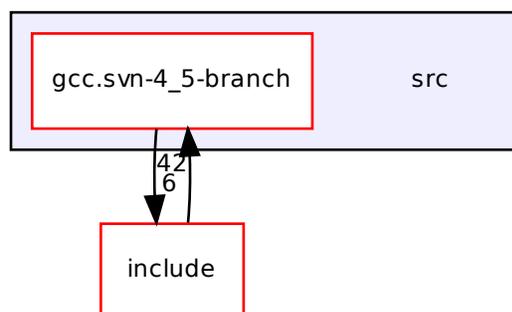
- directory [impl](#)

## Files

- file [profile/base.h](#)
- file [profile/bitset](#)
- file [profile/deque](#)
- file [iterator\\_tracker.h](#)
- file [profile/list](#)
- file [profile/map](#)
- file [profile/map.h](#)
- file [profile/multimap.h](#)
- file [profile/multiset.h](#)
- file [profile/set](#)
- file [profile/set.h](#)
- file [profile/unordered\\_map](#)
- file [profile/unordered\\_set](#)
- file [profile/vector](#)

## 3.18 /mnt/share/src/ Directory Reference

Directory dependency graph for /mnt/share/src/:

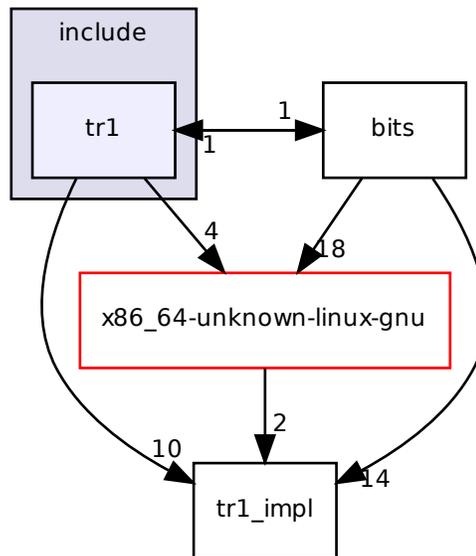


## Directories

- directory [gcc.svn-4\\_5-branch](#)

## 3.19 include/tr1/ Directory Reference

Directory dependency graph for include/tr1/:



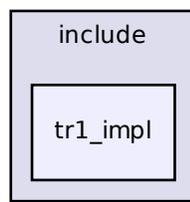
### Files

- file [tr1/complex](#)
- file [tr1/cctype](#)
- file [tr1/cfenv](#)
- file [tr1/cfloat](#)
- file [tr1/cinttypes](#)
- file [tr1/climits](#)
- file [tr1/cmath](#)
- file [tr1/complex](#)
- file [tr1/cstdarg](#)
- file [tr1/cstdbool](#)
- file [tr1/cstdint](#)

- file [tr1/cstdio](#)
- file [tr1/cstdlib](#)
- file [tr1/ctgmath](#)
- file [tr1/ctime](#)
- file [tr1/cwchar](#)
- file [tr1/cwctype](#)

### 3.20 include/tr1\_impl/ Directory Reference

Directory dependency graph for include/tr1\_impl/:

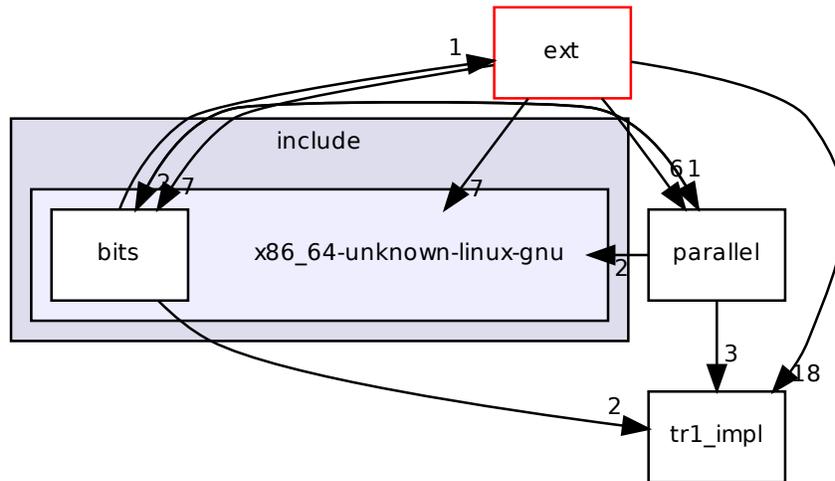


#### Files

- file [tr1\\_impl/array](#)
- file [boost\\_sp\\_counted\\_base.h](#)
- file [tr1\\_impl/cctype](#)
- file [tr1\\_impl/cfenv](#)
- file [tr1\\_impl/cinttypes](#)
- file [tr1\\_impl/cmath](#)
- file [tr1\\_impl/complex](#)
- file [tr1\\_impl/cstdlib](#)
- file [tr1\\_impl/cstdio](#)
- file [tr1\\_impl/cstdlib](#)
- file [tr1\\_impl/cwchar](#)
- file [tr1\\_impl/cwctype](#)
- file [tr1\\_impl/type\\_traits](#)
- file [tr1\\_impl/utility](#)

## 3.21 include/x86\_64-unknown-linux-gnu/ Directory Reference

Directory dependency graph for include/x86\_64-unknown-linux-gnu/:



### Directories

- directory [bits](#)



# Chapter 4

## Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

GNU extensions for public use.

#### Namespaces

- namespace `__detail`
- namespace `typelist`

#### Classes

- struct `__common_pool_policy`  
*Policy for shared `__pool` objects.*
- class `__mt_alloc`  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class `__mt_alloc_base`  
*Base class for `_Tp` dependent member functions.*
- struct `__per_type_pool_policy`  
*Policy for individual `__pool` objects.*

- class [\\_\\_pool< false >](#)  
*Specialization for single thread.*
- class [\\_\\_pool< true >](#)  
*Specialization for thread enabled, via `gthreads.h`.*
- class [\\_\\_pool\\_alloc](#)  
*Allocator using a memory pool with a single lock.*
- class [\\_\\_pool\\_alloc\\_base](#)  
*Base class for `__pool_alloc`.*
- struct [\\_\\_pool\\_base](#)  
*Base class for pool object.*
- class [\\_\\_rc\\_string\\_base](#)
- class [\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*
- class [\\_\\_versa\\_string](#)  
*Template class `__versa_string`.  
Data structure managing sequences of characters and character-like objects.*
- struct [\\_Caster](#)
- struct [\\_Char\\_types](#)  
*Mapping from character type to associated types.*
- class [\\_ExtPtr\\_allocator](#)  
*An example allocator which uses a non-standard pointer type.  
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.*
- struct [\\_Invalid\\_type](#)
- class [\\_Pointer\\_adapter](#)
- class [\\_Relative\\_pointer\\_impl](#)  
*A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.*
- class [\\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_Std\\_pointer\\_impl](#)  
*A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.*

- struct `_Unqualified_type`
- struct `annotate_base`

*Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- class `array_allocator`

*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class `array_allocator_base`

*Base class.*
- class `binary_compose`

*An SGI extension .*
- class `bitmap_allocator`

*Bitmap Allocator, primary template.*
- struct `char_traits`

*Base class used to implement `std::char_traits`.*
- struct `character`

*A POD class that serves as a character abstraction class.*
- struct `condition_base`

*Base struct for condition policy.*
- struct `constant_binary_fun`

*An SGI extension .*
- struct `constant_unary_fun`

*An SGI extension .*
- struct `constant_void_fun`

*An SGI extension .*
- class `debug_allocator`

*A meta-allocator with debugging bits, as per [20.4]. This is precisely the allocator defined in the C++ Standard.*

  - all allocation calls operator `new`
  - all deallocation calls operator `delete`.

- class [enc\\_filebuf](#)  
*class [enc\\_filebuf](#).*
- struct [encoding\\_char\\_traits](#)  
*[encoding\\_char\\_traits](#)*
- class [encoding\\_state](#)  
*Extension to use [icov](#) for dealing with character encodings.*
- struct [forced\\_error](#)  
*Thrown by exception safety machinery.*
- class [free\\_list](#)  
*The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).*
- class [hash\\_map](#)
- class [hash\\_multimap](#)
- class [hash\\_multiset](#)
- class [hash\\_set](#)
- struct [limit\\_condition](#)  
*Base class for incremental control and throw.*
- class [malloc\\_allocator](#)  
*An allocator that uses [malloc](#).  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls [malloc](#)
  - all deallocation calls [free](#).
- class [new\\_allocator](#)  
*An allocator that uses global [new](#), as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator [new](#)
  - all deallocation calls operator [delete](#).
- struct [project1st](#)  
*An [SGI extension](#) .*
- struct [project2nd](#)  
*An [SGI extension](#) .*
- struct [random\\_condition](#)

*Base class for random probability control and throw.*

- struct `rb_tree`
- class `recursive_init_error`

*Exception thrown by `__cxa_guard_acquire`.  
6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `rope`
- struct `select1st`

*An SGI extension .*
- struct `select2nd`

*An SGI extension .*
- class `slist`
- class `stdio_filebuf`

*Provides a layer of compatibility for C/POSIX.  
This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `stdio_sync_filebuf`

*Provides a layer of compatibility for C.  
This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`

*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.  
Note: Deallocate not allowed to throw.*
- struct `throw_allocator_limit`

*Allocator throwing via limit condition.*
- struct `throw_allocator_random`

*Allocator throwing via random condition.*
- struct `throw_value_base`

*Class with exception generation control. Intended to be used as a `value_type` in templated code.*

- struct [throw\\_value\\_limit](#)  
*Type throwing via limit condition.*
- struct [throw\\_value\\_random](#)  
*Type throwing via random condition.*
- class [unary\\_compose](#)  
*An SGI extension .*

## Typedefs

- typedef void(\* [\\_\\_destroy\\_handler](#))(void \*)
- typedef [\\_\\_versa\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_rc\\_string](#)
- typedef [\\_\\_vstring](#) [\\_\\_sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u16rc\\_string](#)
- typedef [\\_\\_u16vstring](#) [\\_\\_u16sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char16\_t > [\\_\\_u16vstring](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_u32rc\\_string](#)
- typedef [\\_\\_u32vstring](#) [\\_\\_u32sso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< char32\_t > [\\_\\_u32vstring](#)
- typedef [\\_\\_versa\\_string](#)< char > [\\_\\_vstring](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, [\\_\\_rc\\_string\\_base](#) > [\\_\\_wrc\\_string](#)
- typedef [\\_\\_wvstring](#) [\\_\\_wsso\\_string](#)
- typedef [\\_\\_versa\\_string](#)< wchar\_t > [\\_\\_wvstring](#)
- typedef [rope](#)< char > [crope](#)
- typedef [rope](#)< wchar\_t > [wrope](#)

## Enumerations

- enum { [\\_S\\_num\\_primes](#) }
- enum [\\_Lock\\_policy](#) { [\\_S\\_single](#), [\\_S\\_mutex](#), [\\_S\\_atomic](#) }

## Functions

- static void `__atomic_add` (volatile `_Atomic_word` \*`__mem`, int `__val`)
- static void `__atomic_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- static `_Atomic_word` `__attribute__((__unused__))` `__exchange_and_add_dispatch`(`_Atomic_word` \*`__mem`)
- template<class `_Tp` >  
void `__aux_require_boolean_expr` (const `_Tp` &`__t`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__const_pointer_cast` (const `_FromType` &`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__const_pointer_cast` (`_FromType` \*`__arg`)
- template<typename `_InputIterator`, typename `_Size`, typename `_OutputIterator` >  
`pair`< `_InputIterator`, `_OutputIterator` > `__copy_n` (`_InputIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `input_iterator_tag`)
- template<typename `_RAIterator`, typename `_Size`, typename `_OutputIterator` >  
`pair`< `_RAIterator`, `_OutputIterator` > `__copy_n` (`_RAIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `random_access_iterator_tag`)
- template<typename `_InputIterator`, typename `_Distance` >  
void `__distance` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_Distance` &`__n`, `std::input_iterator_tag`)
- template<typename `_RandomAccessIterator`, typename `_Distance` >  
void `__distance` (`_RandomAccessIterator` `__first`, `_RandomAccessIterator` `__last`, `_Distance` &`__n`, `std::random_access_iterator_tag`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (const `_FromType` &`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (`_FromType` \*`__arg`)
- void `__error_type_must_be_a_signed_integer_type` ()
- void `__error_type_must_be_an_integer_type` ()
- void `__error_type_must_be_an_unsigned_integer_type` ()
- static `_Atomic_word` `__exchange_and_add` (volatile `_Atomic_word` \*`__mem`, int `__val`)
- static `_Atomic_word` `__exchange_and_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- else return `__exchange_and_add_single` (`__mem`, `__val`)
- template<class `_Concept` >  
void `__function_requires` ()
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type` \*`__ptr`)
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type`)
- int `__lexicographical_compare_3way` (const unsigned char \*`__first1`, const unsigned char \*`__last1`, const unsigned char \*`__first2`, const unsigned char \*`__last2`)

- `int __lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `template<typename _InputIterator1, typename _InputIterator2 >  
int __lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp >  
const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >  
const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference __mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >  
_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >  
_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType >  
_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType __reinterpret_pointer_cast (_FromType *__arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base *__head, const _Slist_node_base *__node)`
- `const _Slist_node_base * __slist_previous (const _Slist_node_base *__head, const _Slist_node_base *__node)`
- `_Slist_node_base * __slist_reverse (_Slist_node_base *__node)`
- `size_t __slist_size (_Slist_node_base *__node)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__head)`
- `template<typename _ToType, typename _FromType >  
_ToType __static_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType __static_pointer_cast (_FromType *__arg)`

- `size_t __stl_hash_string` (const char \* \_\_s)
- `unsigned long __stl_next_prime` (unsigned long \_\_n)
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base> _Ret __stoa` (\_TRet(\*\_\_convf)(const \_CharT \*, \_CharT \*\*, \_Base...), const char \* \_\_name, const \_CharT \* \_\_str, std::size\_t \* \_\_idx, \_Base... \_\_base)
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_forced_error` ()
- `template<typename _String, typename _CharT = typename _String::value_type> _String __to_xstring` (int(\*\_\_convf)(\_CharT \*, std::size\_t, const \_CharT \*, \_\_builtin\_va\_list), std::size\_t \_\_n, const \_CharT \* \_\_fmt,...)
- `template<typename _InputIter, typename _Size, typename _ForwardIter > pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::input\\_iterator\\_tag](#))
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter > pair< _RandomAccessIter, _ForwardIter > __uninitialized_copy_n` (\_RandomAccessIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::random\\_access\\_iterator\\_tag](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter > pair< _InputIter, _ForwardIter > __uninitialized_copy_n` (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator > pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, \_Allocator \_\_alloc)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp > pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a` (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::allocator<\\_Tp >](#))
- `void __verbose_terminate_handler` ()
- `size_t __Bit_scan_forward` (size\_t \_\_num)
- `template<typename _ForwardIterator, typename _Allocator > void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Allocator \_\_alloc)
- `template<typename _ForwardIterator, typename _Tp > void __Destroy_const` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [allocator<\\_Tp >](#))
- `template<class _CharT, class _Traits > void __Rope_fill` ([basic\\_ostream<\\_CharT, \\_Traits >](#) & \_\_o, size\_t \_\_n)
- `template<class _CharT > bool __Rope_is_simple` (\_CharT \*)
- `bool __Rope_is_simple` (char \*)
- `bool __Rope_is_simple` (wchar\_t \*)
- `template<class _Rope_iterator > void __Rope_rotate` (\_Rope\_iterator \_\_first, \_Rope\_iterator \_\_middle, \_Rope\_iterator \_\_last)

- `template<class _CharT >`  
`void _S_cond_store_eos (_CharT &)`
- `void _S_cond_store_eos (char &__c)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`  
`_CharT _S_eos (_CharT *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`  
`bool _S_is_basic_char_type (_CharT *)`
- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<class _CharT >`  
`bool _S_is_one_byte_char_type (_CharT *)`
- `template<class _Operation1 , class _Operation2 >`  
`unary\_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`  
`binary\_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`  
`constant\_void\_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant\_unary\_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant\_binary\_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator , typename _Size , typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > copy\_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator , typename _Tp , typename _Size >`  
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator , typename _Predicate , typename _Size >`  
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator , typename _Distance >`  
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`  
`_Tp identity\_element (std::plus< _Tp >)`
- `template<class _Tp >`  
`_Tp identity\_element (std::multiplies< _Tp >)`
- `static _Atomic_word int __val if (__gthread_active_p()) return __exchange_and_add(__mem`

- `template<typename _ForwardIter, typename _Tp >`  
`void iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`  
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _-`  
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`  
`bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int lexicographical\_compare\_3way (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun1 (_Ret(_Tp::*_f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun1\_ref (_Ret(_Tp::*_f)(_Arg))`
  
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &_-`  
`__hs1, const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs,`  
`const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs,`  
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp, typename _Array >`  
`bool operator!= (const array\_allocator< _Tp, _Array > &, const array\_-`  
`allocator< _Tp, _Array > &)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator<`  
`_Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename,`  
`typename > class _Base>`  
`bool operator!= (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits,`  
`_Alloc, _Base > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const malloc\_allocator< _Tp > &, const malloc\_allocator<`  
`_Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const \_Rope\_const\_iterator< _CharT, _Alloc > &__x, const`  
`\_Rope\_const\_iterator< _CharT, _Alloc > &__y)`

- `template<typename _Tp, typename _Poolp >`  
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`  
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp >`  
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Tp, typename _Cond >`  
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Tp, class _Alloc >`  
`bool operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`  
`lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`  
`lhs, const _CharT *__rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc >`  
`&__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1,`  
`const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Cond >`  
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond >`  
`&__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_`  
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_`  
`iterator< _Iterator, _Container >::difference_type __n, const __normal_`  
`iterator< _Iterator, _Container > &__i)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _`  
`Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT,`  
`_Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_`  
`iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left,`  
`const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left,`  
`const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left,`  
`_CharT __right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- `__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _Cond > throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<class _CharT, class _Alloc > rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc > rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT * __right)`
- `template<class _CharT, class _Alloc > rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc > _Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container > auto operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)-> decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container > __normal_iterator< _Iterator, _Container >::difference_type operator- (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _CharT, class _Alloc > ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc >` **operator-** (const `_Rope_iterator< _CharT, _Alloc >` &\_\_x, `ptrdiff_t` \_\_n)
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t` **operator-** (const `_Rope_iterator< _CharT, _Alloc >` &\_\_x, const `_Rope_iterator< _CharT, _Alloc >` &\_\_y)
- `template<typename _Cond >`  
`throw_value_base< _Cond >` **operator-** (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)
- `template<class _Tp, class _Alloc >`  
`bool` **operator<** (const `slist< _Tp, _Alloc >` &\_SL1, const `slist< _Tp, _Alloc >` &\_SL2)
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool` **operator<** (const `__normal_iterator< _IteratorL, _Container >` &\_\_lhs, const `__normal_iterator< _IteratorR, _Container >` &\_\_rhs)
- `template<typename _Iterator, typename _Container >`  
`bool` **operator<** (const `__normal_iterator< _Iterator, _Container >` &\_\_lhs, const `__normal_iterator< _Iterator, _Container >` &\_\_rhs)
- `template<typename _Tp1, typename _Tp2 >`  
`bool` **operator<** (const `Pointer_adapter< _Tp1 >` &\_\_lhs, `_Tp2` \_\_rhs)
- `template<class _CharT, class _Alloc >`  
`bool` **operator<** (const `_Rope_const_iterator< _CharT, _Alloc >` &\_\_x, const `_Rope_const_iterator< _CharT, _Alloc >` &\_\_y)
- `template<class _CharT, class _Alloc >`  
`bool` **operator<** (const `_Rope_iterator< _CharT, _Alloc >` &\_\_x, const `_Rope_iterator< _CharT, _Alloc >` &\_\_y)
- `template<typename V, typename I, typename S >`  
`bool` **operator<** (const `character< V, I, S >` &lhs, const `character< V, I, S >` &rhs)
- `template<typename _Tp1, typename _Tp2 >`  
`bool` **operator<** (`_Tp1` \_\_lhs, const `Pointer_adapter< _Tp2 >` &\_\_rhs)
- `template<typename _Tp1, typename _Tp2 >`  
`bool` **operator<** (const `Pointer_adapter< _Tp1 >` &\_\_lhs, const `Pointer_adapter< _Tp2 >` &\_\_rhs)
- `template<class _CharT, class _Alloc >`  
`bool` **operator<** (const `rope< _CharT, _Alloc >` &\_\_left, const `rope< _CharT, _Alloc >` &\_\_right)
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool` **operator<** (const `_CharT *`\_\_lhs, const `__versa_string< _CharT, _Traits, _Alloc, _Base >` &\_\_rhs)
- `template<typename _Cond >`  
`bool` **operator<** (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`  
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_lhs, const _CharT * _rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator== (const __normal_iterator< _IteratorL, _Container > &_lhs, const __normal_iterator< _IteratorR, _Container > &_rhs)`
- `template<typename _Tp >`  
`bool operator== (const _Pointer_adapter< _Tp > &_lhs, int _rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator== (const __normal_iterator< _Iterator, _Container > &_lhs, const __normal_iterator< _Iterator, _Container > &_rhs)`
- `template<typename _Tp >`  
`bool operator== (int _lhs, const _Pointer_adapter< _Tp > &_rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &_hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &_hm2)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &_x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &_y)`
- `template<typename _Tp, typename _Cond >`  
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Value, class _HashFn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash_set< _Value, _HashFn, _EqualKey, _Alloc > &_hs1, const hash_set< _Value, _HashFn, _EqualKey, _Alloc > &_hs2)`
- `template<typename _Tp >`  
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const rope< _CharT, _Alloc > &_left, const rope< _CharT, _Alloc > &_right)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &_ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &_ht2)`

- `template<class _Tp, class _Alloc >`  
`bool operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`  
`__enable_if< std::is_char< _CharT >::__value, bool >::__type operator== (const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &_lhs, const \_\_versa\_string< _CharT, std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &_rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const \_Rope\_iterator< _CharT, _Alloc > &_x, const \_Rope\_iterator< _CharT, _Alloc > &_y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &_lhs, const \_Pointer\_adapter< _Tp2 > &_rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator== (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &_hm1, const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &_hm2)`
- `template<typename _Tp >`  
`bool operator== (const malloc\_allocator< _Tp > &, const malloc\_allocator< _Tp > &)`
- `template<typename _Tp, typename _Poolp >`  
`bool operator== (const \_\_mt\_alloc< _Tp, _Poolp > &, const \_\_mt\_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`  
`bool operator== (const new\_allocator< _Tp > &, const new\_allocator< _Tp > &)`
- `template<typename V, typename I, typename S >`  
`bool operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (_Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > &_rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &_lhs, _Tp2 __rhs)`
- `template<typename _Tp >`  
`bool operator== (const \_Pointer\_adapter< _Tp > &_lhs, const \_Pointer\_adapter< _Tp > &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- ```
bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_
_lhs, const _CharT *__rhs)
```
- `template<typename _Cond >`  

```
bool operator== (const throw_value_base< _Cond > &__a, const throw_
value_base< _Cond > &__b)
```
  - `template<typename _Tp, typename _Array >`  

```
bool operator== (const array_allocator< _Tp, _Array > &, const array_
allocator< _Tp, _Array > &)
```
  - `template<class _CharT, class _Alloc >`  

```
bool operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const
_Rope_const_iterator< _CharT, _Alloc > &__y)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>`  

```
bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_
_lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>`  

```
bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > &__rhs)
```
  - `template<typename _IteratorL, typename _IteratorR, typename _Container >`  

```
bool operator> (const __normal_iterator< _IteratorL, _Container > &__lhs,
const __normal_iterator< _IteratorR, _Container > &__rhs)
```
  - `template<typename _Tp >`  

```
bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_
adapter< _Tp > &__rhs)
```
  - `template<typename _Iterator, typename _Container >`  

```
bool operator> (const __normal_iterator< _Iterator, _Container > &__lhs,
const __normal_iterator< _Iterator, _Container > &__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>`  

```
bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__
lhs, const _CharT *__rhs)
```
  - `template<class _CharT, class _Alloc >`  

```
bool operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const
_Rope_const_iterator< _CharT, _Alloc > &__y)
```
  - `template<typename _Tp1, typename _Tp2 >`  

```
bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
```
  - `template<class _CharT, class _Alloc >`  

```
bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _
Rope_iterator< _CharT, _Alloc > &__y)
```
  - `template<typename _Tp1, typename _Tp2 >`  

```
bool operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
```
  - `template<class _CharT, class _Alloc >`  

```
bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT,
_Alloc > &__y)
```

- `template<typename _Tp1 , typename _Tp2 >`  
`bool operator> (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _Tp , class _Alloc >`  
`bool operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const \_Rope\_const\_iterator< _CharT, _Alloc > &__x, const \_Rope\_const\_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool operator>= (const \_normal\_iterator< _Iterator, _Container > &__lhs, const \_normal\_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp >`  
`bool operator>= (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const \_Rope\_iterator< _CharT, _Alloc > &__x, const \_Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT *__lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool operator>= (const \_Pointer\_adapter< _Tp1 > &__lhs, \_Tp2 __rhs)`
- `template<typename _IteratorL , typename _IteratorR, typename _Container >`  
`bool operator>= (const \_normal\_iterator< _IteratorL, _Container > &__lhs, const \_normal\_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool `operator>=` (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &`_lhs`, const `_CharT` \*`_rhs`)
- template<typename `_Tp1`, typename `_Tp2` >  
bool `operator>=` (`_Tp1` `_lhs`, const `__Pointer_adapter`< `_Tp2` > &`_rhs`)
  - template<class `_Tp`, class `_Alloc` >  
bool `operator>=` (const `slist`< `_Tp`, `_Alloc` > &`_SL1`, const `slist`< `_Tp`, `_Alloc` > &`_SL2`)
  - template<typename `_Tp1`, typename `_Tp2` >  
bool `operator>=` (const `__Pointer_adapter`< `_Tp1` > &`_lhs`, const `__Pointer_adapter`< `_Tp2` > &`_rhs`)
  - template<typename `_Tp`, typename `_Integer`, typename `_MonoidOperation` >  
`_Tp` `power` (`_Tp` `_x`, `_Integer` `_n`, `_MonoidOperation` `__monoid_op`)
  - template<typename `_Tp`, typename `_Integer` >  
`_Tp` `power` (`_Tp` `_x`, `_Integer` `_n`)
  - template<typename `_InputIterator`, typename `_RandomAccessIterator`, typename `_RandomNumberGenerator` >  
`_RandomAccessIterator` `random_sample` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_RandomAccessIterator` `__out_first`, `_RandomAccessIterator` `__out_last`, `_RandomNumberGenerator` &`_rand`)
  - template<typename `_InputIterator`, typename `_RandomAccessIterator` >  
`_RandomAccessIterator` `random_sample` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_RandomAccessIterator` `__out_first`, `_RandomAccessIterator` `__out_last`)
  - template<typename `_ForwardIterator`, typename `_OutputIterator`, typename `_Distance`, typename `_RandomNumberGenerator` >  
`_OutputIterator` `random_sample_n` (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, `_OutputIterator` `__out`, const `_Distance` `_n`, `_RandomNumberGenerator` &`_rand`)
  - template<typename `_ForwardIterator`, typename `_OutputIterator`, typename `_Distance` >  
`_OutputIterator` `random_sample_n` (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, `_OutputIterator` `__out`, const `_Distance` `_n`)
  - void `rotate` (`_Rope_iterator`< `char`, `__STL_DEFAULT_ALLOCATOR`(`char`)> `__first`, `_Rope_iterator`< `char`, `__STL_DEFAULT_ALLOCATOR`(`char`)> `__middle`, `_Rope_iterator`< `char`, `__STL_DEFAULT_ALLOCATOR`(`char`)> `__last`)
  - double `stod` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0)
  - float `stof` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0)
  - int `stoi` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0, int `__base`=10)
  - long `stol` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0, int `__base`=10)
  - long double `stold` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0)
  - long long `stoll` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0, int `__base`=10)
  - unsigned long `stoul` (const `__vstring` &`_str`, `std::size_t` \*`_idx`=0, int `__base`=10)
  - unsigned long long `stoull` (const `__vstring` &`_str`, `std::size_t` \*`_idx`, int `__base`=10)

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _CharT, class _Alloc >`  
`void swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<typename _Cond >`  
`void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`  
`void swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`  
`void swap (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<class _CharT, class _Alloc >`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `__vstring to_string` (long double \_\_val)
- `__vstring to_string` (double \_\_val)
- `__vstring to_string` (float \_\_val)
- `__vstring to_string` (unsigned \_\_val)
- `__vstring to_string` (unsigned long \_\_val)
- `__vstring to_string` (unsigned long long \_\_val)
- `__vstring to_string` (int \_\_val)
- `__vstring to_string` (long long \_\_val)
- `__vstring to_string` (long \_\_val)
- `__wvstring to_wstring` (double \_\_val)
- `__wvstring to_wstring` (long long \_\_val)
- `__wvstring to_wstring` (long double \_\_val)

- [\\_\\_wvstring to\\_wstring](#) (long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (float \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned long long \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (unsigned \_\_val)
- [\\_\\_wvstring to\\_wstring](#) (int \_\_val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
[pair<\\_InputIter, \\_ForwardIter > uninitialized\\_copy\\_n](#) (`_InputIter __first, _Size __count, _ForwardIter __result`)

## Variables

- static const `_Lock_policy` `__default_lock_policy`
- static const unsigned long `__stl_prime_list` [`_S_num_primes`]
- static `_Atomic_word` int `__val` `__val`
- `rope<_CharT, _Alloc >` `identity_element` (`_Rope_Concat_fn<_CharT, _Alloc >`)

### 4.1.1 Detailed Description

GNU extensions for public use.

### 4.1.2 Function Documentation

**4.1.2.1** `template<typename _ToType, typename _FromType > _ToType`  
`__gnu_cxx::__static_pointer_cast ( const _FromType & __arg )`  
**[inline]**

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 61 of file `cast.h`.

References `__static_pointer_cast()`.

Referenced by `__static_pointer_cast()`.

**4.1.2.2** `template<typename _ToType, typename _FromType > _ToType`  
`__gnu_cxx::__static_pointer_cast ( _FromType * __arg )` **[inline]**

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 89 of file cast.h.

References `__static_pointer_cast()`.

#### 4.1.2.3 `size_t __gnu_cxx::Bit_scan_forward ( size_t __num ) [inline]`

Generic Version of the bsf instruction.

Definition at line 513 of file bitmap\_allocator.h.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

#### 4.1.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test difference of C string and string.

##### Parameters

`__lhs` C string.

`__rhs` String.

##### Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2175 of file vstring.h.

#### 4.1.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test difference of two strings.

##### Parameters

`__lhs` First string.

`__rhs` Second string.

##### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2162 of file vstring.h.

**4.1.2.6** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test difference of string and C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2188 of file `vstring.h`.

**4.1.2.7** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, _CharT __rhs )`

Concatenate string and character.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 239 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.8** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate two strings.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 179 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.9** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate C string and string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 192 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.10** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( _CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )`

Concatenate character and string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 209 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.11** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs )`

Concatenate string and C string.

#### Parameters

`__lhs` First string.

`__rhs` Last string.

#### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 222 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.1.2.12** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator<( const _CharT * __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string precedes string.

#### Parameters

*\_\_lhs* C string.

*\_\_rhs* String.

#### Returns

True if *\_\_lhs* precedes *\_\_rhs*. False otherwise.

Definition at line 2228 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.13** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator<( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

#### Parameters

*\_\_lhs* String.

*\_\_rhs* C string.

#### Returns

True if *\_\_lhs* precedes *\_\_rhs*. False otherwise.

Definition at line 2215 of file `vstring.h`.

**4.1.2.14** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator<( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test if string precedes string.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2202 of file `vstring.h`.

```
4.1.2.15 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator<= ( const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
[inline]
```

Test if C string doesn't follow string.

**Parameters**

`__lhs` C string.  
`__rhs` String.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2308 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.16 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't follow string.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if *\_\_lhs* doesn't follow *\_\_rhs*. False otherwise.

Definition at line 2282 of file `vstring.h`.

```
4.1.2.17 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]
```

Test if string doesn't follow C string.

**Parameters**

*\_\_lhs* String.

*\_\_rhs* C string.

**Returns**

True if *\_\_lhs* doesn't follow *\_\_rhs*. False otherwise.

Definition at line 2295 of file `vstring.h`.

```
4.1.2.18 template<typename _Tp > bool __gnu_cxx::operator== ( const
_Pointer_adapter< _Tp > & __lhs, const _Pointer_adapter< _Tp >
& __rhs ) [inline]
```

Comparison operators for [\\_Pointer\\_adapter](#) defer to the base class'es comparison operators, when possible.

Definition at line 529 of file `pointer.h`.

```
4.1.2.19 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator== ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]
```

Test equivalence of string and C string.

**Parameters**

*\_\_lhs* String.

*\_\_rhs* C string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2148 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.20 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator==( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of two strings.

**Parameters**

`__lhs` First string.

`__rhs` Second string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2111 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.21 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator==( const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
[inline]
```

Test equivalence of C string and string.

**Parameters**

`__lhs` C string.

`__rhs` String.

**Returns**

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2135 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.22** `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string follows C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2255 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.23** `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string follows string.

#### Parameters

`__lhs` C string.

`__rhs` String.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2268 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.24** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string follows string.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

#### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2242 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.25** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename > class _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't precede string.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

#### Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2322 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.26** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string doesn't precede string.

#### Parameters

*\_\_lhs* C string.

*\_\_rhs* String.

#### Returns

True if *\_\_lhs* doesn't precede *\_\_rhs*. False otherwise.

Definition at line 2348 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.27** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't precede C string.

#### Parameters

*\_\_lhs* String.

*\_\_rhs* C string.

#### Returns

True if *\_\_lhs* doesn't precede *\_\_rhs*. False otherwise.

Definition at line 2335 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.28 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename > class _Base> void
__gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base >
& __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs
) [inline]
```

Swap contents of two strings.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2362 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap()`.

## 4.2 `__gnu_cxx::__detail` Namespace Reference

Implementation details not part of the namespace `__gnu_cxx` interface.

### Classes

- class `__mini_vector`  
*\_\_mini\_vector<> is a stripped down version of the full-fledged `std::vector<>`.*
- class `_Bitmap_counter`  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class `_Ffit_finder`  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*

### Enumerations

- enum { `_S_max_ropo_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

## Functions

- void `__bit_allocate` (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- void `__bit_free` (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
   `__lower_bound` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
   size\_t `__num_bitmaps` (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
   size\_t `__num_blocks` (\_AddrPair \_\_ap)

### 4.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

### 4.2.2 Function Documentation

**4.2.2.1 void `__gnu_cxx::__detail::__bit_allocate` ( size\_t \* `__pbmap`, size\_t `__pos` ) throw () [inline]**

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 492 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

**4.2.2.2 void `__gnu_cxx::__detail::__bit_free` ( size\_t \* `__pbmap`, size\_t `__pos` ) throw () [inline]**

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 503 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

**4.2.2.3 template<typename \_AddrPair > size\_t `__gnu_cxx::__detail::__num_bitmaps` ( \_AddrPair `__ap` ) [inline]**

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 280 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`, and `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

**4.2.2.4** `template<typename AddrPair > size_t __gnu_cxx::detail::__num_blocks ( AddrPair ap )`  
**[inline]**

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 272 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

## 4.3 `__gnu_cxx::typelist` Namespace Reference

GNU `typelist` extensions for public compile-time use.

### Functions

- `template<typename Fn , typename Typelist > void apply (Fn &, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV > void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist > void apply_generator (Fn &fn, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV > void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn , typename Typelist > void apply_generator (Gn &, Typelist)`

### 4.3.1 Detailed Description

GNU `typelist` extensions for public compile-time use.

### 4.3.2 Function Documentation

**4.3.2.1** `template<typename Gn , typename Typelist > void __gnu_cxx::typelist::apply_generator ( Gn &, Typelist )`

Apply all `typelist` types to generator functor.

## 4.4 `__gnu_debug` Namespace Reference

GNU debug classes for public use.

### Classes

- struct `__is_same`
- class `__After_nth_from`
- class `__Not_equal_to`
- class `__Safe_iterator`  
*Safe iterator wrapper.*
- class `__Safe_iterator_base`  
*Basic functionality for a safe iterator.*
- class `__Safe_sequence`  
*Base class for constructing a safe sequence type that tracks iterators that reference it.*
- class `__Safe_sequence_base`  
*Base class that supports tracking of iterators that reference a sequence.*
- class `basic_string`  
*Class `std::basic_string` with safety/checking/debug instrumentation.*

### Typedefs

- typedef `basic_string< char >` `string`
- typedef `basic_string< wchar_t >` `wstring`

### Enumerations

- enum `__Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`,  
`__msg_bad_bitset_flip`,

```

__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_
other,
__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __
msg_init_const_singular,
__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,
__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __
msg_iter_compare_bad,
__msg_compare_different, __msg_iter_order_bad, __msg_order_different,
__msg_distance_bad,
__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __
msg_output_ostream,
__msg_deref_istreambuf, __msg_inc_istreambuf }

```

## Functions

- `template<typename _Iterator >`  
`bool __check_dereferenceable (_Iterator &)`
- `template<typename _Tp >`  
`bool __check_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator _`  
`__last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator _`  
`__last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__value, _Pred __pred)`
- `template<typename _Iterator >`  
`bool __check_singular (_Iterator &__x)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Tp >`  
`bool __check_singular (const _Tp *__ptr)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base *__x)`

- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator , typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator , typename _Predicate >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator , typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1 , typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::\_\_true\_type)`
- `template<typename _InputIterator , typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::\_\_false\_type)`
- `template<typename _InputIterator , typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::\_\_false\_type)`
- `template<typename _CharT >`  
`const _CharT * __check_string (const _CharT *__s)`
- `template<typename _CharT , typename _Integer >`  
`const _CharT * __check_string (const _CharT *__s, const _Integer &__n \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _InputIterator >`  
`_InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`

- `template<typename _InputIterator >`  
`bool \_\_valid\_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence > &__first, const \_Safe\_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Integral >`  
`bool \_\_valid\_range\_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux (const _InputIterator &__first, const _InputIterator &__last, std::__false_type)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_valid\_range\_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence > operator+ (typename _Safe_iterator<`  
`_Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _`  
`Sequence > &__i)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence >::difference_type operator- (const _`  
`Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator,`  
`_Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`_Safe_iterator< _IteratorL, _Sequence >::difference_type operator- (const _`  
`Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _`  
`IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`  
`_CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator >`  
`&__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`

- `template<typename _Iterator , typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT , typename _Traits , typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT , typename _Traits , typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT , typename _Traits , typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT , typename _Traits , typename _Allocator >`  
`void swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

#### 4.4.1 Detailed Description

GNU debug classes for public use.

#### 4.4.2 Function Documentation

**4.4.2.1** `template<typename _Iterator > bool \_\_gnu\_debug::\_\_check\_dereferenceable ( _Iterator & )`  
`[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file functions.h.

**4.4.2.2** `template<typename _Tp > bool \_\_gnu\_debug::\_check\_dereferenceable ( const _Tp * __ptr )`  
`[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 76 of file functions.h.

**4.4.2.3** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::__check_dereferenceable ( const _Safe_iterator<  
_Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 82 of file `functions.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**4.4.2.4** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::__check_singular ( const _Safe_iterator< _Iterator,  
_Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 63 of file `functions.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.5** `template<typename _Tp > bool __gnu_debug::__check_singular (   
const _Tp * __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file `functions.h`.

**4.4.2.6** `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base *  
__x ) [inline]`

Iterators that derive from `_Safe_iterator_base` but that aren't `_Safe_iterators` can be determined singular or non-singular via `_Safe_iterator_base`.

Definition at line 48 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.7** `template<typename _CharT > const _CharT*  
__gnu_debug::__check_string ( const _CharT * __s ) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 176 of file `functions.h`.

**4.4.2.8** `template<typename _CharT, typename _Integer > const _CharT*  
__gnu_debug::_check_string ( const _CharT * __s, const _Integer  
& __n __attribute__( __unused ) ) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 164 of file `functions.h`.

**4.4.2.9** `template<typename _InputIterator > bool __gnu_debug::_valid_  
range ( const _InputIterator & __first, const _InputIterator & __last )  
[inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 134 of file `functions.h`.

References `__valid_range_aux()`.

**4.4.2.10** `template<typename _Iterator, typename _Sequence > bool  
__gnu_debug::_valid_range ( const _Safe_iterator< _Iterator,  
_Sequence > & __first, const _Safe_iterator< _Iterator, _Sequence >  
& __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 143 of file `functions.h`.

**4.4.2.11** `template<typename _Integral > bool __gnu_debug::_valid_range_  
aux ( const _Integral &, const _Integral &, std::__true_type )  
[inline]`

We say that integral types form a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file `functions.h`.

Referenced by `__valid_range()`.

**4.4.2.12** `template<typename _InputIterator > bool __gnu_debug::_valid_  
range_aux ( const _InputIterator & __first, const _InputIterator &  
__last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file `functions.h`.

References `__valid_range_aux2()`.

```
4.4.2.13 template<typename _InputIterator > bool __gnu_debug::__valid_range_aux2 ( const _InputIterator &, const _InputIterator &, std::input_iterator_tag ) [inline]
```

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file `functions.h`.

```
4.4.2.14 template<typename _RandomAccessIterator > bool __gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator & __first, const _RandomAccessIterator & __last, std::random_access_iterator_tag ) [inline]
```

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file `functions.h`.

Referenced by `__valid_range_aux()`.

## 4.5 `__gnu_internal` Namespace Reference

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 4.6 `__gnu_parallel` Namespace Reference

GNU parallel code for public use.

### Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)

*General reduction, using a binary operator.*

- struct [\\_\\_accumulate\\_selector](#)  
*std::accumulate() selector.*
- struct [\\_\\_adjacent\\_difference\\_selector](#)  
*Selector that returns the difference between two adjacent \_\_elements.*
- struct [\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- class [\\_\\_binder1st](#)  
*Similar to [std::binder1st](#), but giving the argument types explicitly.*
- class [\\_\\_binder2nd](#)  
*Similar to [std::binder2nd](#), but giving the argument types explicitly.*
- struct [\\_\\_count\\_if\\_selector](#)  
*std::count\_if() selector.*
- struct [\\_\\_count\\_selector](#)  
*std::count() selector.*
- struct [\\_\\_fill\\_selector](#)  
*std::fill() selector.*
- struct [\\_\\_find\\_first\\_of\\_selector](#)  
*Test predicate on several elements.*
- struct [\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for [std::find\(\)](#) and [std::find\\_if\(\)](#).*
- struct [\\_\\_for\\_each\\_selector](#)  
*std::for\_each() selector.*
- struct [\\_\\_generate\\_selector](#)  
*std::generate() selector.*
- struct [\\_\\_generic\\_find\\_selector](#)  
*Base class of all [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) selectors.*
- struct [\\_\\_generic\\_for\\_each\\_selector](#)

Generic `__selector` for embarrassingly parallel functions.

- struct `__identity_selector`  
*Selector that just returns the passed iterator.*
- struct `__inner_product_selector`  
*`std::inner_product()` selector.*
- struct `__max_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__min_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__mismatch_selector`  
*Test inverted predicate on a single element.*
- struct `__multiway_merge_3_variant_sentinel_switch`  
*Switch for 3-way merging with `__sentinels` turned off.*
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_4_variant_sentinel_switch`  
*Switch for 4-way merging with `__sentinels` turned off.*
- struct `__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 4-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_k_variant_sentinel_switch`  
*Switch for k-way merging with `__sentinels` turned on.*
- struct `__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for k-way merging with `__sentinels` turned off.*
- struct `__replace_if_selector`  
*`std::replace()` selector.*
- struct `__replace_selector`  
*`std::replace()` selector.*

- struct [\\_\\_transform1\\_selector](#)  
*std::transform() \_\_selector, one input sequence variant.*
- struct [\\_\\_transform2\\_selector](#)  
*std::transform() \_\_selector, two input sequences variant.*
- class [\\_\\_unary\\_negate](#)  
*Similar to [std::unary\\_negate](#), but giving the argument types explicitly.*
- struct [\\_DRandomShufflingGlobalData](#)  
*Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DRSSorterPU](#)  
*Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DummyReduct](#)  
*Reduction function doing nothing.*
- class [\\_EqualFromLess](#)  
*Constructs predicate for equality from strict weak ordering predicate.*
- struct [\\_EqualTo](#)  
*Similar to [std::equal\\_to](#), but allows two different types.*
- class [\\_GuardedIterator](#)  
*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- class [\\_IteratorPair](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_IteratorTriple](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*
- struct [\\_Job](#)  
*One \_\_job for a certain thread.*
- struct [\\_Less](#)  
*Similar to [std::less](#), but allows two different types.*

- class `_Lexicographic`  
*Compare \_\_a pair of types lexicographically, ascending.*
- class `_LexicographicReverse`  
*Compare \_\_a pair of types lexicographically, descending.*
- class `_LoserTree`  
*Stable `_LoserTree` variant.*
- class `_LoserTree< false, _Tp, _Compare >`  
*Unstable `_LoserTree` variant.*
- class `_LoserTreeBase`  
*Guarded loser/tournament tree.*
- class `_LoserTreePointer`  
*Stable `_LoserTree` implementation.*
- class `_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `_LoserTreePointerBase`  
*Base class of `_Loser Tree` implementation using pointers.*
- class `_LoserTreePointerUnguarded`  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class `_LoserTreePointerUnguarded< false, _Tp, _Compare >`  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class `_LoserTreePointerUnguardedBase`  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- struct `_LoserTreeTraits`  
*Traits for determining whether the loser tree should use pointers or copies.*
- class `_LoserTreeUnguarded`  
*Stable implementation of unguarded `_LoserTree`.*
- class `_LoserTreeUnguarded< false, _Tp, _Compare >`  
*Non-Stable implementation of unguarded `_LoserTree`.*

- class [\\_LoserTreeUnguardedBase](#)  
*Base class for unguarded [\\_LoserTree](#) implementation.*
- struct [\\_Multiplies](#)  
*Similar to [std::multiplies](#), but allows two different types.*
- struct [\\_Nothing](#)  
*Functor doing nothing.*
- struct [\\_Piece](#)  
*Subsequence description.*
- struct [\\_Plus](#)  
*Similar to [std::plus](#), but allows two different types.*
- struct [\\_PMWMSortingData](#)  
*Data accessed by all threads.*
- class [\\_PseudoSequence](#)  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class [\\_PseudoSequenceIterator](#)  
*Iterator associated with [\\_\\_gnu\\_parallel::\\_PseudoSequence](#). It features the usual random-access iterator functionality.*
- struct [\\_QSBThreadLocal](#)  
*Information local to one thread in the parallel quicksort run.*
- class [\\_RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*
- class [\\_RestrictedBoundedConcurrentQueue](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other, while [pop\\_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct [\\_SamplingSorter](#)  
*Stable sorting functor.*
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)

*Non-`__stable` sorting functor.*

- struct `_Settings`  
*class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.*
- struct `_SplitConsistently`  
*Split consistently.*
- struct `_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`  
*Split by sampling.*
- struct `_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`  
*Split by exact splitting.*
- struct `balanced_quicksort_tag`  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct `balanced_tag`  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct `constant_size_blocks_tag`  
*Selects the constant block size variant for `std::find()`.*
- struct `default_parallel_tag`  
*Recommends parallel execution using the default parallel algorithm.*
- struct `equal_split_tag`  
*Selects the equal splitting variant for `std::find()`.*
- struct `exact_tag`  
*Forces parallel merging with exact splitting, at compile time.*
- struct `find_tag`  
*Base class for for `std::find()` variants.*
- struct `growing_blocks_tag`  
*Selects the growing block size variant for `std::find()`.*
- struct `multiway_mergesort_exact_tag`  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*

- struct [multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [quicksort\\_tag](#)  
*Forces parallel sorting using unbalanced quicksort at compile time.*
- struct [sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [sequential\\_tag](#)  
*Forces sequential execution at compile time.*
- struct [unbalanced\\_tag](#)  
*Recommends parallel execution using static load-balancing at compile time.*

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
-

- enum `_FindAlgorithm` { `GROWING_BLOCKS`, `CONSTANT_SIZE_BLOCKS`, `EQUAL_SPLIT` }
- enum `_MultiwayMergeAlgorithm` { `LOSER_TREE` }
- enum `_Parallelism` {  
`sequential`, `parallel_unbalanced`, `parallel_balanced`, `parallel_omp_loop`,  
`parallel_omp_loop_static`, `parallel_taskqueue` }
- enum `_PartialSumAlgorithm` { `RECURSIVE`, `LINEAR` }
- enum `_SortAlgorithm` { `MWMS`, `QS`, `QS_BALANCED` }
- enum `_SplittingAlgorithm` { `SAMPLING`, `EXACT` }

## Functions

- `template<typename _RAIter, typename _DifferenceTp >`  
`void __calc_borders` (`_RAIter __elements`, `_DifferenceTp __length`, `_DifferenceTp *__off`)
- `template<typename _Tp >`  
`bool __compare_and_swap` (`volatile _Tp *__ptr`, `_Tp __comparand`, `_Tp __replacement`)
- `bool __compare_and_swap_32` (`volatile int32_t *__ptr`, `int32_t __comparand`, `int32_t __replacement`)
- `bool __compare_and_swap_64` (`volatile int64_t *__ptr`, `int64_t __comparand`, `int64_t __replacement`)
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator __copy_tail` (`std::pair< _Iter, _Iter > __b`, `std::pair< _Iter, _Iter > __e`, `_OutputIterator __r`)
- `void __decode2` (`_CASable __x`, `int &__a`, `int &__b`)
- `template<typename _RAIter, typename _DifferenceTp >`  
`void __determine_samples` (`_PMWMSortingData< _RAIter > *__sd`, `_DifferenceTp __num_samples`)
- `_CASable __encode2` (`int __a`, `int __b`)
- `template<typename _Tp >`  
`_Tp __fetch_and_add` (`volatile _Tp *__ptr`, `_Tp __addend`)
- `int32_t __fetch_and_add_32` (`volatile int32_t *__ptr`, `int32_t __addend`)
- `int64_t __fetch_and_add_64` (`volatile int64_t *__ptr`, `int64_t __addend`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`, `_Selector __selector`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`, `_Selector __selector`, `equal_split_tag`)

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing\_blocks\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant\_size\_blocks\_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result >`  
`_UserOp \_\_for\_each\_template\_random\_access (_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator\_traits< _Iter >::difference_type __bound, \_Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op \_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop\_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op \_\_for\_each\_template\_random\_access\_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `\_ThreadIndex \_\_get\_max\_threads ()`
- `bool \_\_is\_parallel (const \_Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool \_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter \_\_median\_of\_three\_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits<_RAIter >::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng= _RandomNumberGenerator())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, type-`

- ```

name std::iterator\_traits< _RAIter >::difference_type __n, \_ThreadIndex __-
num_threads, \_RandomNumberGenerator &__rng)

```
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs\_pu (\_DRSSorterPU< _RAIter, \_-  
RandomNumberGenerator > *__pus)`
  - `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, \_-  
Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
  - `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, \_-  
Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
  - `template<typename _Iter, typename _OutputIterator, typename Operation >`  
`_OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, \_-  
Iter __begin2, _Iter __end2, _OutputIterator __result, Operation __op)`
  - `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter  
__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare \_-  
comp)`
  - `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter \_-  
__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
parallel\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
balanced\_quicksort\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
multiway\_mergesort\_sampling\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_-  
Parallelism __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
multiway\_mergesort\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
multiway\_mergesort\_exact\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
quicksort\_tag __parallelism)`
  - `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,  
default\_parallel\_tag __parallelism)`

- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_`  
`RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_`  
`traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_`  
`traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`_ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _`  
`OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _`  
`OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_conquer (_QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin,`  
`_RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __`  
`num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __`  
`begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > ** __tls,`  
`_Compare &__comp, _ThreadIndex __iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`  
`__RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __`  
`RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, type-`  
`name _DifferenceTp, typename _Compare >`  
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_`  
`traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type`  
`>::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_sequential\_random\_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rng)`
- `template<typename _Iter >`  
`void \_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two,`  
`size_t &__range_length)`
- `template<typename _Iter >`  
`void \_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__-`  
`count_to_two, size_t &__range_length, const bool __make_twice)`
- `void \_\_yield ()`
- `template<typename _DifferenceType, typename _OutputIterator >`  
`_OutputIterator equally\_split (_DifferenceType __n, \_ThreadIndex __num_`  
`threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`  
`_DifferenceType equally\_split\_point (_DifferenceType __n, \_ThreadIndex __-`  
`num_threads, \_ThreadIndex __thread_no)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t list\_partition (const _Iter __begin, const _Iter __end, _Iter *__-`  
`starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __-`  
`oversampling=0)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename`  
`_Compare >`  
`void multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_`  
`seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __-`  
`comp=std::less< typename std::iterator_traits< typename std::iterator_traits<`  
`_RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _`  
`RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp`  
`>())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway\_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, default\_parallel\_tag __tag)`

- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >`  
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename`

- [std::iterator\\_traits](#)< typename [std::iterator\\_traits](#)< [\\_RAIterIterator](#) >::value\_type::first\_type >::value\_type &\_\_sentinel, [\\_DifferenceTp](#) \_\_length, [\\_Compare](#) \_\_comp)
- `template<bool __stable, typename \_RAIterIterator , typename \_Compare , typename \_DifferenceType >`  
`void multiway\_merge\_sampling\_splitting (\_RAIterIterator __seqs_begin, \_RAIterIterator __seqs_end, \_DifferenceType __length, \_DifferenceType __total_length, \_Compare __comp, std::vector< std::pair< \_DifferenceType, \_DifferenceType > > *__pieces)`
  - `template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp , typename \_Compare >`  
`\_RAIterOut multiway\_merge\_sentinels (\_RAIterPairIterator __seqs_begin, \_RAIterPairIterator __seqs_end, \_RAIterOut __target, \_DifferenceTp __length, \_Compare __comp, gnu\_parallel::sequential\_tag)`
  - `template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp , typename \_Compare >`  
`\_RAIterOut multiway\_merge\_sentinels (\_RAIterPairIterator __seqs_begin, \_RAIterPairIterator __seqs_end, \_RAIterOut __target, \_DifferenceTp __length, \_Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
  - `template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp , typename \_Compare >`  
`\_RAIterOut multiway\_merge\_sentinels (\_RAIterPairIterator __seqs_begin, \_RAIterPairIterator __seqs_end, \_RAIterOut __target, \_DifferenceTp __length, \_Compare __comp, gnu\_parallel::exact\_tag __tag)`
  - `template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp , typename \_Compare >`  
`\_RAIterOut multiway\_merge\_sentinels (\_RAIterPairIterator __seqs_begin, \_RAIterPairIterator __seqs_end, \_RAIterOut __target, \_DifferenceTp __length, \_Compare __comp, default\_parallel\_tag __tag)`
  - `template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp , typename \_Compare >`  
`\_RAIterOut multiway\_merge\_sentinels (\_RAIterPairIterator __seqs_begin, \_RAIterPairIterator __seqs_end, \_RAIterOut __target, \_DifferenceTp __length, \_Compare __comp, sampling\_tag __tag)`
  - `template<bool __stable, bool __sentinels, typename \_RAIterIterator , typename \_RAIter3 , typename \_DifferenceTp , typename \_Splitter , typename \_Compare >`  
`\_RAIter3 parallel\_multiway\_merge (\_RAIterIterator __seqs_begin, \_RAIterIterator __seqs_end, \_RAIter3 __target, \_Splitter __splitter, \_DifferenceTp __length, \_Compare __comp, ThreadIndex __num_threads)`
  - `template<bool __stable, bool __exact, typename \_RAIter , typename \_Compare >`  
`void parallel\_sort\_mwms (\_RAIter __begin, \_RAIter __end, \_Compare __comp, ThreadIndex __num_threads)`
  - `template<bool __stable, bool __exact, typename \_RAIter , typename \_Compare >`  
`void parallel\_sort\_mwms\_pu (PMWMSortingData< \_RAIter > *__sd, \_Compare &__comp)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`

- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.

### 4.6.2 Typedef Documentation

#### 4.6.2.1 typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 4.6.2.2 typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

#### 4.6.2.3 typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

#### 4.6.2.4 typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 `enum __gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file `types.h`.

#### 4.6.3.2 `enum __gnu_parallel::_FindAlgorithm`

Find algorithms:

Definition at line 106 of file `types.h`.

#### 4.6.3.3 `enum __gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

Definition at line 85 of file `types.h`.

#### 4.6.3.4 `enum __gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

##### Enumerator:

*sequential* Not parallel.

*parallel\_unbalanced* Parallel unbalanced (equal-sized chunks).

*parallel\_balanced* Parallel balanced (work-stealing).

*parallel\_omp\_loop* Parallel with OpenMP dynamic load-balancing.

*parallel\_omp\_loop\_static* Parallel with OpenMP static load-balancing.

*parallel\_taskqueue* Parallel with OpenMP taskqueue construct.

Definition at line 44 of file `types.h`.

#### 4.6.3.5 `enum __gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

Definition at line 91 of file `types.h`.

#### 4.6.3.6 enum `__gnu_parallel::_SortAlgorithm`

Sorting algorithms:

Definition at line 76 of file `types.h`.

#### 4.6.3.7 enum `__gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: `sampling`, `__exact`.

Definition at line 98 of file `types.h`.

### 4.6.4 Function Documentation

**4.6.4.1** `template<typename _RAIter, typename _DifferenceTp > void  
__gnu_parallel::_calc_borders ( _RAIter __elements, _DifferenceTp  
__length, _DifferenceTp * __off )`

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

#### Parameters

*\_\_elements* Begin iterator of sequence to search for.

*\_\_length* Length of sequence to search for.

*\_\_advances* Returned `__offsets`.

Definition at line 51 of file `search.h`.

Referenced by `__search_template()`.

**4.6.4.2** `template<typename _Tp > bool __gnu_parallel::_compare_and_swap  
( volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement )  
[inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

#### Parameters

*\_\_ptr* Pointer to signed integer.

*\_\_comparand* Compare value.

*\_\_replacement* Replacement value.

Definition at line 337 of file `parallel/compatibility.h`.

References `__compare_and_swap_32()`, and `__compare_and_swap_64()`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_front()`.

#### 4.6.4.3 `bool __gnu_parallel::__compare_and_swap_32 ( volatile int32_t* __ptr, int32_t __comparand, int32_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

##### Parameters

`__ptr` Pointer to 32-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 240 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

#### 4.6.4.4 `bool __gnu_parallel::__compare_and_swap_64 ( volatile int64_t* __ptr, int64_t __comparand, int64_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

##### Parameters

`__ptr` Pointer to 64-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 283 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

#### 4.6.4.5 `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int & __b ) [inline]`

Decode two integers from one `gnu_parallel::_CASable`.

**Parameters**

- `__x` [\\_\\_gnu\\_parallel::\\_CASable](#) to decode integers from.
- `__a` First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.
- `__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

**See also**

[\\_\\_encode2](#)

Definition at line 133 of file `parallel/base.h`.

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

#### 4.6.4.6 `template<typename _RAIter , typename _DifferenceTp > void __gnu_parallel::_determine_samples ( _PMWMSortingData< _RAIter > * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

**Parameters**

- `__sd` Pointer to algorithm data. `_Result` will be placed in `__sd->_M_samples`.
- `__num_samples` Number of `_M_samples` to select.

Definition at line 97 of file `multiway_mergesort.h`.

References `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_samples`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_starts`, and `equally_split()`.

#### 4.6.4.7 `_CASable __gnu_parallel::_encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

**Parameters**

- `__a` First integer, to be encoded in the most-significant `_CASable_bits/2` bits.
- `__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits.

**Returns**

value encoding `__a` and `__b`.

**See also**

[\\_\\_decode2](#)

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece >::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece >::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece >::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece >::push_front()`.

#### 4.6.4.8 `template<typename _Tp > _Tp __gnu_parallel::_fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a signed integer.

`__addend` Value to add.

Definition at line 186 of file `parallel/compatibility.h`.

References `__fetch_and_add_32()`, and `__fetch_and_add_64()`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece >::push_front()`.

#### 4.6.4.9 `int32_t __gnu_parallel::_fetch_and_add_32 ( volatile int32_t * __ptr, int32_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a 32-bit signed integer.

`__addend` Value to add.

Definition at line 95 of file parallel/compatibility.h.

Referenced by `__fetch_and_add()`.

**4.6.4.10** `int64_t __gnu_parallel::__fetch_and_add_64 ( volatile int64_t * __ptr, int64_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to a 64-bit signed integer.

`__addend` Value to add.

Definition at line 134 of file parallel/compatibility.h.

Referenced by `__fetch_and_add()`.

**4.6.4.11** `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector ) [inline]`

Parallel `std::find`, switch for different algorithms.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Must have same length as first sequence.

`__pred` Find predicate.

`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

Definition at line 60 of file find.h.

References `__gnu_parallel::_Settings::get()`.

**4.6.4.12** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,  
equal_split_tag )`

Parallel `std::find`, equal splitting variant.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.

`__pred` Find predicate.

`__selector` `_Functionality` (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

Definition at line 97 of file `find.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

**4.6.4.13** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,  
growing_blocks_tag )`

Parallel `std::find`, growing block size variant.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.

`__pred` Find predicate.

`__selector` `_Functionality` (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_initial\\_block\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_maximum\\_block\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_increasing\\_factor](#)

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 187 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_increasing_factor`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_maximum_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.14** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag )`

Parallel `std::find`, constant block size variant.

**Parameters**

*\_\_begin1* Begin iterator of first sequence.  
*\_\_end1* End iterator of first sequence.  
*\_\_begin2* Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.  
*\_\_pred* Find predicate.  
*\_\_selector* `_Functionality` (e. g. `std::find_if()`, `std::equal()`,...)

**Returns**

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.15** `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::_for_each_template_random_access ( _Iter  
__begin, _Iter __end, _UserOp __user_op, _Functionality &  
__functionality, _Red __reduction, _Result __reduction_start, _Result  
& __output, typename std::iterator_traits< _Iter >::difference_type  
__bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

#### Parameters

`__begin` Begin iterator of input sequence.

`__end` End iterator of input sequence.

`__user_op` A user-specified functor (comparator, predicate, associative operator,...)

`__functionality` functor to *process* an element with `__user_op` (depends on desired functionality, e. g. `accumulate`, `for_each`,...)

`__reduction` Reduction functor.

`__reduction_start` Initial value for reduction.

`__output` Output iterator.

`__bound` Maximum number of elements processed.

`__parallelism_tag` Parallelization method

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

**4.6.4.16** `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::_for_each_template_random_access_ed ( _RAIter  
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
__base, _Result & __output, typename std::iterator_traits< _RAIter  
>::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, ...)
- `__f` Functor to "process" an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to "add" a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.
- `__output` Pointer to position where final result is written to
- `__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.17** `template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_omp_loop (  
_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, etc.).
- `__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to *add* a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.

`__output` Pointer to position where final result is written to

`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

```
4.6.4.18 template<typename _RAIter, typename _Op, typename _Fu,
typename _Red, typename _Result > _Op __gnu_parallel::__-
for_each_template_random_access_omp_loop_static ( _RAIter
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result
__base, _Result & __output, typename std::iterator_traits< _RAIter
>::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

### Parameters

`__begin` Begin iterator of element sequence.

`__end` End iterator of element sequence.

`__o` User-supplied functor (comparator, predicate, adding functor, ...).

`__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).

`__r` Functor to *add* a single `__result` to the already processed `__elements` (depends on functionality).

`__base` Base value for reduction.

`__output` Pointer to position where final result is written to

`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

**4.6.4.19** `template<typename _RAIter , typename _Op , typename  
_Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_workstealing (   
_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

#### Parameters

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__op` User-supplied functor (comparator, predicate, adding functor, ...).
- `__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to *add* a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.
- `__output` Pointer to position where final result is written to
- `__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Job< _DifferenceTp >::_M_first`, `__gnu_parallel::_Job< _DifferenceTp >::_M_last`, `__gnu_parallel::_Job< _DifferenceTp >::_M_load`, `__gnu_parallel::_Settings::cache_line_size`, `__gnu_parallel::_Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.20** `template<typename _IIter , typename _Compare > bool  
__gnu_parallel::__is_sorted ( _IIter __begin, _IIter __end,  
_Compare __comp )`

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

#### Parameters

- `__begin` Begin iterator of sequence.

`__end` End iterator of sequence.  
`__comp` Comparator.

**Returns**

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

**4.6.4.21** `template<typename _RAIter, typename _Compare > _RAIter  
 __gnu_parallel::__median_of_three_iterators ( _RAIter __a, _RAIter  
 __b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to `__comp`.

**Parameters**

`__a` First iterator.  
`__b` Second iterator.  
`__c` Third iterator.  
`__comp` Comparator.

Definition at line 398 of file `parallel/base.h`.

Referenced by `__qsb_divide()`.

**4.6.4.22** `template<typename _RAIter1, typename _RAIter2, typename  
 _OutputIterator, typename _DifferenceTp, typename _Compare  
 > _OutputIterator __gnu_parallel::__merge_advance ( _RAIter1  
 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
 __end2, _OutputIterator __target, _DifferenceTp __max_length,  
 _Compare __comp ) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

**Parameters**

`__begin1` Begin iterator of first sequence.

***\_\_end1*** End iterator of first sequence.  
***\_\_begin2*** Begin iterator of second sequence.  
***\_\_end2*** End iterator of second sequence.  
***\_\_target*** Target begin iterator.  
***\_\_max\_length*** Maximum number of elements to merge.  
***\_\_comp*** Comparator.

#### Returns

Output end iterator.

Definition at line 171 of file merge.h.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

**4.6.4.23** `template<typename _RAIter1, typename _RAIter2, typename  
 _OutputIterator, typename _DifferenceTp, typename _Compare >  
 _OutputIterator __gnu_parallel::__merge_advance_movc ( _RAIter1  
 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
 __end2, _OutputIterator __target, _DifferenceTp __max_length,  
 _Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

#### Parameters

***\_\_begin1*** Begin iterator of first sequence.  
***\_\_end1*** End iterator of first sequence.  
***\_\_begin2*** Begin iterator of second sequence.  
***\_\_end2*** End iterator of second sequence.  
***\_\_target*** Target begin iterator.  
***\_\_max\_length*** Maximum number of elements to merge.  
***\_\_comp*** Comparator.

#### Returns

Output end iterator.

Definition at line 105 of file merge.h.

Referenced by `__merge_advance()`.

```
4.6.4.24 template<typename _RAIter1 , typename _RAIter2 , typename
    _OutputIterator , typename _DifferenceTp , typename _Compare >
    _OutputIterator __gnu_parallel::__merge_advance_usual ( _RAIter1
    & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2
    __end2, _OutputIterator __target, _DifferenceTp __max_length,
    _Compare __comp )
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.
- `__target` Target begin iterator.
- `__max_length` Maximum number of elements to merge.
- `__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

```
4.6.4.25 template<typename _RAIter1 , typename _RAIter2 ,
    typename _RAIter3 , typename _Compare > _RAIter3
    __gnu_parallel::__parallel_merge_advance ( _RAIter1 & __begin1,
    _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2,
    _RAIter3 __target, typename std::iterator_traits< _RAIter1
    >::difference_type __max_length, _Compare __comp ) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.

*\_\_target* Target begin iterator.  
*\_\_max\_length* Maximum number of elements to merge.  
*\_\_comp* Comparator.

### Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

```
4.6.4.26 template<typename _RAIter1 , typename _RAIter3 , typename
          _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (
          _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2,
          _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<
          _RAIter1 >::difference_type __max_length, _Compare __comp )
          [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

### Parameters

*\_\_begin1* Begin iterator of first sequence.  
*\_\_end1* End iterator of first sequence.  
*\_\_begin2* Begin iterator of second sequence.  
*\_\_end2* End iterator of second sequence.  
*\_\_target* Target begin iterator.  
*\_\_max\_length* Maximum number of elements to merge.  
*\_\_comp* Comparator.

### Returns

Output end iterator.

Definition at line 223 of file merge.h.

References `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.27** `template<typename _RAIter, typename _Compare > void  
__gnu_parallel::__parallel_nth_element ( _RAIter __begin, _RAIter  
__nth, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::nth_element()`.

#### Parameters

`__begin` Begin iterator of input sequence.

`__nth` Iterator of element that must be in position afterwards.

`__end` End iterator of input sequence.

`__comp` Comparator.

Definition at line 336 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, `__gnu_parallel::_Settings::partition_minimal_n`, and `std::swap()`.

Referenced by `__parallel_partial_sort()`.

**4.6.4.28** `template<typename _RAIter, typename _Compare > void  
__gnu_parallel::__parallel_partial_sort ( _RAIter __begin, _RAIter  
__middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

#### Parameters

`__begin` Begin iterator of input sequence.

`__middle` Sort until this position.

`__end` End iterator of input sequence.

`__comp` Comparator.

Definition at line 426 of file `partition.h`.

References `__parallel_nth_element()`.

**4.6.4.29** `template<typename _IIter, typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum ( _IIter __begin, _IIter  
__end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-`__end`.

**Parameters**

- \_\_begin* Begin iterator of input sequence.
- \_\_end* End iterator of input sequence.
- \_\_result* Begin iterator of output sequence.
- \_\_bin\_op* Associative binary function.

**Returns**

End iterator of output sequence.

Definition at line 200 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.30** `template<typename _Iiter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::_parallel_partial_sum_basecase ( _Iiter __begin,  
_Iiter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iiter >::value_type __value )`

Base case prefix sum routine.

**Parameters**

- \_\_begin* Begin iterator of input sequence.
- \_\_end* End iterator of input sequence.
- \_\_result* Begin iterator of output sequence.
- \_\_bin\_op* Associative binary function.
- \_\_value* Start value. Must be passed since the neutral element is unknown in general.

**Returns**

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

**4.6.4.31** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_linear ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::difference_type __n )`

Parallel partial sum implementation, two-phase approach, no recursion.

#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__result` Begin iterator of output sequence.
- `__bin_op` Associative binary function.
- `__n` Length of sequence.
- `__num_threads` Number of threads to use.

#### Returns

End iterator of output sequence.

Definition at line 90 of file `partial_sum.h`.

References `__parallel_partial_sum_basecase()`, `equally_split()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

**4.6.4.32** `template<typename _RAIter , typename _Predicate  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__parallel_partition ( _RAIter __begin, _RAIter  
__end, _Predicate __pred, _ThreadIndex __num_threads )`

Parallel implementation of `std::partition`.

#### Parameters

- `__begin` Begin iterator of input sequence to split.
- `__end` End iterator of input sequence to split.
- `__pred` Partition predicate, possibly including some kind of pivot.
- `__num_threads` Maximum number of threads to use for this task.

#### Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::partition_chunk_share`, `__gnu_parallel::_Settings::partition_chunk_size`, and `std::swap()`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

```
4.6.4.33 template<typename _RAIter , typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle ( _RAIter
__begin, _RAIter __end, _RandomNumberGenerator __rng =
__RandomNumber ( ) ) [inline]
```

Parallel random public call.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__rng` Random number generator to use.

Definition at line 517 of file random\_shuffle.h.

References `__parallel_random_shuffle_drs()`.

```
4.6.4.34 template<typename _RAIter , typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle_drs ( _RAIter
__begin, _RAIter __end, typename std::iterator_traits<
_RAIter >::difference_type __n, _ThreadIndex __num_threads,
_RandomNumberGenerator & __rng )
```

Main parallel random shuffle step.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__n` Length of sequence.
- `__num_threads` Number of threads to use.
- `__rng` Random number generator to use.

Definition at line 263 of file random\_shuffle.h.

References `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_bins_end`, `__parallel_random_shuffle_drs_pu()`, `__rd_log2()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `_GLIBCXX_CALL`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_bin_proc`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_bins_begin`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bits`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_num_threads`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_sd`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_seed`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_starts`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_temporaries`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle()`.

**4.6.4.35** `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::_parallel_random_shuffle_drs_pu (   
     __DRSSorterPU<_RAIter, _RandomNumberGenerator> * __pus )`

Random shuffle code executed by each thread.

#### Parameters

`__pus` Array of thread-local data records.

Definition at line 122 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bits`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_num_threads`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_sd`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_seed`, and `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_starts`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.36** `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::_parallel_sort (   
     _RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

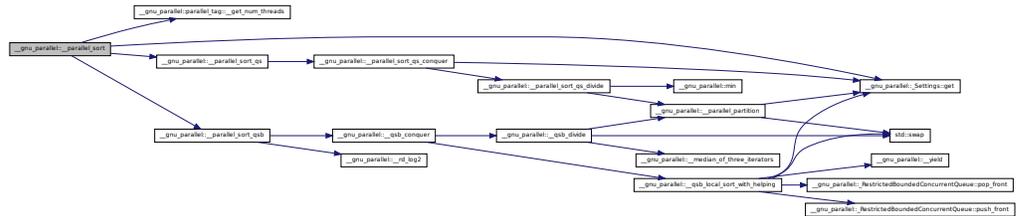
**Parameters**

- \_\_begin*** Begin iterator of input sequence.
- \_\_end*** End iterator of input sequence.
- \_\_comp*** Comparator.
- \_\_stable*** Sort *\_\_stable*.

Definition at line 198 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



**4.6.4.37** `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter`  
`__end, _Compare __comp, balanced_quicksort_tag __parallelism )`  
`[inline]`

Choose balanced quicksort for parallel sorting.

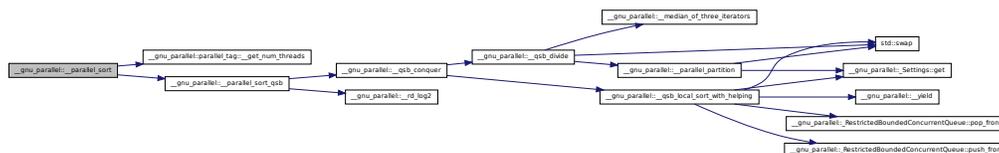
**Parameters**

- \_\_begin*** Begin iterator of input sequence.
- \_\_end*** End iterator of input sequence.
- \_\_comp*** Comparator.
- \_\_stable*** Sort *\_\_stable*.

Definition at line 157 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



```
4.6.4.38  template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter
    __end, _Compare __comp, multiway_mergesort_sampling_tag
    __parallelism ) [inline]
```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

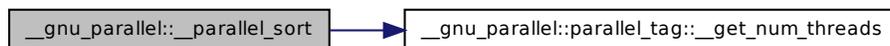
#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 117 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.39** `template<bool __stable, typename _RAIter , typename _Compare >  
void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
__end, _Compare __comp, multiway_mergesort_tag __parallelism )  
[inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

#### Parameters

*\_\_begin* Begin iterator of input sequence.

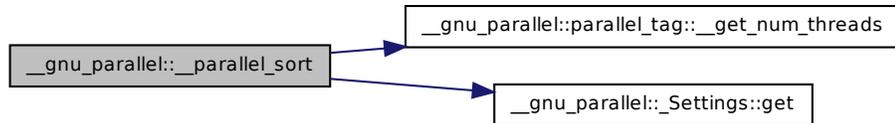
*\_\_end* End iterator of input sequence.

*\_\_comp* Comparator.

Definition at line 74 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`,  
and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



**4.6.4.40** `template<bool __stable, typename _RAIter , typename _Compare >  
void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
__end, _Compare __comp, multiway_mergesort_exact_tag  
__parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

*\_\_begin* Begin iterator of input sequence.

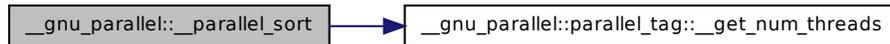
*\_\_end* End iterator of input sequence.

*\_\_comp* Comparator.

Definition at line 97 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



```

4.6.4.41 template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter
__end, _Compare __comp, quicksort_tag __parallelism )
[inline]
  
```

Choose quicksort for parallel sorting.

#### Parameters

`__begin` Begin iterator of input sequence.

`__end` End iterator of input sequence.

`__comp` Comparator.

Definition at line 136 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.42** `template<bool __stable, typename _RAIter , typename _Compare >  
void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
__end, _Compare __comp, default_parallel_tag __parallelism )  
[inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

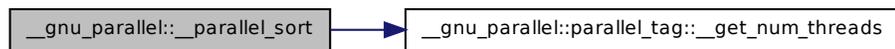
#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 178 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.43** `template<typename _RAIter , typename _Compare > void  
__gnu_parallel::__parallel_sort_qs ( _RAIter __begin, _RAIter  
__end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort main call.

#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator input sequence, ignored.
- `__comp` Comparator.
- `__num_threads` Number of threads that are allowed to work on this part.

Definition at line 152 of file `quicksort.h`.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

**4.6.4.44** `template<typename _RAIter, typename _Compare > void  
__gnu_parallel::__parallel_sort_qs_conquer ( _RAIter __begin,  
_RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort conquer step.

#### Parameters

`__begin` Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 99 of file `quicksort.h`.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

**4.6.4.45** `template<typename _RAIter, typename _Compare  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__parallel_sort_qs_divide ( _RAIter __begin,  
_RAIter __end, _Compare __comp, typename std::iterator_traits<  
_RAIter >::difference_type __pivot_rank, typename  
std::iterator_traits<_RAIter >::difference_type __num_samples,  
_ThreadIndex __num_threads )`

Unbalanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__pivot_rank` Desired `__rank` of the pivot.

`__num_samples` Choose pivot from that many samples.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 51 of file `quicksort.h`.

References `__parallel_partition()`, and `min()`.

Referenced by `__parallel_sort_qs_conquer()`.

**4.6.4.46** `template<typename _RAIter, typename _Compare > void  
 __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, _ThreadIndex __num_threads )`

Top-level quicksort routine.

#### Parameters

*\_\_begin* Begin iterator of sequence.

*\_\_end* End iterator of sequence.

*\_\_comp* Comparator.

*\_\_num\_threads* Number of threads that are allowed to work on this part.

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

**4.6.4.47** `template<typename _IIter, class _OutputIterator > _OutputIterator  
 __gnu_parallel::__parallel_unique_copy ( _IIter __first, _IIter  
 __last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

#### Parameters

*\_\_first* Begin iterator of input sequence.

*\_\_last* End iterator of input sequence.

*\_\_result* Begin iterator of result `__sequence`.

#### Returns

End iterator of result `__sequence`.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

**4.6.4.48** `template<typename _IIter, class _OutputIterator, class  
 _BinaryPredicate > _OutputIterator __gnu_parallel::__parallel_  
 unique_copy ( _IIter __first, _IIter __last, _OutputIterator __result,  
 _BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/ `__o` explicit equality predicate.

**Parameters**

- `__first` Begin iterator of input sequence.
- `__last` End iterator of input sequence.
- `__result` Begin iterator of result `__sequence`.
- `__binary_pred` Equality predicate.

**Returns**

End iterator of result `__sequence`.

Definition at line 50 of file `unique_copy.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

Referenced by `__parallel_unique_copy()`.

**4.6.4.49** `template<typename _RAIter, typename _Compare > void  
 __gnu_parallel::__qsb_conquer ( _QSBThreadLocal< _RAIter >  
 ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp,  
 _ThreadIndex __iam, _ThreadIndex __num_threads, bool  
 __parent_wait )`

Quicksort conquer step.

**Parameters**

- `__tls` Array of thread-local storages.
- `__begin` Begin iterator of subsequence.
- `__end` End iterator of subsequence.
- `__comp` Comparator.
- `__iam` Number of the thread processing this function.
- `__num_threads` Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`.

Referenced by `__parallel_sort_qsb()`.

**4.6.4.50** `template<typename _RAIter , typename _Compare  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end,  
_Compare __comp, _ThreadIndex __num_threads )`

Balanced quicksort divide step.

#### Parameters

*\_\_begin* Begin iterator of subsequence.  
*\_\_end* End iterator of subsequence.  
*\_\_comp* Comparator.  
*\_\_num\_threads* Number of threads that are allowed to work on this part.

#### Precondition

$(\text{__end} - \text{__begin}) \geq 1$

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, `__parallel_partition()`, and `std::swap()`.

Referenced by `__qsb_conquer()`.

**4.6.4.51** `template<typename _RAIter , typename _Compare > void  
__gnu_parallel::__qsb_local_sort_with_helping ( _QSBThreadLocal<  
_RAIter > ** __tls, _Compare & __comp, _ThreadIndex __iam,  
bool __wait )`

Quicksort step doing load-balanced local sort.

#### Parameters

*\_\_tls* Array of thread-local storages.  
*\_\_comp* Comparator.  
*\_\_iam* Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front()`, `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`, and `std::swap()`.

Referenced by `__qsb_conquer()`.

**4.6.4.52** `template<typename _RandomNumberGenerator > int  
__gnu_parallel::__random_number_pow2 ( int __logp,  
_RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

`__logp` Logarithm (basis 2) of the upper range `__bound`.

`__rng` Random number generator to use.

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**4.6.4.53** `template<typename _Size > _Size __gnu_parallel::__rd_log2 ( _Size  
__n ) [inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

#### Parameters

`__n` Argument.

#### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.54** `template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2  
( _Tp __x )`

Round up to the next greater power of 2.

#### Parameters

`__x` Integer to round up

Definition at line 246 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

**4.6.4.55** `template<typename __RAIter1 , typename __RAIter2 , typename  
__Pred > __RAIter1 __gnu_parallel::__search_template ( __RAIter1  
__begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2  
__end2, __Pred __pred )`

Parallel `std::search`.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.
- `__pred` Find predicate.

#### Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `_GLIBCXX_CALL`, `equally_split()`, and `min()`.

**4.6.4.56** `template<bool __stable, bool __sentinels, typename __RAIterIterator ,  
typename __RAIter3 , typename __DifferenceTp , typename __Compare  
> __RAIter3 __gnu_parallel::__sequential_multiway_merge (   
__RAIterIterator __seqs_begin, __RAIterIterator __seqs_end,  
__RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< __RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, __DifferenceTp __length, __Compare  
__comp )`

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

#### Parameters

- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, possibly larger than the number of elements available.

`__stable` Stable merging incurs a performance penalty.

`__sentinel` The sequences have `__a` `__sentinel` element.

### Returns

End iterator of output sequence.

Definition at line 917 of file `multiway_merge.h`.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**4.6.4.57** `template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__sequential_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rng )`

Sequential cache-efficient random shuffle.

### Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__rng` Random number generator to use.

Definition at line 408 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.58** `template<typename _IIter > void __gnu_parallel::__shrink ( std::vector<_IIter > & __os_starts, size_t & __count_to_two, size_t & __range_length )`

Combines two ranges into one and thus halves the number of ranges.

### Parameters

`__os_starts` Start positions worked on (oversampled).

*\_\_count\_to\_two* Counts up to 2.  
*\_\_range\_length* Current length of a chunk.

Definition at line 70 of file list\_partition.h.

References `std::vector<_Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

**4.6.4.59** `template<typename _Iter > void __gnu_parallel::__shrink_and_double ( std::vector<_Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

#### Parameters

*\_\_os\_starts* Start positions worked on (oversampled).  
*\_\_count\_to\_two* Counts up to 2.  
*\_\_range\_length* Current length of a chunk.  
*\_\_make\_twice* Whether the `__os_starts` is allowed to be grown or not

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector<_Tp, _Alloc >::resize()`, and `std::vector<_Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

**4.6.4.60** `void __gnu_parallel::__yield ( ) [inline]`

Yield the control to another thread, without waiting for the end to the time slice.

Definition at line 354 of file parallel/compatibility.h.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

**4.6.4.61** `template<typename _DifferenceType, typename _OutputIterator > _OutputIterator __gnu_parallel::equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts  
`__s` Splitters

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**4.6.4.62** `template<typename _DifferenceType > _DifferenceType  
 __gnu_parallel::equally_split_point ( _DifferenceType __n,  
 _ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts

**Returns**

splitting point

Definition at line 74 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

**4.6.4.63** `template<typename _IIter , typename _FuncType > size_t  
 __gnu_parallel::list_partition ( const _IIter __begin, const _IIter  
 __end, _IIter * __starts, size_t * __lengths, const int __num_parts,  
 _FuncType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

**Parameters**

*\_\_begin* Begin iterator of input sequence.

*\_\_end* End iterator of input sequence.

*\_\_starts* Start iterators for the resulting parts, dimension `__num_parts+1`. For convenience, `__starts [__num_parts]` contains the end iterator of the sequence.

*\_\_lengths* Length of the resulting parts.

*\_\_num\_parts* Number of parts to split the sequence into.

*\_\_f* Functor to be applied to each element by traversing `__it`

*\_\_oversampling* Oversampling factor. If 0, then the partitions will differ in at most  $\{ \{ \_end \} - \{ \_begin \} \}$  `__elements`. Otherwise, the ratio between the longest and the shortest part is bounded by  $1 / ( \{ \_oversampling \} \{ num \} )$

**Returns**

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc >::size()`.

**4.6.4.64** `template<typename _Tp > const _Tp& __gnu_parallel::max ( const _Tp & __a, const _Tp & __b )`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

Referenced by `__parallel_nth_element()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.65** `template<typename _Tp > const _Tp& __gnu_parallel::min ( const _Tp & __a, const _Tp & __b )`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`, `__parallel_sort_qs_divide()`, `__search_template()`, `multiseq_partition()`, and `multiseq_selection()`.

```

4.6.4.66 template<typename _RanSeqs , typename _RankType ,
            typename _RankIterator , typename _Compare > void __gnu_
            parallel::multiseq_partition ( _RanSeqs __begin_seqs, _RanSeqs
            __end_seqs, _RankType __rank, _RankIterator __begin_offsets,
            _Compare __comp = std::less< typename std::iterator_
            traits<typename std::iterator_traits<_RanSeqs>::value_
            type:: first_type>::value_type> ()
            )

```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

#### Parameters

`__begin_seqs` Begin of the sequence of iterator pairs.

`__end_seqs` End of the sequence of iterator pairs.

`__rank` The global rank to partition at.

`__begin_offsets` A random-access `__sequence` `__begin` where the `__result` will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective `__sequence`.

`__comp` The ordering functor, defaults to `std::less<_Tp>`.

Definition at line 124 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc >::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare >::empty()`, `std::vector<_Tp, _Alloc >::end()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare >::push()`, `std::vector<_Tp, _Alloc >::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare >::top()`.

Referenced by `multiway_merge_exact_splitting()`.

```

4.6.4.67 template<typename _Tp , typename _RanSeqs , typename _RankType
            , typename _Compare > _Tp __gnu_parallel::multiseq_selection (
            _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,
            _RankType & __offset, _Compare __comp = std::less<_Tp> () )

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

**Parameters**

- `__begin_seqs` Begin of the sequence of iterator pairs.
- `__end_seqs` End of the sequence of iterator pairs.
- `__rank` The global rank to partition at.
- `__offset` The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
- `__comp` The ordering functor, defaults to `std::less`.

Definition at line 390 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc >::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare >::empty()`, `std::vector<_Tp, _Alloc >::end()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare >::push()`, `std::vector<_Tp, _Alloc >::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare >::top()`.

**4.6.4.68** `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare > _RAIterOut`  
`__gnu_parallel::multiway_merge ( _RAIterPairIterator`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`  
`__target, _DifferenceTp __length, _Compare __comp,`  
`__gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 10; ++__j)
        sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                    sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

### See also

`stable_multiway_merge`

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

### Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
return `__value` - `__target` = min(`__length`, number of elements in all sequences).

### Parameters

*`__RAIterPairIterator`* iterator over sequence of pairs of iterators

*`__RAIterOut`* iterator over target sequence

*`__DifferenceTp`* difference type for the sequence

*`__Compare`* strict weak ordering type to compare elements in sequences

*`__seqs_begin`* `__begin` of sequence `__sequence`

*`__seqs_end`* `__M_end` of sequence `__sequence`

*`__target`* target sequence to merge to.

*`__comp`* strict weak ordering to use for element comparison.

*`__length`* Maximum length to merge, possibly larger than the number of elements available.

### Returns

`__M_end` iterator of output sequence

Definition at line 1410 of file multiway\_merge.h.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**4.6.4.69** `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_3_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp )`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 234 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.70** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_4_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 353 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**4.6.4.71** `template<bool __stable, typename _RAIterIterator ,  
typename _Compare , typename _DifferenceType > void  
__gnu_parallel::multiway_merge_exact_splitting ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _DifferenceType  
__length, _DifferenceType __total_length, _Compare __comp,  
std::vector< std::pair< _DifferenceType, _DifferenceType > > *  
__pieces )`

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1112 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equally_split()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.72** `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp )`

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 484 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

```

4.6.4.73 template<typename UnguardedLoserTree , typename _RAIterIterator
, typename _RAIter3 , typename _DifferenceTp , typename _Compare
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
  _RAIter3 __target, const typename std::iterator_traits< typename
std::iterator_traits< _RAIterIterator >::value_type::first_type
>::value_type & __sentinel, _DifferenceTp __length, _Compare
__comp )

```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

#### Parameters

- `__stable` The value must be the same as for the used LoserTrees.
- `UnguardedLoserTree` \_Loser Tree variant to use for the unguarded merging.
- `GuardedLoserTree` \_Loser Tree variant to use for the guarded merging.
- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 658 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

```

4.6.4.74 template<typename _LT , typename _RAIterIterator , typename
_RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
  _RAIter3 __target, const typename std::iterator_traits< typename
std::iterator_traits< _RAIterIterator >::value_type::first_type
>::value_type & __sentinel, _DifferenceTp __length, _Compare
__comp )

```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

*\_\_seqs\_begin* Begin iterator of iterator pair input sequence.

*\_\_seqs\_end* End iterator of iterator pair input sequence.

*\_\_target* Begin iterator of output sequence.

*\_\_comp* Comparator.

*\_\_length* Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 567 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.75** `template<bool __stable, typename _RAIterIterator ,  
typename _Compare , typename _DifferenceType > void  
__gnu_parallel::multiway_merge_sampling_splitting (   
_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_DifferenceType __length, _DifferenceType __total_length,  
_Compare __comp, std::vector< std::pair< _DifferenceType,  
_DifferenceType > > * __pieces )`

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1032 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`,  
and `__gnu_parallel::_Settings::merge_oversampling`.

**4.6.4.76** `template<typename _RAIterPairIterator , typename _RAIterOut ,  
typename _DifferenceTp , typename _Compare > _RAIterOut  
__gnu_parallel::multiway_merge_sentinels ( _RAIterPairIterator  
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
__target, _DifferenceTp __length, _Compare __comp,  
__gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_`-  
`begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an

iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 11; ++__j)
        sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                    sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

**See also**

`stable_multiway_merge_sentinels`

**Parameters**

*\_RAIterPairIterator* iterator over sequence of pairs of iterators

*\_RAIterOut* iterator over target sequence

*\_DifferenceTp* difference type for the sequence

*\_Compare* strict weak ordering type to compare elements in sequences

*\_\_seqs\_begin* \_\_begin of sequence \_\_sequence

*\_\_seqs\_end* \_\_M\_end of sequence \_\_sequence

*\_\_target* target sequence to merge to.

*\_\_comp* strict weak ordering to use for element comparison.

*\_\_length* Maximum length to merge, possibly larger than the number of elements available.

**Returns**

*\_\_M\_end* iterator of output sequence

Definition at line 1774 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

```
4.6.4.77 template<bool __stable, bool __sentinels, typename
    _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
    typename _Splitter, typename _Compare > _RAIter3
    __gnu_parallel::parallel_multiway_merge ( _RAIterIterator
    __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,
    _Splitter __splitter, _DifferenceTp __length, _Compare __comp,
    _ThreadIndex __num_threads )
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

**Parameters**

*\_Splitter* functor to split input (either `__exact` or `sampling` based)

*\_\_seqs\_begin* Begin iterator of iterator pair input sequence.

*\_\_seqs\_end* End iterator of iterator pair input sequence.

- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, possibly larger than the number of elements available.
- `__stable` Stable merging incurs a performance penalty.
- `__sentinel` Ignored.

**Returns**

End iterator of output sequence.

Definition at line 1217 of file `multiway_merge.h`.

References `__is_sorted()`, `_GLIBCXX_CALL`, `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.78** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

**Parameters**

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__comp` Comparator.
- `__n` Length of sequence.
- `__num_threads` Number of threads to use.

Definition at line 394 of file `multiway_mergesort.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_threads`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_offsets`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_samples`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::_Settings::get()`, `std::vector< _Tp, _Alloc >::resize()`, and `__gnu_parallel::_Settings::sort_mwms_oversampling`.

**4.6.4.79** `template<bool __stable, bool __exact, typename _RAIter , typename  
_Compare > void __gnu_parallel::parallel_sort_mwms_pu (  
_PMWMSortingData< _RAIter > * __sd, _Compare & __comp )`

PMWMS code executed by each thread.

#### Parameters

`__sd` Pointer to algorithm data.

`__comp` Comparator.

Definition at line 308 of file `multiway_mergesort.h`.

References `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_threads`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

## 4.6.5 Variable Documentation

**4.6.5.1** `const int __gnu_parallel::_CASable_bits [static]`

Number of bits of `_CASable`.

Definition at line 130 of file `types.h`.

Referenced by `__decode2()`, and `__encode2()`.

**4.6.5.2** `const _CASable __gnu_parallel::_CASable_mask [static]`

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file `types.h`.

Referenced by `__decode2()`.

## 4.7 \_\_gnu\_pbds Namespace Reference

GNU extensions for policy-based data structures for public use.

### Classes

- struct [associative\\_container\\_tag](#)

*Basic associative-container.*

- class `basic_hash_table`  
*An abstract basic hash-based associative container.*
- struct `basic_hash_tag`  
*Basic hash.*
- class `basic_tree`  
*An abstract basic tree-like (tree, trie) associative container.*
- struct `basic_tree_tag`  
*Basic tree.*
- struct `binary_heap_tag`  
*Binary-heap (array-based).*
- struct `binomial_heap_tag`  
*Binomial-heap.*
- class `cc_hash_table`  
*A concrete collision-chaining hash-based associative container.*
- struct `cc_hash_tag`  
*Collision-chaining hash.*
- class `container_base`  
*An abstract basic associative container.*
- struct `container_tag`  
*Base data structure tag.*
- struct `container_traits`  
*container\_traits*
- class `gp_hash_table`  
*A concrete general-probing hash-based associative container.*
- struct `gp_hash_tag`  
*General-probing hash.*
- class `list_update`

---

*A list-update based associative container.*

- struct [list\\_update\\_tag](#)  
*List-update.*
- struct [null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [sequence\\_tag](#)  
*Basic sequence.*
- struct [splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [string\\_tag](#)  
*Basic string container, inclusive of strings, ropes, etc.*
- struct [thin\\_heap\\_tag](#)  
*Thin heap.*
- class [tree](#)  
*A concrete basic tree-based associative container.*
- struct [tree\\_tag](#)

*tree.*

- class `trie`  
*A concrete basic trie-based associative container.*
- struct `trie_tag`  
*trie.*

## Typedefs

- typedef void `trivial_iterator_difference_type`

## Functions

- void `__throw_container_error` (void)
- void `__throw_insert_error` (void)
- void `__throw_join_error` (void)
- void `__throw_resize_error` (void)

### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_profile` Namespace Reference

GNU profile code for public use.

## Classes

- class `__container_size_info`  
*A container size instrumentation line in the object table.*
- class `__container_size_stack_info`  
*A container size instrumentation line in the stack table.*
- class `__hashfunc_info`  
*A hash performance instrumentation line in the object table.*
- class `__hashfunc_stack_info`

*A hash performance instrumentation line in the stack table.*

- class [\\_\\_list2vector\\_info](#)  
*A list-to-vector instrumentation line in the object table.*
- class [\\_\\_map2umap\\_info](#)  
*A map-to-unordered\_map instrumentation line in the object table.*
- class [\\_\\_map2umap\\_stack\\_info](#)  
*A map-to-unordered\_map instrumentation line in the stack table.*
- class [\\_\\_object\\_info\\_base](#)  
*Base class for a line in the object table.*
- struct [\\_\\_reentrance\\_guard](#)  
*Reentrance guard.*
- class [\\_\\_stack\\_hash](#)  
*Hash function for summary trace using call stack as index.*
- class [\\_\\_stack\\_info\\_base](#)  
*Base class for a line in the stack table.*
- class [\\_\\_trace\\_base](#)  
*Base class for all trace producers.*
- class [\\_\\_trace\\_container\\_size](#)  
*Container size instrumentation trace producer.*
- class [\\_\\_trace\\_hash\\_func](#)  
*Hash performance instrumentation producer.*
- class [\\_\\_trace\\_hashtable\\_size](#)  
*Hashtable size instrumentation trace producer.*
- class [\\_\\_trace\\_map2umap](#)  
*Map-to-unordered\_map instrumentation producer.*
- class [\\_\\_trace\\_vector\\_size](#)  
*Hashtable size instrumentation trace producer.*
- class [\\_\\_trace\\_vector\\_to\\_list](#)

*Vector-to-list instrumentation producer.*

- class `__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__vector2list_stack_info`  
*A vector-to-list instrumentation line in the stack table.*
- struct `__warning_data`  
*Representation of a warning.*

## Typedefs

- typedef `std::vector< __cost_factor * >` `__cost_factor_vector`
- typedef `std::unordered_map< std::string, std::string >` `__env_t`
- typedef `void *` `__instruction_address_t`
- typedef `const void *` `__object_t`
- typedef `std::vector< __instruction_address_t >` `__stack_npt`
- typedef `__stack_npt *` `__stack_t`
- typedef `std::vector< __warning_data >` `__warning_vector_t`

## Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

## Functions

- `size_t __env_to_size_t` (`const char * __env_var`, `size_t __default_value`)
- `__cost_factor_vector *& __get__cost_factors` ()
- `__env_t & __get__env` ()
- `__gnu_cxx::__mutex & __get__global_lock` ()
- `__cost_factor & __get__list_iterate_cost_factor` ()
- `__cost_factor & __get__list_resize_cost_factor` ()
- `__cost_factor & __get__list_shift_cost_factor` ()
- `__cost_factor & __get__map_erase_cost_factor` ()
- `__cost_factor & __get__map_find_cost_factor` ()
- `__cost_factor & __get__map_insert_cost_factor` ()
- `__cost_factor & __get__map_iterate_cost_factor` ()
- `__cost_factor & __get__umap_erase_cost_factor` ()
- `__cost_factor & __get__umap_find_cost_factor` ()

- `__cost_factor & __get__umap_insert_cost_factor ()`
- `__cost_factor & __get__umap_iterate_cost_factor ()`
- `__cost_factor & __get__vector_iterate_cost_factor ()`
- `__cost_factor & __get__vector_resize_cost_factor ()`
- `__cost_factor & __get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __get__S_hash_func ()`
- `__trace_hashtable_size *& __get__S_hashtable_size ()`
- `__trace_list_to_slist *& __get__S_list_to_slist ()`
- `__trace_list_to_vector *& __get__S_list_to_vector ()`
- `__trace_map2umap *& __get__S_map2umap ()`
- `size_t & __get__S_max_mem ()`
- `size_t & __get__S_max_stack_depth ()`
- `size_t & __get__S_max_warn_count ()`
- `const char *& __get__S_trace_file_name ()`
- `__trace_vector_size *& __get__S_vector_size ()`
- `__trace_vector_to_list *& __get__S_vector_to_list ()`
- `__stack_t __get_stack ()`
- `bool __is_invalid ()`
- `bool __is_off ()`
- `bool __is_on ()`
- `int __log2 (size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (size_t __size)`
- `float __map_find_cost (size_t __size)`
- `float __map_insert_cost (size_t __size)`
- `size_t __max_mem ()`
- `FILE * __open_output_file (const char *__extension)`
- `bool __profctx_init (void)`
- `void __profctx_init_unconditional ()`
- `void __read_cost_factors ()`
- `void __report (void)`
- `void __set_cost_factors ()`
- `void __set_max_mem ()`
- `void __set_max_stack_trace_depth ()`
- `void __set_max_warn_count ()`
- `void __set_trace_path ()`
- `__size (const __stack_t &__stack)`
- `size_t __stack_max_depth ()`
- `void __trace_hash_func_construct (const void *)`
- `void __trace_hash_func_destruct (const void *, size_t, size_t, size_t)`
- `void __trace_hash_func_init ()`
- `void __trace_hash_func_report (FILE *_f, __warning_vector_t &__warnings)`

- void `__trace_hashtable_size_construct` (const void \*, size\_t)
- void `__trace_hashtable_size_destruct` (const void \*, size\_t, size\_t)
- void `__trace_hashtable_size_init` ()
- void `__trace_hashtable_size_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__trace_hashtable_size_resize` (const void \*, size\_t, size\_t)
- void `__trace_list_to_set_construct` (const void \*)
- void `__trace_list_to_set_destruct` (const void \*)
- void `__trace_list_to_set_find` (const void \*, size\_t)
- void `__trace_list_to_set_insert` (const void \*, size\_t, size\_t)
- void `__trace_list_to_set_invalid_operator` (const void \*)
- void `__trace_list_to_set_iterate` (const void \*, size\_t)
- void `__trace_list_to_slist_construct` (const void \*)
- void `__trace_list_to_slist_destruct` (const void \*)
- void `__trace_list_to_slist_init` ()
- void `__trace_list_to_slist_operation` (const void \*)
- void `__trace_list_to_slist_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__trace_list_to_slist_rewind` (const void \*)
- void `__trace_list_to_vector_construct` (const void \*)
- void `__trace_list_to_vector_destruct` (const void \*)
- void `__trace_list_to_vector_init` ()
- void `__trace_list_to_vector_insert` (const void \*, size\_t, size\_t)
- void `__trace_list_to_vector_invalid_operator` (const void \*)
- void `__trace_list_to_vector_iterate` (const void \*, size\_t)
- void `__trace_list_to_vector_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__trace_list_to_vector_resize` (const void \*, size\_t, size\_t)
- void `__trace_map_to_unordered_map_construct` (const void \*)
- void `__trace_map_to_unordered_map_destruct` (const void \*)
- void `__trace_map_to_unordered_map_erase` (const void \*, size\_t, size\_t)
- void `__trace_map_to_unordered_map_find` (const void \*, size\_t)
- void `__trace_map_to_unordered_map_init` ()
- void `__trace_map_to_unordered_map_insert` (const void \*, size\_t, size\_t)
- void `__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__trace_map_to_unordered_map_iterate` (const void \*, size\_t)
- void `__trace_map_to_unordered_map_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__trace_vector_size_construct` (const void \*, size\_t)
- void `__trace_vector_size_destruct` (const void \*, size\_t, size\_t)
- void `__trace_vector_size_init` ()
- void `__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__trace_vector_size_resize` (const void \*, size\_t, size\_t)

- void **\_\_trace\_vector\_to\_list\_construct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_destruct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_find** (const void \*, size\_t)
- void **\_\_trace\_vector\_to\_list\_init** ()
- void **\_\_trace\_vector\_to\_list\_insert** (const void \*, size\_t, size\_t)
- void **\_\_trace\_vector\_to\_list\_invalid\_operator** (const void \*)
- void **\_\_trace\_vector\_to\_list\_iterate** (const void \*, size\_t)
- void **\_\_trace\_vector\_to\_list\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_trace\_vector\_to\_list\_resize** (const void \*, size\_t, size\_t)
- bool **\_\_turn** (\_\_state\_type \_\_s)
- bool **\_\_turn\_off** ()
- bool **\_\_turn\_on** ()
- void **\_\_write** (FILE \* \_\_f, const \_\_stack\_t \_\_stack)
- void **\_\_write\_cost\_factors** ()
- **\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_state\_type, \_\_state, \_\_INVALID)

### 4.8.1 Detailed Description

GNU profile code for public use.

### 4.8.2 Typedef Documentation

#### 4.8.2.1 typedef std:: std::unordered\_map<std::string, std::string> \_\_gnu\_profile::\_\_env\_t

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdcxx-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: export \_\_map\_iterate\_cost\_factor=5.0. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 75 of file profiler\_trace.h.

### 4.8.3 Function Documentation

#### 4.8.3.1 \_\_gnu\_cxx::\_\_mutex& \_\_gnu\_profile::\_\_get\_\_global\_lock ( ) [inline]

Master lock.

Definition at line 79 of file profiler\_trace.h.

#### 4.8.3.2 `bool __gnu_profile::__profctx_init ( void ) [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 617 of file `profiler_trace.h`.

#### 4.8.3.3 `void __gnu_profile::__report ( void ) [inline]`

Final report method, registered with `atexit`.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 398 of file `profiler_trace.h`.

References `std::min()`.

## 4.9 `__gnu_sequential` Namespace Reference

GNU sequential classes for public use.

### 4.9.1 Detailed Description

GNU sequential classes for public use.

## 4.10 `abi` Namespace Reference

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

### 4.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`. A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.11 std Namespace Reference

ISO C++ entities toplevel namespace is std.

### Namespaces

- namespace [\\_\\_debug](#)
- namespace [\\_\\_detail](#)
- namespace [\\_\\_parallel](#)
- namespace [\\_\\_profile](#)
- namespace [chrono](#)
- namespace [decimal](#)
- namespace [placeholders](#)
- namespace [rel\\_ops](#)
- namespace [this\\_thread](#)
- namespace [tr1](#)

### Classes

- class [\\_\\_basic\\_future](#)  
*Common implementation for future and shared\_future.*
- class [\\_\\_codecvt\\_abstract\\_base](#)  
*Common base for codecvt functions.*
- class [\\_\\_ctype\\_abstract\\_base](#)  
*Common base for ctype facet.*
- struct [\\_\\_declval\\_protector](#)  
*declval*

- struct [\\_\\_future\\_base](#)  
*Base class and enclosing scope.*
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_numeric\\_limits\\_base](#)  
*Part of `std::numeric_limits`.*
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- struct [\\_Build\\_index\\_tuple](#)  
*Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.*
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)  
*A `deque::iterator`.*
- struct [\\_Derives\\_from\\_binary\\_function](#)  
*Determines if the type `_Tp` derives from `binary_function`.*
- struct [\\_Derives\\_from\\_unary\\_function](#)  
*Determines if the type `_Tp` derives from `unary_function`.*
- class [\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [\\_Function\\_to\\_function\\_pointer](#)  
*Turns a function type into a function pointer type.*
- struct [\\_Fwd\\_list\\_base](#)  
*Base class for `forward_list`.*
- struct [\\_Fwd\\_list\\_const\\_iterator](#)  
*A `forward_list::const_iterator`.*
- struct [\\_Fwd\\_list\\_iterator](#)  
*A `forward_list::iterator`.*
- struct [\\_Fwd\\_list\\_node](#)  
*A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.*
- struct [\\_Fwd\\_list\\_node\\_base](#)

*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*

- class [\\_Has\\_result\\_type\\_helper](#)
- struct [\\_Index\\_tuple](#)
- class [\\_List\\_base](#)  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- struct [\\_List\\_const\\_iterator](#)  
*A `list::const_iterator`.*
- struct [\\_List\\_iterator](#)  
*A `list::iterator`.*
- struct [\\_List\\_node](#)  
*An actual node in the list.*
- struct [\\_List\\_node\\_base](#)  
*Common part of a node in the list.*
- struct [\\_Maybe\\_get\\_result\\_type](#)  
*If we have found a `result_type`, extract it.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1 >](#)  
*Derives from `unary_function`, as appropriate.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1, \\_T2 >](#)  
*Derives from `binary_function`, as appropriate.*
- struct [\\_Maybe\\_wrap\\_member\\_pointer](#)
- struct [\\_Maybe\\_wrap\\_member\\_pointer<\\_Tp \\_Class::\\* >](#)
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of `mem_fn` for const member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Implementation of `mem_fn` for volatile member function pointers.*
- class [\\_Mem\\_fn<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) >](#)  
*Implementation of `mem_fn` for member function pointers.*

- class [\\_Mu< \\_Arg, false, false >](#)
- class [\\_Mu< \\_Arg, false, true >](#)
- class [\\_Mu< \\_Arg, true, false >](#)
- class [\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- struct [\\_Placeholder](#)

*The type of placeholder objects defined by libstdc++.*

- struct [\\_Reference\\_wrapper\\_base](#)
- struct [\\_Safe\\_tuple\\_element](#)
- struct [\\_Safe\\_tuple\\_element\\_impl](#)
- struct [\\_Safe\\_tuple\\_element\\_impl< \\_\\_i, \\_Tuple, false >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Tuple\\_impl< \\_Idx >](#)
- struct [\\_Tuple\\_impl< \\_Idx, \\_Head, \\_Tail...>](#)
- struct [\\_Vector\\_base](#)

*See [bits/stl\\_deque.h](#)'s [\\_Deque\\_base](#) for an explanation.*

- struct [\\_Weak\\_result\\_type](#)
- struct [\\_Weak\\_result\\_type\\_impl](#)
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(&\)\(\\_ArgTypes...\)>](#)

*Retrieve the result type for a function reference.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\*\)\(\\_ArgTypes...\)>](#)

*Retrieve the result type for a function pointer.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_ArgTypes...\)>](#)

*Retrieve the result type for a function type.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)

*Retrieve result type for a const member function pointer.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)

*Retrieve result type for a const volatile member function pointer.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)

*Retrieve result type for a volatile member function pointer.*

- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)

*Retrieve result type for a member function pointer.*

- struct [add\\_lvalue\\_reference](#)

*add\_lvalue\_reference*

- struct `add_rvalue_reference`  
*add\_rvalue\_reference*
- struct `adopt_lock_t`  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct `aligned_storage`  
*Alignment type.*
- class `allocator`  
*The standard allocator, as per [20.4].  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*
- class `allocator< void >`  
*allocator<void> specialization.*
- struct `atomic`  
*atomic /// 29.4.3, Generic atomic type, primary class template.*
- struct `atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `atomic< bool >`  
*Explicit specialization for bool.*
- struct `atomic< char >`  
*Explicit specialization for char.*
- struct `atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `atomic< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `atomic< int >`  
*Explicit specialization for int.*
- struct `atomic< long >`  
*Explicit specialization for long.*
- struct `atomic< long long >`

*Explicit specialization for long long.*

- struct `atomic< short >`  
*Explicit specialization for short.*
- struct `atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `atomic< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `atomic< void * >`  
*Explicit specialization for void\*.*
- struct `atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*
- class `auto_ptr`  
*A simple smart pointer providing strict ownership semantics.*
- struct `auto_ptr_ref`
- class `back_insert_iterator`  
*Turns assignment into insertion.*
- class `bad_alloc`  
*Exception possibly thrown by new.  
bad\_alloc (or classes derived from it) is used to report allocation errors from the throwing forms of new.*
- class `bad_cast`

*Thrown during incorrect typecasting.*

*If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*

- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)

*Exception class thrown when class template function's `operator()` is called with an empty target.*

- class [bad\\_typeid](#)

*Thrown when a `NULL` pointer in a `typeid` expression is used.*

- class [basic\\_filebuf](#)

*The actual work of input and output (for files).*

*This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.*

- class [basic\\_fstream](#)

*Controlling input and output for files.*

*This class supports reading from and writing to named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class [basic\\_ifstream](#)

*Controlling input for files.*

*This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class [basic\\_ios](#)

*Virtual base class for all stream classes.*

*Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*

- class [basic\\_iostream](#)

*Merging `istream` and `ostream` capabilities.*

*This class multiply inherits from the input and output stream classes simply to provide a single interface.*

- class [basic\\_istream](#)

*Controlling input.*

*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*

- class `basic_istream`

Controlling input for `std::string`.  
This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.
- class `basic_ofstream`

Controlling output for files.  
This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.
- class `basic_ostream`

Controlling output.  
This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.
- class `basic_ostringstream`

Controlling output for `std::string`.  
This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.
- class `basic_streambuf`

The actual work of input and output (interface).  
This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.
- class `basic_string`

Managing sequences of characters and character-like objects.
- class `basic_stringbuf`

The actual work of input and output (for `std::string`).  
This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).
- class `basic_stringstream`

Controlling input and output for `std::string`.  
This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_iostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class [bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct [binary\\_function](#)
- class [binary\\_negate](#)  
*One of the negation functors.*
- class [binder1st](#)  
*One of the binder functors.*
- class [binder2nd](#)  
*One of the binder functors.*
- class [binomial\\_distribution](#)  
*A discrete binomial random number distribution.*
- class [bitset](#)  
*The bitset class represents a fixed-size sequence of bits.*
- class [cauchy\\_distribution](#)  
*A cauchy\_distribution random number distribution.*
- struct [char\\_traits](#)  
*Basis for explicit traits specializations.*
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)  
*char\_traits<\_\_gnu\_cxx::character> specialization.*
- struct [char\\_traits< char >](#)  
*21.1.3.1 char\_traits specializations*
- struct [char\\_traits< wchar\\_t >](#)  
*21.1.3.2 char\_traits specializations*
- class [chi\\_squared\\_distribution](#)  
*A chi\_squared\_distribution random number distribution.*
- class [codecvt](#)  
*Primary class template codecvt.  
NB: Generic, mostly useless implementation.*

- class `codecvt<_InternT, _ExternT, encoding_state >`  
*codecvt<InternT, \_ExternT, encoding\_state> specialization.*
- class `codecvt<char, char, mbstate_t >`  
*class codecvt<char, char, mbstate\_t> specialization.*
- class `codecvt<wchar_t, char, mbstate_t >`  
*class codecvt<wchar\_t, char, mbstate\_t> specialization.*
- class `codecvt_base`  
*Empty base class for codecvt facet [22.2.1.5].*
- class `codecvt_byname`  
*class codecvt\_byname [22.2.1.6].*
- class `collate`  
*Facet for localized string comparison.*
- class `collate_byname`  
*class collate\_byname [22.2.4.2].*
- struct `complex`
- class `condition_variable`  
*condition\_variable*
- class `condition_variable_any`  
*condition\_variable\_any*
- struct `conditional`  
*conditional*
- class `const_mem_fun1_ref_t`  
*One of the adaptors for member /// pointers.*
- class `const_mem_fun1_t`  
*One of the adaptors for member /// pointers.*
- class `const_mem_fun_ref_t`  
*One of the adaptors for member /// pointers.*
- class `const_mem_fun_t`

One of the *adaptors for member /// pointers*.

- class [ctype](#)  
*Primary class template ctype facet.  
This template class defines classification and conversion functions for character sets.  
It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class [ctype< char >](#)  
*The ctype<char> specialization.  
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class [ctype< wchar\\_t >](#)  
*The ctype<wchar\_t> specialization.  
This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- struct [ctype\\_base](#)  
*Base class for ctype.*
- class [ctype\\_byname](#)  
*class ctype\_byname [22.2.1.2].*
- class [ctype\\_byname< char >](#)  
*22.2.1.4 Class ctype\_byname specializations.*
- class [decay](#)  
*decay*
- struct [default\\_delete](#)  
*Primary template, default\_delete.*
- struct [default\\_delete< \\_Tp\[ \]>](#)  
*Specialization, default\_delete.*
- struct [defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [deque](#)  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*

- class [discard\\_block\\_engine](#)
- class [discrete\\_distribution](#)  
*A [discrete\\_distribution](#) random number distribution.*
- struct [divides](#)  
*One of the [math](#) functors.*
- class [domain\\_error](#)
- struct [enable\\_if](#)  
*[enable\\_if](#)*
- class [enable\\_shared\\_from\\_this](#)  
*Base class allowing use of member function [shared\\_from\\_this](#).*
- struct [equal\\_to](#)  
*One of the [comparison](#) functors.*
- class [error\\_category](#)  
*[error\\_category](#)*
- struct [error\\_code](#)  
*[error\\_code](#)*
- struct [error\\_condition](#)  
*[error\\_condition](#)*
- class [exception](#)  
*Base class for all library exceptions.*
- class [exponential\\_distribution](#)  
*An exponential continuous distribution for random numbers.*
- class [extreme\\_value\\_distribution](#)  
*A [extreme\\_value\\_distribution](#) random number distribution.*
- class [fisher\\_f\\_distribution](#)  
*A [fisher\\_f\\_distribution](#) random number distribution.*
- struct [forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- class [forward\\_list](#)

---

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class [fpos](#)  
*Class representing stream positions.*
- class [front\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [function< \\_Res\(\\_ArgTypes...\)>](#)  
*Primary class template for `std::function`.  
Polymorphic function wrapper.*
- class [future](#)  
*Primary template for future.*
- class [future< \\_Res & >](#)  
*Partial specialization for `future<R&>`*
- class [future< void >](#)  
*Explicit specialization for `future<void>`*
- class [future\\_error](#)  
*Exception type thrown by futures.*
- class [gamma\\_distribution](#)  
*A gamma continuous distribution for random numbers.*
- class [geometric\\_distribution](#)  
*A discrete geometric random number distribution.*
- struct [greater](#)  
*One of the *comparison functors*.*
- struct [greater\\_equal](#)  
*One of the *comparison functors*.*
- class [gslice](#)  
*Class defining multi-dimensional subset of an array.*
- class [gslice\\_array](#)  
*Reference to multi-dimensional subset of an array.*

- struct `has_nothrow_assign`  
*has\_nothrow\_assign*
- struct `has_nothrow_copy_constructor`  
*has\_nothrow\_copy\_constructor*
- struct `has_nothrow_default_constructor`  
*has\_nothrow\_default\_constructor*
- struct `has_trivial_assign`  
*has\_trivial\_assign*
- struct `has_trivial_copy_constructor`  
*has\_trivial\_copy\_constructor*
- struct `has_trivial_default_constructor`  
*has\_trivial\_default\_constructor*
- struct `has_trivial_destructor`  
*has\_trivial\_destructor*
- struct `hash`  
*Primary class template hash.*
- struct `hash< __debug::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash< __debug::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `hash< __gnu_cxx::throw_value_limit >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __gnu_cxx::throw_value_random >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __profile::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash< __profile::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*

- struct `hash< _Tp * >`  
*Partial specializations for pointer types.*
- struct `hash< error_code >`  
*std::hash specialization for error\_code.*
- struct `hash< string >`  
*std::hash specialization for string.*
- struct `hash< thread::id >`  
*std::hash specialization for thread::id.*
- struct `hash< u16string >`  
*std::hash specialization for u16string.*
- struct `hash< u32string >`  
*std::hash specialization for u32string.*
- struct `hash< wstring >`  
*std::hash specialization for wstring.*
- struct `hash<::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash<::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `identity`  
*identity*
- class `independent_bits_engine`
- class `indirect_array`  
*Reference to arbitrary subset of an array.*
- class `initializer_list`  
*initializer\_list*
- struct `input_iterator_tag`  
*Marking input iterators.*
- class `insert_iterator`  
*Turns assignment into insertion.*

- class [invalid\\_argument](#)
- class [ios\\_base](#)

*The base of the I/O class hierarchy.*  
*This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios\\_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- struct [is\\_base\\_of](#)

*is\_base\_of*
- struct [is\\_bind\\_expression](#)

*Determines if the given type `_Tp` is a function object should be treated as a sub-expression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].*
- struct [is\\_bind\\_expression<\\_Bind<\\_Signature>>](#)

*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression<\\_Bind\\_result<\\_Result, \\_Signature>>](#)

*Class template `_Bind` is always a bind expression.*
- struct [is\\_constructible](#)

*is\_constructible*
- struct [is\\_convertible](#)

*is\_convertible*
- struct [is\\_error\\_code\\_enum](#)

*is\_error\_code\_enum*
- struct [is\\_error\\_condition\\_enum](#)

*is\_error\_condition\_enum*
- struct [is\\_explicitly\\_convertible](#)

*is\_explicitly\_convertible*
- struct [is\\_lvalue\\_reference](#)

*is\_lvalue\_reference*
- struct [is\\_placeholder](#)

*Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].*

- struct `is_placeholder< _Placeholder< _Num > >`
- struct `is_pod`  
*is\_pod*
- struct `is_reference`  
*is\_reference*
- struct `is_rvalue_reference`  
*is\_rvalue\_reference*
- struct `is_signed`  
*is\_signed*
- struct `is_standard_layout`  
*is\_standard\_layout*
- struct `is_trivial`  
*is\_trivial*
- struct `is_unsigned`  
*is\_unsigned*
- class `istream_iterator`  
*Provides input iterator semantics for streams.*
- class `istreambuf_iterator`  
*Provides input iterator semantics for streambufs.*
- struct `iterator`  
*Common iterator class.*
- struct `iterator_traits`  
*Traits class for iterators.*
- struct `iterator_traits< _Tp * >`  
*Partial specialization for pointer types.*
- struct `iterator_traits< const _Tp * >`  
*Partial specialization for const pointer types.*
- class `length_error`
- struct `less`

One of the *comparison functors*.

- struct `less_equal`  
One of the *comparison functors*.
- class `linear_congruential_engine`  
A model of a linear congruential random number generator.
- class `list`  
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `locale`  
Container class for localization functionality.  
The `locale` class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.
- class `lock_guard`  
Scoped lock idiom.
- class `logic_error`  
One of two subclasses of exception.
- struct `logical_and`  
One of the *Boolean operations functors*.
- struct `logical_not`  
One of the *Boolean operations functors*.
- struct `logical_or`  
One of the *Boolean operations functors*.
- class `lognormal_distribution`  
A *lognormal\_distribution* random number distribution.
- struct `make_signed`  
*make\_signed*
- struct `make_unsigned`  
*make\_unsigned*
- class `map`

---

*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

- class [mask\\_array](#)  
*Reference to selected subset of an array.*
- class [mem\\_fun1\\_ref\\_t](#)  
*One of the [adaptors for member /// pointers](#).*
- class [mem\\_fun1\\_t](#)  
*One of the [adaptors for member /// pointers](#).*
- class [mem\\_fun\\_ref\\_t](#)  
*One of the [adaptors for member /// pointers](#).*
- class [mem\\_fun\\_t](#)  
*One of the [adaptors for member /// pointers](#).*
- class [messages](#)  
*Primary class template messages.  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [messages\\_byname](#)  
*class [messages\\_byname](#) [22.2.7.2].*
- struct [minus](#)  
*One of the [math functors](#).*
- struct [modulus](#)  
*One of the [math functors](#).*
- class [money\\_base](#)  
*Money format ordering data.  
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*
- class [money\\_get](#)

Primary class template [money\\_get](#).

*This facet encapsulates the code to parse and return a monetary amount from a string.*

- class [money\\_put](#)

Primary class template [money\\_put](#).

*This facet encapsulates the code to format and output a monetary amount.*

- class [moneypunct](#)

Primary class template [moneypunct](#).

*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*

- class [moneypunct\\_byname](#)

*class [moneypunct\\_byname](#) [22.2.6.4].*

- class [move\\_iterator](#)

- class [multimap](#)

*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

- struct [multiplies](#)

*One of the [math functors](#).*

- class [multiset](#)

*A standard container made up of elements, which can be retrieved in logarithmic time.*

- class [mutex](#)

*mutex*

- struct [negate](#)

*One of the [math functors](#).*

- class [negative\\_binomial\\_distribution](#)

*A [negative\\_binomial\\_distribution](#) random number distribution.*

- class [nested\\_exception](#)

*Exception class with [exception\\_ptr](#) data member.*

- class [normal\\_distribution](#)

*A normal continuous distribution for random numbers.*

- struct [not\\_equal\\_to](#)

*One of the [comparison functors](#).*

- class [num\\_get](#)  
*Primary class template [num\\_get](#).  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class [num\\_put](#)  
*Primary class template [num\\_put](#).  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- struct [numeric\\_limits](#)  
*Properties of fundamental types.*
- struct [numeric\\_limits< bool >](#)  
*[numeric\\_limits<bool>](#) specialization.*
- struct [numeric\\_limits< char >](#)  
*[numeric\\_limits<char>](#) specialization.*
- struct [numeric\\_limits< char16\\_t >](#)  
*[numeric\\_limits<char16\\_t>](#) specialization.*
- struct [numeric\\_limits< char32\\_t >](#)  
*[numeric\\_limits<char32\\_t>](#) specialization.*
- struct [numeric\\_limits< double >](#)  
*[numeric\\_limits<double>](#) specialization.*
- struct [numeric\\_limits< float >](#)  
*[numeric\\_limits<float>](#) specialization.*
- struct [numeric\\_limits< int >](#)  
*[numeric\\_limits<int>](#) specialization.*
- struct [numeric\\_limits< long >](#)  
*[numeric\\_limits<long>](#) specialization.*
- struct [numeric\\_limits< long double >](#)  
*[numeric\\_limits<long double>](#) specialization.*
- struct [numeric\\_limits< long long >](#)  
*[numeric\\_limits<long long>](#) specialization.*

- struct `numeric_limits< short >`  
*numeric\_limits<short> specialization.*
- struct `numeric_limits< signed char >`  
*numeric\_limits<signed char> specialization.*
- struct `numeric_limits< unsigned char >`  
*numeric\_limits<unsigned char> specialization.*
- struct `numeric_limits< unsigned int >`  
*numeric\_limits<unsigned int> specialization.*
- struct `numeric_limits< unsigned long >`  
*numeric\_limits<unsigned long> specialization.*
- struct `numeric_limits< unsigned long long >`  
*numeric\_limits<unsigned long long> specialization.*
- struct `numeric_limits< unsigned short >`  
*numeric\_limits<unsigned short> specialization.*
- struct `numeric_limits< wchar_t >`  
*numeric\_limits<wchar\_t> specialization.*
- class `numpunct`  
*Primary class template numpunct.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class `numpunct_byname`  
*class numpunct\_byname [22.2.3.2].*
- struct `once_flag`  
*once\_flag*
- class `ostream_iterator`  
*Provides output iterator semantics for streams.*
- class `ostreambuf_iterator`  
*Provides output iterator semantics for streambufs.*

- class [out\\_of\\_range](#)
- struct [output\\_iterator\\_tag](#)  
*Marking output iterators.*
- class [overflow\\_error](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)  
*Partial specialization of [owner\\_less](#) for [shared\\_ptr](#).*
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)  
*Partial specialization of [owner\\_less](#) for [weak\\_ptr](#).*
- class [packaged\\_task< \\_Res\(\\_ArgTypes...\)>](#)  
*[packaged\\_task](#)*
- struct [pair](#)  
*pair holds two objects of arbitrary type.*
- class [piecewise\\_constant\\_distribution](#)  
*A [piecewise\\_constant\\_distribution](#) random number distribution.*
- class [piecewise\\_linear\\_distribution](#)  
*A [piecewise\\_linear\\_distribution](#) random number distribution.*
- struct [plus](#)  
*One of the [math functors](#).*
- class [pointer\\_to\\_binary\\_function](#)  
*One of the [adaptors for function pointers](#).*
- class [pointer\\_to\\_unary\\_function](#)  
*One of the [adaptors for function pointers](#).*
- class [poisson\\_distribution](#)  
*A discrete Poisson random number distribution.*
- class [priority\\_queue](#)  
*A standard container automatically sorting its contents.*
- class [promise](#)  
*Primary template for [promise](#).*

- class `promise<_Res &>`  
*Partial specialization for `promise<R&>`*
- class `promise<void>`  
*Explicit specialization for `promise<void>`*
- class `queue`  
*A standard container giving FIFO behavior.*
- struct `random_access_iterator_tag`  
*Random-access iterators support a superset of bidirectional /// iterator operations.*
- class `random_device`
- class `range_error`
- struct `ratio`  
*Provides compile-time rational arithmetic.*
- struct `ratio_add`  
*`ratio_add`*
- struct `ratio_divide`  
*`ratio_divide`*
- struct `ratio_equal`  
*`ratio_equal`*
- struct `ratio_greater`  
*`ratio_greater`*
- struct `ratio_greater_equal`  
*`ratio_greater_equal`*
- struct `ratio_less`  
*`ratio_less`*
- struct `ratio_less_equal`  
*`ratio_less_equal`*
- struct `ratio_multiply`  
*`ratio_multiply`*
- struct `ratio_not_equal`

*ratio\_not\_equal*

- struct `ratio_subtract`

*ratio\_subtract*

- class `raw_storage_iterator`

- class `recursive_mutex`

*recursive\_mutex*

- class `recursive_timed_mutex`

*recursive\_timed\_mutex*

- class `reference_wrapper`

*Primary class template for `reference_wrapper`.*

- struct `remove_reference`

*remove\_reference*

- class `reverse_iterator`

- class `runtime_error`

*One of two subclasses of exception.*

- class `seed_seq`

*The `seed_seq` class generates sequences of seeds for random number generators.*

- class `set`

*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

- class `shared_future`

*Primary template for `shared_future`.*

- class `shared_future<_Res &>`

*Partial specialization for `shared_future<R&>`*

- class `shared_future<void>`

*Explicit specialization for `shared_future<void>`*

- class `shared_ptr`

*A smart pointer with reference-counted copy semantics.*

- class `shuffle_order_engine`

---

*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.*

- class `slice`  
*Class defining one-dimensional subset of an array.*
- class `slice_array`  
*Reference to one-dimensional subset of an array.*
- class `stack`  
*A standard container giving FILO behavior.*
- class `student_t_distribution`  
*A `student_t_distribution` random number distribution.*
- class `system_error`  
*Thrown to indicate error code of underlying system.*
- class `thread`  
*thread*
- class `time_base`  
*Time format ordering data.  
This class provides an enum representing different orderings of time: day, month, and year.*
- class `time_get`  
*Primary class template `time_get`.  
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*
- class `time_get_byname`  
*class `time_get_byname` [22.2.5.2].*
- class `time_put`  
*Primary class template `time_put`.  
This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*
- class `time_put_byname`  
*class `time_put_byname` [22.2.5.4].*
- class `timed_mutex`  
*timed\_mutex*

- struct `try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `tuple`  
*tuple*
- class `tuple< _T1, _T2 >`  
*tuple (2-element), with construction and assignment from a pair.*
- struct `tuple_element< 0, tuple< _Head, _Tail...> >`
- struct `tuple_element< __i, tuple< _Head, _Tail...> >`
- struct `tuple_size< tuple< _Elements...> >`  
*class tuple\_size*
- class `type_info`  
*Part of RTTI.*
- struct `unary_function`
- class `unary_negate`  
*One of the negation functors.*
- class `underflow_error`
- class `uniform_int_distribution`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class `uniform_real_distribution`  
*Uniform continuous distribution for random numbers.*
- class `unique_lock`  
*unique\_lock*
- class `unique_ptr`  
*20.7.12.2 unique\_ptr for single objects.*
- class `unique_ptr< _Tp[ ], _Tp_Deleter >`  
*20.7.12.3 unique\_ptr for array objects with a runtime length*
- class `unordered_map`  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*

- class `unordered_multimap`  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*
- class `unordered_multiset`  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*
- class `unordered_set`  
*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*
- class `valarray`  
*Smart array designed to support numeric processing.*
- class `vector`  
*A standard container which offers fixed time access to individual elements in any order.*
- class `vector< bool, _Alloc >`  
*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*
- class `weak_ptr`  
*A smart pointer with weak semantics.*
- class `weibull_distribution`  
*A weibull\_distribution random number distribution.*

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base **\_\_atomic\_flag\_base**
- typedef FILE **\_\_c\_file**
- typedef \_\_locale\_t **\_\_c\_locale**
- typedef \_\_gthread\_mutex\_t **\_\_c\_lock**
- typedef unsigned long **\_Bit\_type**
- typedef `atomic_short` **atomic\_int\_fast16\_t**
- typedef `atomic_int` **atomic\_int\_fast32\_t**
- typedef `atomic_llong` **atomic\_int\_fast64\_t**
- typedef `atomic_schar` **atomic\_int\_fast8\_t**
- typedef `atomic_short` **atomic\_int\_least16\_t**
- typedef `atomic_int` **atomic\_int\_least32\_t**

- typedef [atomic\\_llong](#) **atomic\_int\_least64\_t**
- typedef [atomic\\_schar](#) **atomic\_int\_least8\_t**
- typedef [atomic\\_llong](#) **atomic\_intmax\_t**
- typedef [atomic\\_long](#) **atomic\_intptr\_t**
- typedef [atomic\\_long](#) **atomic\_ptrdiff\_t**
- typedef [atomic\\_ulong](#) **atomic\_size\_t**
- typedef [atomic\\_long](#) **atomic\_ssize\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_fast16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_fast32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_fast64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_fast8\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_least16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_least32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_least64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_least8\_t**
- typedef [atomic\\_ullong](#) **atomic\_uintmax\_t**
- typedef [atomic\\_ulong](#) **atomic\_uintptr\_t**
- typedef [ratio](#)< 1, 1000000000000000000 > **atto**
- typedef [ratio](#)< 1, 100 > **centi**
- typedef [ratio](#)< 10, 1 > **deca**
- typedef [ratio](#)< 1, 10 > **deci**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- typedef [ratio](#)< 1000000000000000000, 1 > **exa**
- typedef [ratio](#)< 1, 1000000000000000 > **femto**
- typedef [basic\\_filebuf](#)< char > **filebuf**
- typedef [basic\\_fstream](#)< char > **fstream**
- typedef [ratio](#)< 1000000000, 1 > **giga**
- typedef [ratio](#)< 100, 1 > **hecto**
- typedef [basic\\_ifstream](#)< char > **ifstream**
- typedef [basic\\_ios](#)< char > **ios**
- typedef [basic\\_iostream](#)< char > **iostream**
- typedef [basic\\_istream](#)< char > **istream**
- typedef [ratio](#)< 1000, 1 > **kilo**
- typedef [shuffle\\_order\\_engine](#)< [minstd\\_rand0](#), 256 > **knuth\_b**
- typedef [ratio](#)< 1000000, 1 > **mega**
- typedef enum [std::memory\\_order](#) **memory\_order**
- typedef [ratio](#)< 1, 1000000 > **micro**
- typedef [ratio](#)< 1, 1000 > **milli**
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 48271UL, 0UL, 2147483647UL > **minstd\_rand**
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 16807UL, 0UL, 2147483647UL > **minstd\_rand0**

- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef [ratio](#)< 1, 1000000000 > **nano**
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > [ofstream](#)
- typedef [basic\\_ostream](#)< char > [ostream](#)
- typedef [basic\\_ostringstream](#)< char > [ostringstream](#)
- typedef [ratio](#)< 1000000000000000, 1 > **petta**
- typedef [ratio](#)< 1, 1000000000000 > **pico**
- typedef [discard\\_block\\_engine](#)< ranlux24\_base, 223, 23 > **ranlux24**
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **ranlux24\_base**
- typedef [discard\\_block\\_engine](#)< ranlux48\_base, 389, 11 > **ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **ranlux48\_base**
  
- typedef [basic\\_streambuf](#)< char > [streambuf](#)
- typedef long long [streamoff](#)
- typedef [fpos](#)< mbstate\_t > [streampos](#)
- typedef ptrdiff\_t [streamsize](#)
- typedef [basic\\_string](#)< char > [string](#)
- typedef [basic\\_stringbuf](#)< char > [stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [stringstream](#)
- typedef [ratio](#)< 1000000000000, 1 > **tera**
- typedef void(\* [terminate\\_handler](#) )()
- typedef [fpos](#)< mbstate\_t > [u16streampos](#)
- typedef [basic\\_string](#)< char16\_t > [u16string](#)
- typedef [fpos](#)< mbstate\_t > [u32streampos](#)
- typedef [basic\\_string](#)< char32\_t > [u32string](#)
- typedef void(\* [unexpected\\_handler](#) )()
- typedef [basic\\_filebuf](#)< wchar\_t > [wfilebuf](#)
- typedef [basic\\_fstream](#)< wchar\_t > [wfstream](#)
- typedef [basic\\_ifstream](#)< wchar\_t > [wifstream](#)
- typedef [basic\\_ios](#)< wchar\_t > [wios](#)
- typedef [basic\\_iostream](#)< wchar\_t > [wiostream](#)
- typedef [basic\\_istream](#)< wchar\_t > [wistream](#)
- typedef [basic\\_istringstream](#)< wchar\_t > [wistringstream](#)
- typedef [basic\\_ofstream](#)< wchar\_t > [wofstream](#)
- typedef [basic\\_ostream](#)< wchar\_t > [wostream](#)

- typedef [basic\\_ostringstream](#)< wchar\_t > [wostringstream](#)
- typedef [basic\\_streambuf](#)< wchar\_t > [wstreambuf](#)
- typedef [fpos](#)< mbstate\_t > [wstreampos](#)
- typedef [basic\\_string](#)< wchar\_t > [wstring](#)
- typedef [basic\\_stringbuf](#)< wchar\_t > [wstringbuf](#)
- typedef [basic\\_stringstream](#)< wchar\_t > [wstringstream](#)

## Enumerations

- enum { [\\_S\\_threshold](#) }
- enum { [\\_S\\_chunk\\_size](#) }
- enum { [\\_S\\_word\\_bit](#) }
- enum [\\_Ios\\_Fmtflags](#) {  
[\\_S\\_boolalpha](#), [\\_S\\_dec](#), [\\_S\\_fixed](#), [\\_S\\_hex](#),  
[\\_S\\_internal](#), [\\_S\\_left](#), [\\_S\\_oct](#), [\\_S\\_right](#),  
[\\_S\\_scientific](#), [\\_S\\_showbase](#), [\\_S\\_showpoint](#), [\\_S\\_showpos](#),  
[\\_S\\_skipws](#), [\\_S\\_unitbuf](#), [\\_S\\_uppercase](#), [\\_S\\_adjustfield](#),  
[\\_S\\_basefield](#), [\\_S\\_floatfield](#), [\\_S\\_ios\\_fmtflags\\_end](#) }
- enum [\\_Ios\\_Iostate](#) {  
[\\_S\\_goodbit](#), [\\_S\\_badbit](#), [\\_S\\_eofbit](#), [\\_S\\_failbit](#),  
[\\_S\\_ios\\_iostate\\_end](#) }
- enum [\\_Ios\\_Openmode](#) {  
[\\_S\\_app](#), [\\_S\\_ate](#), [\\_S\\_bin](#), [\\_S\\_in](#),  
[\\_S\\_out](#), [\\_S\\_trunc](#), [\\_S\\_ios\\_openmode\\_end](#) }
- enum [\\_Ios\\_Seekdir](#) { [\\_S\\_beg](#), [\\_S\\_cur](#), [\\_S\\_end](#), [\\_S\\_ios\\_seekdir\\_end](#) }
- enum [\\_Manager\\_operation](#) { [\\_\\_get\\_type\\_info](#), [\\_\\_get\\_functor\\_ptr](#), [\\_\\_clone\\_functor](#), [\\_\\_destroy\\_functor](#) }
- enum [\\_Rb\\_tree\\_color](#) { [\\_S\\_red](#), [\\_S\\_black](#) }
- enum [cv\\_status](#) { [no\\_timeout](#), [timeout](#) }
- enum [errc](#) {  
[address\\_family\\_not\\_supported](#), [address\\_in\\_use](#), [address\\_not\\_available](#),  
[already\\_connected](#),  
[argument\\_list\\_too\\_long](#), [argument\\_out\\_of\\_domain](#), [bad\\_address](#), [bad\\_file\\_descriptor](#),  
[bad\\_message](#), [broken\\_pipe](#), [connection\\_aborted](#), [connection\\_already\\_in\\_progress](#),  
[connection\\_refused](#), [connection\\_reset](#), [cross\\_device\\_link](#), [destination\\_address\\_required](#),  
[device\\_or\\_resource\\_busy](#), [directory\\_not\\_empty](#), [executable\\_format\\_error](#),  
[file\\_exists](#),

---

**file\_too\_large, filename\_too\_long, function\_not\_supported, host\_unreachable,**  
**identifier\_removed, illegal\_byte\_sequence, inappropriate\_io\_control\_operation, interrupted,**  
**invalid\_argument, invalid\_seek, io\_error, is\_a\_directory,**  
**message\_size, network\_down, network\_reset, network\_unreachable,**  
**no\_buffer\_space, no\_child\_process, no\_link, no\_lock\_available,**  
**no\_message\_available, no\_message, no\_protocol\_option, no\_space\_on\_device,**  
**no\_stream\_resources, no\_such\_device\_or\_address, no\_such\_device, no\_such\_file\_or\_directory,**  
**no\_such\_process, not\_a\_directory, not\_a\_socket, not\_a\_stream,**  
**not\_connected, not\_enough\_memory, not\_supported, operation\_canceled,**  
**operation\_in\_progress, operation\_not\_permitted, operation\_not\_supported, operation\_would\_block,**  
**owner\_dead, permission\_denied, protocol\_error, protocol\_not\_supported,**  
**read\_only\_file\_system, resource\_deadlock\_would\_occur, resource\_unavailable\_try\_again, result\_out\_of\_range,**  
**state\_not\_recoverable, stream\_timeout, text\_file\_busy, timed\_out,**  
**too\_many\_files\_open\_in\_system, too\_many\_files\_open, too\_many\_links, too\_many\_symbolic\_link\_levels,**  
**value\_too\_large, wrong\_protocol\_type }**

- enum [float\\_denorm\\_style](#) { [denorm\\_indeterminate](#), [denorm\\_absent](#), [denorm\\_present](#) }
- enum [float\\_round\\_style](#) {  
[round\\_indeterminate](#), [round\\_toward\\_zero](#), [round\\_to\\_nearest](#), [round\\_toward\\_infinity](#),  
[round\\_toward\\_neg\\_infinity](#) }
- enum [future\\_errc](#) { [broken\\_promise](#), [future\\_already\\_retrieved](#), [promise\\_already\\_satisfied](#), [no\\_state](#) }
- enum [launch](#) { [any](#), [async](#), [sync](#) }
- enum [memory\\_order](#) {  
[memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#),  
[memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

---

## Functions

- `template<typename _CharT >`  
`_CharT * __add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg,`  
`size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`  
`Compare >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _InputIterator, typename _Distance >`  
`void __advance (_InputIterator &__i, _Distance __n, input\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void __advance (_BidirectionalIterator &__i, _Distance __n, bidirectional\_`  
`iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __advance (_RandomAccessIterator &__i, _Distance __n, random\_`  
`access\_iterator\_tag)`
- `void __atomic_flag_wait_explicit (__atomic_flag_base *, memory\_order) \_`  
`GLIBCXX\_NOTHROW`
- `\_\_attribute__ ((__pure__)) \_Rb\_tree\_node\_base *\_Rb\_tree\_increment(\_Rb\_`  
`tree\_node\_base *__x) throw ()`
- `\_\_attribute__ ((__const__)) \_\_atomic\_flag\_base *\_\_atomic\_flag\_for\_`  
`address(const void *__z) \_GLIBCXX\_NOTHROW`
- `memory\_order __calculate_memory_order (memory\_order __m)`
- `template<typename _Member, typename _Class >`  
`\_Mem\_fn< _Member _Class::* > __callable_functor (_Member _Class::*&__`  
`__p)`
- `template<typename _Member, typename _Class >`  
`\_Mem\_fn< _Member _Class::* > __callable_functor (_Member \_`  
`Class::*const &__p)`
- `template<typename _Functor >`  
`\_Functor & __callable_functor (_Functor &__f)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, \_`  
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, \_`  
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`

- `template<typename _Tp >`  
`_Tp __cmath_power (_Tp, unsigned int)`
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)),`  
`char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, long double &, ios\_base::iostate &, const __c_locale &) throw ()`

- `template<bool _IsMove, typename _II, typename _OI >  
_OI copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT >::__type copy_move_a2 (_CharT *__first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT >::__type copy_move_a2 (const _CharT *__first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > >::__type copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > >::__type copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > > >)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > > >, istreambuf_iterator< _CharT, char_traits< _CharT > > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >  
_OI copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >  
_BI2 copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >  
_BI2 copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >  
_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >  
_OutputIterator copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits >  
streamsize copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >  
streamsize copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`

- `template<>`  
[streamsize](#) [\\_\\_copy\\_streambufs\\_eof](#) ([basic\\_streambuf](#)< wchar\_t > \*\_\_sbin, [basic\\_streambuf](#)< wchar\_t > \*\_\_sbout, bool &\_\_ineof)
- `template<>`  
[streamsize](#) [\\_\\_copy\\_streambufs\\_eof](#) ([basic\\_streambuf](#)< char > \*\_\_sbin, [basic\\_streambuf](#)< char > \*\_\_sbout, bool &\_\_ineof)
- `size_t` [\\_\\_deque\\_buf\\_size](#) (size\_t \_\_size)
- `template<typename _InputIterator >`  
[iterator\\_traits](#)< \_InputIterator >::difference\_type [\\_\\_distance](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [input\\_iterator\\_tag](#))
- `template<typename _RandomAccessIterator >`  
[iterator\\_traits](#)< \_RandomAccessIterator >::difference\_type [\\_\\_distance](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, [random\\_access\\_iterator\\_tag](#))
- `template<typename _II1, typename _II2 >`  
bool [\\_\\_equal\\_aux](#) (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2)
- `template<typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_byte](#)< \_Tp >::\_\_value, void >::\_\_type [\\_\\_fill\\_a](#) (\_Tp \* \_\_first, \_Tp \* \_\_last, const \_Tp &\_\_c)
- `template<typename _ForwardIterator, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)<! [\\_\\_is\\_scalar](#)< \_Tp >::\_\_value, void >::\_\_type [\\_\\_fill\\_a](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_value)
- void [\\_\\_fill\\_bvector](#) (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, bool \_\_x)
- `template<typename _Size, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_byte](#)< \_Tp >::\_\_value, \_Tp \* >::\_\_type [\\_\\_fill\\_n\\_a](#) (\_Tp \* \_\_first, \_Size \_\_n, const \_Tp &\_\_c)
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)<! [\\_\\_is\\_scalar](#)< \_Tp >::\_\_value, \_OutputIterator >::\_\_type [\\_\\_fill\\_n\\_a](#) (\_OutputIterator \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- `template<typename _RandomAccessIterator >`  
void [\\_\\_final\\_insertion\\_sort](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- `template<typename _RandomAccessIterator, typename _Compare >`  
void [\\_\\_final\\_insertion\\_sort](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- `template<typename _InputIterator, typename _Tp >`  
\_InputIterator [\\_\\_find](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Tp &\_\_val, [input\\_iterator\\_tag](#))
- `template<typename _RandomAccessIterator, typename _Tp >`  
\_RandomAccessIterator [\\_\\_find](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, const \_Tp &\_\_val, [random\\_access\\_iterator\\_tag](#))
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
\_FowardIterator1 [\\_\\_find\\_end](#) (\_ForwardIterator1 \_\_first1, \_ForwardIterator1

- 
- ```

__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_
iterator\_tag, forward\_iterator\_tag)

```
  - ```

• template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate
>
  _ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1
  __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_
  iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)

```
  - ```

• template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 >
  _BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,
  _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _
  BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_
  iterator\_tag)

```
  - ```

• template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _
  BinaryPredicate >
  _BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,
  _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _
  BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_
  iterator\_tag, _BinaryPredicate __comp)

```
  - ```

• template<typename _InputIterator , typename _Predicate >
  _InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate
  __pred, input\_iterator\_tag)

```
  - ```

• template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator __find_if (_RandomAccessIterator __first, _
  RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)

```
  - ```

• template<typename _InputIterator , typename _Predicate >
  _InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _
  Predicate __pred, input\_iterator\_tag)

```
  - ```

• template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator __find_if_not (_RandomAccessIterator __first, _
  RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)

```
  - ```

• template<typename _EuclideanRingElement >
  _EuclideanRingElement __gcd (_EuclideanRingElement __m, _
  EuclideanRingElement __n)

```
  - ```

• template<std::size_t __i, typename _Head , typename... _Tail>
  __add_ref< _Head >::type __get_helper (_Tuple_impl< __i, _Head, _Tail...>
  &__t)

```
  - ```

• template<std::size_t __i, typename _Head , typename... _Tail>
  __add_c_ref< _Head >::type __get_helper (const _Tuple_impl< __i, _Head,
  _Tail...> &__t)

```
  - ```

• template<typename _Ex >
  const nested\_exception * __get_nested_exception (const _Ex &__ex)

```
  - ```

• mutex & __get_once_mutex ()

```
  - ```

• template<typename _RandomAccessIterator >
  void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator _
  __middle, _RandomAccessIterator __last)

```
-

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_heap\_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`size_t \_\_iconv\_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`  
`_ForwardIterator \_\_inplace\_stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`  
`void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >`  
`_GLIBCXX_END_LDBL_NAMESPACE int \_\_int\_to\_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios\_base::fmtflags __flags, bool __dec)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void \_\_introslect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introslect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _-`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _Iter >`  
`iterator\_traits<_Iter >::iterator_category \_\_iterator\_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`  
`bool __lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2,`  
`_II2 __last2)`
- `template<typename _Size >`  
`_Size __lg (_Size __n)`
- `int __lg (int __n)`
- `long __lg (long __n)`
- `long long __lg (long long __n)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`__middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`  
`Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _-`  
`Compare >`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`__middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`  
`Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3, typename _Compare >`  
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare`  
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3 >`  
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`  
`_Distance >`  
`void __merge_sort_loop (_RandomAccessIterator1 __first, _-`  
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`  
`__step_size)`

- `template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance , typename _Compare >`  
`void \_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _-`  
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`  
`__step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator , typename _Pointer >`  
`void \_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >`  
`void \_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator , typename _Distance >`  
`void \_\_merge\_without\_buffer (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`  
`__len1, _Distance __len2)`
- `template<typename _BidirectionalIterator , typename _Distance , typename _Compare >`  
`void \_\_merge\_without\_buffer (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`  
`__len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type \_\_miter\_base (_Iterator __it)`
- `template<typename _Iterator >`  
`void \_\_move\_median\_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator , typename _Compare >`  
`void \_\_move\_median\_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare`  
`__comp)`
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type \_\_niter\_base (_Iterator __it)`
- `void \_\_once\_proxy ()`
- `template<typename _CharT , typename _Traits >`  
`void \_\_ostream\_fill (basic\_ostream< _CharT, _Traits > &__out, streamsize _-`  
`__n)`
- `template<typename _CharT , typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & \_\_ostream\_insert (basic\_ostream< _-`  
`CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template ostream & \_\_ostream\_insert (ostream &, const char *, streamsize)`
- `template wostream & \_\_ostream\_insert (wostream &, const wchar_t *, stream-`  
`size)`
- `template<typename _CharT , typename _Traits >`  
`void \_\_ostream\_write (basic\_ostream< _CharT, _Traits > &__out, const _-`  
`CharT *__s, streamsize __n)`
- `template<typename _BidirectionalIterator , typename _Predicate >`  
`_BidirectionalIterator \_\_partition (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __last, _Predicate __pred, bidirectional\_iterator\_tag)`

- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator \_\_partition (_ForwardIterator __first, _ForwardIterator __last,`  
`_Predicate __pred, forward\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`void \_\_pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp \_\_pow\_helper (_Tp __x, int __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void \_\_push\_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`  
`Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _-`  
`Compare >`  
`void \_\_push\_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`  
`Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void \_\_reverse (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator >`  
`void \_\_reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`  
`bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void \_\_rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last, forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator >`  
`void \_\_rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _-`  
`BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`void \_\_rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`Distance >`  
`_BidirectionalIterator1 \_\_rotate\_adaptive (_BidirectionalIterator1 __first, _-`  
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`  
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_-`  
`size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator \_\_search\_n (_ForwardIterator __first, _ForwardIterator __last,`  
`_Integer __count, const _Tp &__val, std::forward\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _-`  
`BinaryPredicate >`

- [\\_ForwardIterator \\_\\_search\\_n](#) ([\\_ForwardIterator \\_\\_first](#), [\\_ForwardIterator \\_\\_last](#), [\\_Integer \\_\\_count](#), [const \\_Tp &\\_\\_val](#), [\\_BinaryPredicate \\_\\_binary\\_pred](#), [std::forward\\_iterator\\_tag](#))
- [template<typename \\_RandomAccessIter , typename \\_Integer , typename \\_Tp , typename \\_BinaryPredicate >](#)  
[\\_RandomAccessIter \\_\\_search\\_n](#) ([\\_RandomAccessIter \\_\\_first](#), [\\_RandomAccessIter \\_\\_last](#), [\\_Integer \\_\\_count](#), [const \\_Tp &\\_\\_val](#), [\\_BinaryPredicate \\_\\_binary\\_pred](#), [std::random\\_access\\_iterator\\_tag](#))
  - [template<typename \\_RandomAccessIter , typename \\_Integer , typename \\_Tp >](#)  
[\\_RandomAccessIter \\_\\_search\\_n](#) ([\\_RandomAccessIter \\_\\_first](#), [\\_RandomAccessIter \\_\\_last](#), [\\_Integer \\_\\_count](#), [const \\_Tp &\\_\\_val](#), [std::random\\_access\\_iterator\\_tag](#))
  - [void \\_\\_set\\_once\\_functor\\_lock\\_ptr](#) ([unique\\_lock< mutex > \\*](#))
  - [template<typename \\_ForwardIterator , typename \\_Pointer , typename \\_Predicate , typename \\_Distance >](#)  
[\\_ForwardIterator \\_\\_stable\\_partition\\_adaptive](#) ([\\_ForwardIterator \\_\\_first](#), [\\_ForwardIterator \\_\\_last](#), [\\_Predicate \\_\\_pred](#), [\\_Distance \\_\\_len](#), [\\_Pointer \\_\\_buffer](#), [\\_Distance \\_\\_buffer\\_size](#))
  - [template<typename \\_RandomAccessIterator , typename \\_Pointer , typename \\_Distance >](#)  
[void \\_\\_stable\\_sort\\_adaptive](#) ([\\_RandomAccessIterator \\_\\_first](#), [\\_RandomAccessIterator \\_\\_last](#), [\\_Pointer \\_\\_buffer](#), [\\_Distance \\_\\_buffer\\_size](#))
  - [template<typename \\_RandomAccessIterator , typename \\_Pointer , typename \\_Distance , typename \\_Compare >](#)  
[void \\_\\_stable\\_sort\\_adaptive](#) ([\\_RandomAccessIterator \\_\\_first](#), [\\_RandomAccessIterator \\_\\_last](#), [\\_Pointer \\_\\_buffer](#), [\\_Distance \\_\\_buffer\\_size](#), [\\_Compare \\_\\_comp](#))
  - [void \\_\\_throw\\_bad\\_alloc](#) ([void](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_bad\\_cast](#) ([void](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_bad\\_exception](#) ([void](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_bad\\_function\\_call](#) ([void](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_bad\\_typeid](#) ([void](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_domain\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_future\\_error](#) ([int](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_invalid\\_argument](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_ios\\_failure](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_length\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_logic\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_out\\_of\\_range](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_overflow\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_range\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_runtime\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_system\\_error](#) ([int](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)
  - [void \\_\\_throw\\_underflow\\_error](#) ([const char \\*](#)) [\\_\\_attribute\\_\\_\(\(\\_\\_noreturn\\_\\_\)\)](#)

- `template<typename _Ex >`  
`void __throw_with_nested (_Ex &&, const nested\_exception *=0) __-`  
`attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void __throw_with_nested (_Ex &&,...) __attribute__((__noreturn__))`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _TElements...> &_t, const \_\_index\_holder< _TIdx...> &, const tuple< _UElements...> &_u, const \_\_index\_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _TElements...> &&_t, const \_\_index\_holder< _TIdx...> &, const tuple< _UElements...> &_u, const \_\_index\_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _TElements...> &_t, const \_\_index\_holder< _TIdx...> &, tuple< _UElements...> &&_u, const \_\_index\_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _TElements...> &&_t, const \_\_index\_holder< _TIdx...> &, tuple< _UElements...> &&_u, const \_\_index\_holder< _UIdx...> &)`
- `template<typename _RandomAccessIterator >`  
`void \_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, __- RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, __- RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`\_RandomAccessIterator \_\_unguarded\_partition (\_RandomAccessIterator __first, \_RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`\_RandomAccessIterator \_\_unguarded\_partition (\_RandomAccessIterator __first, \_RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`\_RandomAccessIterator \_\_unguarded\_partition\_pivot (\_RandomAccessIterator __first, \_RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator \_\_unguarded\_partition\_pivot (_RandomAccessIterator`  
`__first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void \_\_uninitialized\_construct\_range (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator`  
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator`  
`__last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`  
`name _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_move (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_InputIterator __first, _Size __n, _-`  
`ForwardIterator __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_RandomAccessIterator __first, _-`  
`Size __n, _ForwardIterator __result, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void \_\_uninitialized\_fill\_a (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void \_\_uninitialized\_fill\_a (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _-`  
`Allocator >`  
`_ForwardIterator \_\_uninitialized\_fill\_move (_ForwardIterator __result, _-`  
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`  
`__last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`void \_\_uninitialized\_fill\_n\_a (_ForwardIterator __first, _Size __n, const _Tp`  
`&__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`void \_\_uninitialized\_fill\_n\_a (_ForwardIterator __first, _Size __n, const _Tp`  
`&__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_move\_a (_InputIterator __first, _InputIterator`  
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`  
`name _Allocator >`

- ```

    _ForwardIterator __uninitialized_move_copy (_InputIterator __first1, _-
    InputIterator __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-
    ForwardIterator __result, _Allocator &__alloc)

```
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _-
 Allocator >`  
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator`  
`&__alloc)`
  - `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __-`  
`last, _OutputIterator __result, forward\_iterator\_tag, output\_iterator\_tag)`
  - `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, input\_iterator\_tag, output\_iterator\_tag)`
  - `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _-`  
`ForwardIterator __result, input\_iterator\_tag, forward\_iterator\_tag)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator _-`  
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward\_-`  
`iterator\_tag, output\_iterator\_tag)`
  - `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag,`  
`output\_iterator\_tag)`
  - `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last,`  
`_ForwardIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag,`  
`forward\_iterator\_tag)`
  - `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __a, size_t __n, _Tp *__-`  
`restrict__ __b)`
  - `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp`  
`*__restrict__ __b)`
  - `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __a, _Tp *__restrict__ __b,`  
`size_t __n, size_t __s)`
  - `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __src, size_t __n, size_t __s1,`  
`_Tp *__restrict__ __dst, size_t __s2)`
  - `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __a, const size_t *__restrict__`  
`__i, _Tp *__restrict__ __b, size_t __n)`
-

- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b, const size_t *__restrict__ __i)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict__ __src, size_t __n, const size_t *__restrict__ __i, _Tp *__restrict__ __dst, const size_t *__restrict__ __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`

- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void *__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `_GLIBCXX_PURE bool __verify_grouping (const char *__grouping, size_t __-`  
`__grouping_size, const string &__grouping_tmp) throw ()`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s,`  
`const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _-`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _-`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > _-`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _-`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > _-`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > _-`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > _-`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp`  
`&__t)`

- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __`  
`_s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b,`  
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __`  
`m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp`  
`&__t)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b,`  
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`  
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`  
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const`  
`__Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`  
`> __i, const __Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`  
`__m, const __Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const`  
`__Tp &__t)`
- `template<typename _T1, typename _T2 >`  
`void _Construct (_T1 *__p, _T2 &&__value)`
- `template<typename _ForwardIterator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator`  
`&__alloc)`
- `template<typename _Tp >`  
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _-`  
`__Tp > &)`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_-`  
`node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header)`  
`throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base`  
`*const __z, _Rb_tree_node_base &__header) throw ()`
- `void abort (void) _GLIBCXX_NORETURN throw ()`

- float **abs** (float \_\_x)
- long double **abs** (long double \_\_x)
- template<typename \_Tp >  
\_Tp **abs** (const [complex](#)<\_Tp > &)
- double **abs** (double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer<\_Tp >::\_\_value, double >::\_\_type **abs**  
(\_Tp \_\_x)
- template<class \_Dom >  
\_Expr<\_UnClos<\_Abs, \_Expr, \_Dom >, typename \_Dom::value\_type > **abs**  
(const \_Expr<\_Dom, typename \_Dom::value\_type > &\_e)
- template<typename \_Tp >  
\_Expr<\_UnClos<\_Abs, \_ValArray, \_Tp >, \_Tp > **abs** (const [valarray](#)<\_Tp  
> &\_v)
- template<typename \_InputIterator, typename \_Tp >  
\_Tp **accumulate** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init)
- template<typename \_InputIterator, typename \_Tp, typename \_BinaryOperation >  
\_Tp **accumulate** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init, \_-  
BinaryOperation \_\_binary\_op)
- float **acos** (float \_\_x)
- long double **acos** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer<\_Tp >::\_\_value, double >::\_\_type  
**acos** (\_Tp \_\_x)
- template<class \_Dom >  
\_Expr<\_UnClos<\_Acos, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**acos** (const \_Expr<\_Dom, typename \_Dom::value\_type > &\_e)
- template<typename \_Tp >  
\_Expr<\_UnClos<\_Acos, \_ValArray, \_Tp >, \_Tp > **acos** (const [valarray](#)<\_Tp  
> &\_v)
- template<typename \_InputIterator, typename \_OutputIterator >  
\_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last,  
\_OutputIterator \_\_result)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_BinaryOperation >  
\_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last,  
\_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename \_FIter >  
\_FIter **adjacent\_find** (\_FIter, \_FIter)
- template<typename \_FIter, typename \_BinaryPredicate >  
\_FIter **adjacent\_find** (\_FIter, \_FIter, \_BinaryPredicate)
- template<typename \_ForwardIterator >  
\_ForwardIterator **adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_-  
last)

- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator adjacent\_find (_ForwardIterator __first, _ForwardIterator __-`  
`last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`  
`void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _Iter, typename _Predicate >`  
`bool all\_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared\_ptr< _Tp > allocate\_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Iter, typename _Predicate >`  
`bool any\_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp arg (const complex< _Tp > &)`
- `float asin (float __x)`
- `long double asin (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`asin (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp`  
`> &__v)`
- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::__type > async (launch __policy,`  
`_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`  
`enable\_if<!is_same< typename decay< _Fn >::__type, launch >::__value, future<`  
`decltype(std::declval< _Fn >)(std::declval< _Args >...)> >::__type async (_Fn`  
`&&__fn, _Args &&...__args)`
- `float atan (float __x)`
- `long double atan (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`atan (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type >`  
`atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &_v)`
- `float atan2 (float __y, float __x)`
- `long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp >::__type, _Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &_v, const _Tp &_t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &_e1, const _Expr< _Dom2, typename _Dom2::value_type > &_e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &_e, const typename _Dom::value_type &_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &_v, const valarray< _Tp > &_w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2 (const _Tp &_t, const valarray< _Tp > &_v)`
- `int atexit (void(*)()) throw ()`
- `bool atomic_compare_exchange_strong (atomic_bool * __a, bool * __i1, bool __i2)`

- `bool atomic_compare_exchange_strong (atomic_address *__a, void **__v1, void *__v2)`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (__atomic_base<_ITp > *__a, _ITp *__i1, _ITp __i2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (__atomic_base<_ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak (atomic_bool *__a, bool *__i1, bool __i2)`
- `bool atomic_compare_exchange_weak (atomic_address *__a, void **__v1, void *__v2)`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (__atomic_base<_ITp > *__a, _ITp *__i1, _ITp __i2)`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (__atomic_base<_ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak_explicit (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak_explicit (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_exchange (atomic_bool *__a, bool __i)`
- `void * atomic_exchange (atomic_address *__a, void *__v)`
- `template<typename _ITp >`  
`_ITp atomic_exchange (__atomic_base<_ITp > *__a, _ITp __i)`
- `bool atomic_exchange_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (__atomic_base<_ITp > *__a, _ITp __i, memory_order __m)`
- `void * atomic_exchange_explicit (atomic_address *__a, void *__v, memory_order __m)`
- `void * atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (__atomic_base<_ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (__atomic_base<_ITp > *__a, _ITp __i, memory_order __m)`
- `void * atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d, memory_order __m)`

- `template<typename _ITp >`  
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `void * atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `void atomic_flag_clear (__atomic_flag_base *__a)`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory\_order __m)`
- `void atomic_flag_clear_explicit (__atomic_flag_base *, memory\_order) \_GLIBCXX\_NOTHROW`
- `bool atomic_flag_test_and_set (__atomic_flag_base *__a)`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory\_order __m)`
- `bool atomic_flag_test_and_set_explicit (__atomic_flag_base *, memory\_order) \_GLIBCXX\_NOTHROW`
- `bool atomic_is_lock_free (const atomic_bool *__a)`
- `bool atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`  
`bool atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool atomic_load (const atomic_bool *__a)`
- `void * atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp atomic_load (const __atomic_base< _ITp > *__a)`
- `bool atomic_load_explicit (const atomic_bool *__a, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const __atomic_base< _ITp > *__a, memory\_order __m)`

- void \* **atomic\_load\_explicit** (const atomic\_address \* \_\_a, [memory\\_order](#) \_\_m)
- void **atomic\_store** (atomic\_address \* \_\_a, void \* \_\_v)
- void **atomic\_store** (atomic\_bool \* \_\_a, bool \_\_i)
- template<typename \_ITp >  
void **atomic\_store** (\_\_atomic\_base< \_ITp > \* \_\_a, \_ITp \_\_i)
- void **atomic\_store\_explicit** (atomic\_bool \* \_\_a, bool \_\_i, [memory\\_order](#) \_\_m)
- template<typename \_ITp >  
void **atomic\_store\_explicit** (\_\_atomic\_base< \_ITp > \* \_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m)
- void **atomic\_store\_explicit** (atomic\_address \* \_\_a, void \* \_\_v, [memory\\_order](#) \_\_m)
- template<typename \_Container >  
[back\\_insert\\_iterator](#)< \_Container > [back\\_inserter](#) (\_Container & \_\_x)
- template<typename \_Filter, typename \_Tp, typename \_Compare >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &, \_Compare)
- template<typename \_ForwardIterator, typename \_Tp >  
bool [binary\\_search](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
bool [binary\\_search](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_Filter, typename \_Tp >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &)
- template<typename \_Functor, typename... \_ArgTypes>  
\_Bind< typename [Maybe\\_wrap\\_member\\_pointer](#)< \_Functor >::type(\_ArgTypes...) > [bind](#) (\_Functor \_\_f, \_ArgTypes... \_\_args)
- template<typename \_Result, typename \_Functor, typename... \_ArgTypes>  
\_Bind\_result< \_Result, typename [Maybe\\_wrap\\_member\\_pointer](#)< \_Functor >::type(\_ArgTypes...) > [bind](#) (\_Functor \_\_f, \_ArgTypes... \_\_args)
- template<typename \_Operation, typename \_Tp >  
[binder1st](#)< \_Operation > [bind1st](#) (const \_Operation & \_\_fn, const \_Tp & \_\_x)
- template<typename \_Operation, typename \_Tp >  
[binder2nd](#)< \_Operation > [bind2nd](#) (const \_Operation & \_\_fn, const \_Tp & \_\_x)
- [ios\\_base](#) & [boolalpha](#) ([ios\\_base](#) & \_\_base)
- template<typename \_Callable, typename... \_Args>  
void [call\\_once](#) ([once\\_flag](#) & \_\_once, \_Callable \_\_f, \_Args &&... \_\_args)
- float **ceil** (float \_\_x)
- long double **ceil** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::enable\_if< \_\_is\_integer< \_Tp >::value, double >::\_\_type **ceil** (\_Tp \_\_x)
- template<typename \_Tp >  
[complex](#)< \_Tp > [conj](#) (const [complex](#)< \_Tp > &)

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type conj (_Tp __x)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, const _  
Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp *  
> __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_  
iterator< _CharT > >::__type copy (istreambuf_iterator< _CharT > __first,  
istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __  
result)`
- `template<typename _Iter, typename _OIter >`  
`_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _II, typename _OI >`  
`_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, _Tp &,  
_Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_  
iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator<  
_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &,  
const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BIter1, typename _BIter2 >`  
`_BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator<  
_Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last,  
_Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Ex >`  
`exception_ptr copy_exception (_Ex __ex) throw ()`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _  
OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __  
result)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`_OIter copy_n (_Iter, _Size, _OIter)`

- float **cos** (float \_\_x)
- long double **cos** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **cos**  
(\_Tp \_\_x)
- template<typename \_Tp >  
**complex**< \_Tp > **cos** (const **complex**< \_Tp > &)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Cos, \_Expr, \_Dom >, typename \_Dom::value\_type > **cos**  
(const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Cos, \_ValArray, \_Tp >, \_Tp > **cos** (const **valarray**< \_Tp  
> &\_\_v)
- float **cosh** (float \_\_x)
- long double **cosh** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**cosh** (\_Tp \_\_x)
- template<typename \_Tp >  
**complex**< \_Tp > **cosh** (const **complex**< \_Tp > &)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Cosh, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**cosh** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Cosh, \_ValArray, \_Tp >, \_Tp > **cosh** (const **valarray**< \_Tp  
> &\_\_v)
- template<typename \_Iter, typename \_Tp >  
**iterator\_traits**< \_Iter >::difference\_type **count** (\_Iter, \_Iter, const \_Tp &)
- template<typename \_InputIterator, typename \_Tp >  
**iterator\_traits**< \_InputIterator >::difference\_type **count** (\_InputIterator \_\_first,  
\_InputIterator \_\_last, const \_Tp &\_\_value)
- template<typename \_Iter, typename \_Predicate >  
**iterator\_traits**< \_Iter >::difference\_type **count\_if** (\_Iter, \_Iter, \_Predicate)
- template<typename \_InputIterator, typename \_Predicate >  
**iterator\_traits**< \_InputIterator >::difference\_type **count\_if** (\_InputIterator \_\_-  
first, \_InputIterator \_\_last, \_Predicate \_\_pred)
- exception\_ptr **current\_exception** () throw ()
- ios\_base & **dec** (ios\_base &\_\_base)
- template<typename \_Tp >  
**add\_rvalue\_reference**< \_Tp >::type **declval** ()
- template<typename \_InputIterator >  
**iterator\_traits**< \_InputIterator >::difference\_type **distance** (\_InputIterator \_\_-  
first, \_InputIterator \_\_last)
- template<typename \_Tp, typename \_Tp1 >  
**shared\_ptr**< \_Tp > **dynamic\_pointer\_cast** (const **shared\_ptr**< \_Tp1 > &\_\_r)

- `template<typename _CharT, typename _Traits >`  
[basic\\_ostream](#)< \_CharT, \_Traits > & [endl](#) ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os)
- `template<typename _CharT, typename _Traits >`  
[basic\\_ostream](#)< \_CharT, \_Traits > & [ends](#) ([basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os)
- `template<typename _IIter1, typename _IIter2 >`  
**bool equal** (\_IIter1, \_IIter1, \_IIter2)
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`  
**bool equal** (\_IIter1 \_\_first1, \_IIter1 \_\_last1, \_IIter2 \_\_first2, \_BinaryPredicate \_\_binary\_pred)
- `template<typename _II1, typename _II2 >`  
**bool equal** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2)
- `template<typename _FIter, typename _Tp >`  
[pair](#)< \_FIter, \_FIter > **equal\_range** (\_FIter, \_FIter, const \_Tp &)
- `template<typename _FIter, typename _Tp, typename _Compare >`  
[pair](#)< \_FIter, \_FIter > **equal\_range** (\_FIter, \_FIter, const \_Tp &, \_Compare)
- `template<typename _ForwardIterator, typename _Tp >`  
[pair](#)< \_ForwardIterator, \_ForwardIterator > **equal\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val)
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
[pair](#)< \_ForwardIterator, \_ForwardIterator > **equal\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- `void exit` (int) \_GLIBCXX\_NORETURN throw ()
- `float exp` (float \_\_x)
- `long double exp` (long double \_\_x)
- `template<typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_integer](#)< \_Tp >::\_\_value, double >::\_\_type **exp** (\_Tp \_\_x)
- `template<typename _Tp >`  
[complex](#)< \_Tp > **exp** (const [complex](#)< \_Tp > &)
- `template<class _Dom >`  
[\\_Expr](#)< [\\_UnClos](#)< \_Exp, \_Expr, \_Dom >, typename \_Dom::value\_type > **exp** (const [\\_Expr](#)< \_Dom, typename \_Dom::value\_type > & \_\_e)
- `template<typename _Tp >`  
[\\_Expr](#)< [\\_UnClos](#)< \_Exp, \_ValArray, \_Tp >, \_Tp > **exp** (const [valarray](#)< \_Tp > & \_\_v)
- `float fabs` (float \_\_x)
- `long double fabs` (long double \_\_x)
- `template<typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_integer](#)< \_Tp >::\_\_value, double >::\_\_type **fabs** (\_Tp \_\_x)
- `template<typename _Tp >`  
**void fill** (const [Deque\\_iterator](#)< \_Tp, \_Tp &, \_Tp \* > & \_\_first, const [Deque\\_iterator](#)< \_Tp, \_Tp &, \_Tp \* > & \_\_last, const \_Tp & \_\_value)

- `template<typename _ForwardIterator, typename _Tp >`  
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Filter, typename _Tp >`  
`void fill (_Filter, _Filter, const _Tp &)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred)`
- `ios_base & fixed (ios_base &__base)`
- `long double floor (long double __x)`
- `float floor (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits >`  
`&__os)`
- `float fmod (float __x, float __y)`
- `long double fmod (long double __x, long double __y)`
- `template<typename _Iter, typename _Funct >`  
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`  
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`  
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`  
`name std::identity< _Tp >::type &__t)`
- `template<typename _Tp >`  
`enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`  
`name std::identity< _Tp >::type &&__t)`
- `template<typename _Tp >`  
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`  
`name std::identity< _Tp >::type __t)`
- `template<typename _Tp >`  
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`  
`name std::remove_reference< _Tp >::type &&__t)`
- `float frexp (float __x, int *__exp)`
- `long double frexp (long double __x, int *__exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type fr-`  
`exp (_Tp __x, int *__exp)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > front_inserter (_Container &__x)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __-`  
`gen)`

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType generate\_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator > _OIter generate\_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator > _OutputIterator generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `_GLIBCXX_CONST const error\_category & generic\_category () throw ()`
- `template<std::size_t __i, typename... _Elements> __add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type get (tuple< _Elements...> &__t)`
- `template<std::size_t __i, typename... _Elements> __add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type get (const tuple< _Elements...> &__t)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del * get\_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _MoneyT > _Get_money< _MoneyT > get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _Tp > pair< _Tp *, ptrdiff_t > get\_temporary\_buffer (ptrdiff_t __len)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<> basic_istream< char > & getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<> basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _Facet > bool has\_facet (const locale &__loc) throw ()`
- `ios_base & hex (ios_base &__base)`

- `template<typename _Tp >`  
`_Tp imag (const complex< _Tp > &_z)`
- `template<typename _Iter1 , typename _Iter2 >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1 , typename _Iter2 , typename _Compare >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1 , typename _InputIterator2 >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _BinaryOperation1 , typename _BinaryOperation2 >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BIter >`  
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter , typename _Compare >`  
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator , typename _Compare >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Container , typename _Iterator >`  
`insert\_iterator< _Container > inserter (_Container &__x, _Iterator __i)`
- `ios\_base & internal (ios\_base & __base)`
- `template<typename _ForwardIterator , typename _Tp >`  
`void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter >`  
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter , typename _Compare >`  
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`  
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RAIter, typename _Compare >`  
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __-`  
`pred)`
- `template<typename _Iter, typename _Predicate >`  
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _ForwardIterator >`  
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __-`  
`comp)`
- `template<typename _Filter >`  
`bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`  
`last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Compare __comp)`
- `template<typename _CharT >`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool iscentrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isgraph (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _Filter1 , typename _Filter2 >`  
`void iter_swap (_Filter1, _Filter2)`
- `template<typename _ForwardIterator1 , typename _ForwardIterator2 >`  
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Tp >`  
`_Tp kill_dependency (_Tp __y)`
- `float ldexp (float __x, int __exp)`
- `long double ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _IIter1 , typename _IIter2 >`  
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2)`
- `template<typename _IIter1 , typename _IIter2 , typename _Compare >`  
`bool lexicographical_compare (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)`
- `template<typename _II1 , typename _II2 >`  
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1 , typename _II2 , typename _Compare >`  
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _L1 , typename _L2 , typename... _L3>`  
`void lock (_L1 &, _L2 &, _L3 &...)`
- `template<typename _Tp >`  
`complex< _Tp > log (const complex< _Tp > &)`
- `float log (float __x)`
- `long double log (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`

- `template<typename _Tp >`  
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`complex< _Tp > log10 (const complex< _Tp > &)`
- `float log10 (float __x)`
- `long double log10 (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log10 (_Tp __x)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Filter, typename _Tp >`  
`_Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `error\_code make_error_code (errc __e)`
- `error\_code make_error_code (future\_errc __errc)`
- `error\_condition make_error_condition (errc __e)`
- `error\_condition make_error_condition (future\_errc __errc)`
- `template<typename _RAIter >`  
`void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _Iterator >`  
`move\_iterator< _Iterator > make_move_iterator (const _Iterator &__i)`

- `template<class _T1, class _T2 >`  
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >` **make\_pair** (`_T1 &&_x, _T2 &&_y`)
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _Tp >` **make\_shared** (`_Args &&..._args`)
- `template<typename... _Elements>`  
`tuple< typename __decay_and_strip< _Elements >::__type...>` **make\_tuple** (`_Elements &&..._args`)
- `template<typename _Tp, typename _Compare >`  
`const _Tp &` **max** (`const _Tp &_a, const _Tp &_b, _Compare __comp`)
- `template<typename _Tp >`  
`_Tp` **max** (`initializer_list< _Tp >`)
- `template<typename _Tp, typename _Compare >`  
`_Tp` **max** (`initializer_list< _Tp >, _Compare`)
- `template<typename _Tp >`  
`const _Tp &` **max** (`const _Tp &_a, const _Tp &_b`)
- `template<typename _FIter >`  
`_FIter` **max\_element** (`_FIter, _FIter`)
- `template<typename _FIter, typename _Compare >`  
`_FIter` **max\_element** (`_FIter, _FIter, _Compare`)
- `template<typename _ForwardIterator >`  
`_ForwardIterator` **max\_element** (`_ForwardIterator __first, _ForwardIterator __last`)
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator` **max\_element** (`_ForwardIterator __first, _ForwardIterator __last, _Compare __comp`)
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* >` **mem\_fn** (`_Tp _Class::*_pm`)
- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp >` **mem\_fun** (`_Ret(_Tp::*_f)()`)
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg >` **mem\_fun** (`_Ret(_Tp::*_f)(_Arg)`)
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg >` **mem\_fun\_ref** (`_Ret(_Tp::*_f)(_Arg)`)
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp >` **mem\_fun\_ref** (`_Ret(_Tp::*_f)()`)
- `void *` **memchr** (`void *_s, int _c, size_t _n`)
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter` **merge** (`_IIter1, _IIter1, _IIter2, _IIter2, _OIter`)
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter` **merge** (`_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator` **merge** (`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result`)

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp min (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp min (initializer\_list< _Tp >, _Compare)`
- `template<typename _FIter >`  
`_FIter min\_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min\_element (_FIter, _FIter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __-`  
`last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b,`  
`_Compare __comp)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > minmax (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > minmax (initializer\_list< _Tp >, _Compare)`
- `template<typename _FIter >`  
`pair< _FIter, _FIter > minmax\_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`pair< _FIter, _FIter > minmax\_element (_FIter, _FIter, _Compare)`
- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator`  
`__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator`  
`__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`

- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_-`  
`pred)`
- `float modf (float __x, float * __iptr)`
- `long double modf (long double __x, long double * __iptr)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move ( _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp *`  
`> __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`std::remove_reference< _Tp >::type && move ( _Tp && __t)`
- `template<typename _II, typename _OI >`  
`_OI move ( _II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move ( _Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_-`  
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward ( _Deque_iterator<`  
`_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &,`  
`const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 move_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward ( _Deque_iterator<`  
`_Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last,`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _ForwardIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _ForwardIterator >::__value, _-`  
`ForwardIterator >::__type next ( _ForwardIterator __x, typename iterator_-`  
`traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BIter >`  
`bool next_permutation ( _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool next_permutation ( _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`bool next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __-`  
`last)`

- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base &__base)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _Tp >`  
`_Tp norm (const complex< _Tp > &)`
- `ios_base & noshowbase (ios_base &__base)`
- `ios_base & noshowpoint (ios_base &__base)`
- `ios_base & noshowpos (ios_base &__base)`
- `ios_base & noskipws (ios_base &__base)`
- `template<typename _Predicate >`  
`unary_negate< _Predicate > not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > not2 (const _Predicate &__pred)`
- `ios_base & nunitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base &__base)`
- `template<typename _Iterator >`  
`bool operator!= (const reverse_iterator< _Iterator > &__x, const reverse_-`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_-`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_-`  
`iterator< _IteratorR > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const`  
`std::uniform_int_distribution< _IntType > &__d2)`

- `template<typename _Val >`  
`bool operator!= (const \_List\_iterator< _Val > &__x, const \_List\_const\_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`bool operator!= (const \_Fwd\_list\_iterator< _Tp > &__x, const \_Fwd\_list\_const\_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::geometric\_distribution< _IntType > &__d1, const std::geometric\_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::binomial\_distribution< _IntType > &__d1, const std::binomial\_distribution< _IntType > &__d2)`
- `bool operator!= (const error\_code &__lhs, const error\_code &__rhs)`
- `template<typename _RealType >`  
`bool operator!= (const std::student\_t\_distribution< _RealType > &__d1, const std::student\_t\_distribution< _RealType > &__d2)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator!= (const \_Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const \_Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`

- `template<typename _RealType >`  
`bool operator!= (const std::cauchy\_distribution< _RealType > &__d1, const std::cauchy\_distribution< _RealType > &__d2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool operator!= (const istream\_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream\_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator!= (const unique\_ptr< _Tp, _Tp_Deleter > &__x, const unique\_ptr< _Up, _Up_Deleter > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code >`  
`bool operator!= (const \_\_unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const \_\_unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code >`  
`bool operator!= (const \_\_unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const \_\_unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code >`  
`bool operator!= (const \_\_unordered\_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const \_\_unordered\_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `bool operator!= (const std::bernoulli\_distribution &__d1, const std::bernoulli\_distribution &__d2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 > &__b)`
- `template<class _Dom1, class _Dom2 >`  
`\_Expr< \_BinClos< \_\_not\_equal\_to, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_not\_equal\_to, typename \_Dom1::value\_type >::result_type >`  
`operator!= (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &__v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &__w)`
- `template<class _Dom >`  
`\_Expr< \_BinClos< \_\_not\_equal\_to, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_not\_equal\_to, typename`

- `_Dom::value_type >::result_type > operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _RealType >  
bool operator!= (const std::weibull\_distribution< _RealType > &__d1, const std::weibull\_distribution< _RealType > &__d2)`
  - `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
  - `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
  - `template<typename _StateT >  
bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
  - `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>  
bool operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
  - `template<typename _RandomNumberEngine, size_t __k>  
bool operator!= (const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__rhs)`
  - `template<class _T1, class _T2 >  
bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
  - `template<typename _IntType >  
bool operator!= (const std::uniform\_real\_distribution< _IntType > &__d1, const std::uniform\_real\_distribution< _IntType > &__d2)`
  - `template<typename _RealType >  
bool operator!= (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _RealType > &__d2)`
  - `template<typename _RealType >  
bool operator!= (const std::chi\_squared\_distribution< _RealType > &__d1, const std::chi\_squared\_distribution< _RealType > &__d2)`
  - `template<typename _Val >  
bool operator!= (const \_Rb\_tree\_iterator< _Val > &__x, const \_Rb\_tree\_iterator< _Val > &__y)`
  - `template<typename _RealType >  
bool operator!= (const std::fisher\_f\_distribution< _RealType > &__d1, const std::fisher\_f\_distribution< _RealType > &__d2)`

- `template<typename _RandomNumberEngine , size_t _p, size_t _r>`  
`bool operator!= (const std::discard\_block\_engine< _RandomNumberEngine, _p, _r > &_lhs, const std::discard\_block\_engine< _RandomNumberEngine, _p, _r > &_rhs)`
- `template<typename _UIntType , size_t _w, size_t _n, size_t _m, size_t _r, _UIntType _a, size_t _u, _UIntType _d, size_t _s, _UIntType _b, size_t _t, _UIntType _c, size_t _l, _UIntType _f>`  
`bool operator!= (const std::mersenne\_twister\_engine< _UIntType, _w, _n, _m, _r, _a, _u, _d, _s, _b, _t, _c, _l, _f > &_lhs, const std::mersenne\_twister\_engine< _UIntType, _w, _n, _m, _r, _a, _u, _d, _s, _b, _t, _c, _l, _f > &_rhs)`
- `template<typename _UIntType , size_t _w, size_t _s, size_t _r>`  
`bool operator!= (const std::subtract\_with\_carry\_engine< _UIntType, _w, _s, _r > &_lhs, const std::subtract\_with\_carry\_engine< _UIntType, _w, _s, _r > &_rhs)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc , bool _cache_hash_code>`  
`bool operator!= (const \_\_unordered\_multiset< _Value, _Hash, _Pred, _Alloc, _cache_hash_code > &_x, const \_\_unordered\_multiset< _Value, _Hash, _Pred, _Alloc, _cache_hash_code > &_y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >`  
`bool operator!= (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`bool operator!= (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &_x, const unordered\_set< _Value, _Hash, _Pred, _Alloc > &_y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<typename _CharT , typename _Traits >`  
`bool operator!= (const istreambuf\_iterator< _CharT, _Traits > &_a, const istreambuf\_iterator< _CharT, _Traits > &_b)`
- `template<typename _RealType >`  
`bool operator!= (const std::exponential\_distribution< _RealType > &_d1, const std::exponential\_distribution< _RealType > &_d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::negative\_binomial\_distribution< _IntType > &_d1, const std::negative\_binomial\_distribution< _IntType > &_d2)`
- `bool operator!= (const error\_condition &_lhs, const error\_code &_rhs)`
- `template<typename _RealType >`  
`bool operator!= (const std::extreme\_value\_distribution< _RealType > &_d1, const std::extreme\_value\_distribution< _RealType > &_d2)`

- `template<typename _IntType >`  
`bool operator!= (const std::poisson\_distribution< _IntType > &__d1, const std::poisson\_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::lognormal\_distribution< _RealType > &__d1, const std::lognormal\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::gamma\_distribution< _RealType > &__d1, const std::gamma\_distribution< _RealType > &__d2)`
- `bool operator!= (thread::id __x, thread::id __y)`
- `bool operator!= (const error\_condition &__lhs, const error\_condition &__rhs)`
- `template<typename _IntType >`  
`bool operator!= (const std::discrete\_distribution< _IntType > &__d1, const std::discrete\_distribution< _IntType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool operator!= (const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Res, typename... _Args >`  
`bool operator!= (const function< _Res(_Args...) > &__f, _M_clear_type *)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise\_constant\_distribution< _RealType > &__d1, const std::piecewise\_constant\_distribution< _RealType > &__d2)`
- `template<typename _Res, typename... _Args >`  
`bool operator!= (_M_clear_type *, const function< _Res(_Args...) > &__f)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise\_linear\_distribution< _RealType > &__d1, const std::piecewise\_linear\_distribution< _RealType > &__d2)`
- `template<typename _Tp >`  
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _T1, typename _T2 >`  
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator!= (const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator!= (const \_Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const \_Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray`  
`< _Tp > &__v, const _Tp &__t)`
- `template<typename... _TElements, typename... _UElements >`  
`bool operator!= (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const _Tp &__-`  
`__t, const valarray< _Tp > &__v)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`  
`> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `bool operator!= (const error\_code &__lhs, const error\_condition &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`_CharT *__rhs)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`  
`__fun< __modulus, typename _Dom1::value_type >::result_type > operator%`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`  
`type >::result_type > operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`  
`type >::result_type > operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > operator% (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`

- 
- >::result\_type > **operator%** (const typename \_Dom::value\_type &\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v)
  - template<typename \_Tp >  
\_Expr< \_BinClos< \_\_modulus, \_ValArray, \_ValArray, \_Tp, \_Tp >, typename \_\_fun< \_\_modulus, \_Tp >::result\_type > **operator%** (const valarray< \_Tp > &\_v, const valarray< \_Tp > &\_w)
  - template<typename \_Tp >  
\_Expr< \_BinClos< \_\_modulus, \_ValArray, \_Constant, \_Tp, \_Tp >, typename \_\_fun< \_\_modulus, \_Tp >::result\_type > **operator%** (const valarray< \_Tp > &\_v, const \_Tp &\_t)
  - template<typename \_Tp >  
\_Expr< \_BinClos< \_\_modulus, \_Constant, \_ValArray, \_Tp, \_Tp >, typename \_\_fun< \_\_modulus, \_Tp >::result\_type > **operator%** (const \_Tp &\_t, const valarray< \_Tp > &\_v)
  - \_Ios\_Fmtflags **operator&** (\_Ios\_Fmtflags \_\_a, \_Ios\_Fmtflags \_\_b)
  - \_Ios\_Openmode **operator&** (\_Ios\_Openmode \_\_a, \_Ios\_Openmode \_\_b)
  - \_Ios\_Iostate **operator&** (\_Ios\_Iostate \_\_a, \_Ios\_Iostate \_\_b)
  - template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom1::value\_type >::result\_type > **operator&** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_w)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **operator&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_e, const valarray< typename \_Dom::value\_type > &\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **operator&** (const valarray< typename \_Dom::value\_type > &\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_e)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **operator&** (const typename \_Dom::value\_type &\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **operator&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v, const typename \_Dom::value\_type &\_t)
-

- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valarray`  
`< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valarray`  
`< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const _Tp &__-`  
`__t, const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`  
`__fun< __logical_and, typename _Dom1::value_type >::result_type > opera-`  
`tor&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > operator&& (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > operator&& (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > operator&& (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valarray`  
`< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valarray`  
`< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const _Tp`  
`&__t, const valarray< _Tp > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > operator&& (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `_Ios_Openmode & operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`  
`type >::result_type > operator* (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`  
`>::result_type > operator* (const typename _Dom::value_type &__t, const _`  
`Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`  
`type >::result_type > operator* (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`  
`>::result_type > operator* (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __multiplies, typename _Dom1::value_type >::result_type >`  
`operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`

- `__fun< __multiplies, _Tp >::result_type > operator* (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Iterator > reverse\_iterator< _Iterator > operator+ (typename reverse\_iterator< _Iterator >::difference_type __n, const reverse\_iterator< _Iterator > &__x)`
  - `template<typename _Iterator > move\_iterator< _Iterator > operator+ (typename move\_iterator< _Iterator >::difference_type __n, const move\_iterator< _Iterator > &__x)`
  - `template<class _Dom > \_Expr< \_BinClos< __plus, _Expr, \_ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const \_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
  - `template<class _Dom > \_Expr< \_BinClos< __plus, _Expr, \_Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const \_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
  - `template<class _Dom > \_Expr< \_BinClos< __plus, \_Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const \_Expr< _Dom, typename _Dom::value_type > &__v)`
  - `template<class _Dom > \_Expr< \_BinClos< __plus, \_ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const \_Expr< _Dom, typename _Dom::value_type > &__e)`
  - `template<class _Dom1, class _Dom2 > \_Expr< \_BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const \_Expr< _Dom1, typename _Dom1::value_type > &__v, const \_Expr< _Dom2, typename _Dom2::value_type > &__w)`
  - `template<typename _Tp > complex< _Tp > operator+ (const complex< _Tp > &__x)`
  - `template<typename _Tp > \_Expr< \_BinClos< __plus, \_ValArray, \_ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
  - `template<typename _Tp > \_Expr< \_BinClos< __plus, \_ValArray, \_Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
  - `template<typename _Tp > \_Expr< \_BinClos< __plus, \_Constant, \_ValArray, _Tp, _Tp >, typename __-`

```

fun< __plus, _Tp >::result_type > operator+ (const _Tp &__t, const valarray<
_Tp > &__v)

```

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_`  
`string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const _Deque_`  
`iterator< _Tp, _Ref, _Ptr > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::difference_type operator- (const reverse_`  
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`  
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator<`  
`_IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const _`  
`Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`  
`RefR, _PtrR > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`  
`type > operator- (const typename _Dom::value_type &__t, const _Expr< _`  
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename _`  
`fun< __minus, typename _Dom1::value_type >::result_type > operator- (const`

- `_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
  - `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
  - `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v, const _Tp &__t)`
  - `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type > operator- (const _Tp &__t, const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
  - `template<typename _Tp >`  
`complex< _Tp > operator- (const complex< _Tp > &__x)`
  - `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)`
  - `template<typename _Tp, typename _Ref, typename _Ptr >`  
`Deque\_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
  - `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`

- `_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
  - `template<class _Dom >  
_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
  - `template<class _Dom >  
_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
  - `template<typename _Tp >  
_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
  - `template<typename _Tp >  
_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
  - `template<typename _Tp >  
_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type > operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
  - `template<typename _Iterator >  
bool operator< (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
  - `template<typename _IteratorL, typename _IteratorR >  
bool operator< (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
  - `template<typename _IteratorL, typename _IteratorR >  
bool operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool operator< (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`  
`set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`  
`_Tp, _Alloc > &__ly)`
- `template<class _T1, class _T2 >`  
`bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`  
`_Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool operator< (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _-`  
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __-`  
`fun< __less, typename _Dom1::value_type >::result_type > operator< (const`  
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_-`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`  
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const`  
`typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_-`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`  
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const`  
`valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Seq >`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`  
`&__y)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator< (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type > operator< (const _Tp &__t, const valarray< _Tp > &__v)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`bool operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`_CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Setiosflags __f)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine,`  
`__k > &__x)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`

- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`  
`operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_-`  
`type >::result_type > operator<< (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _-`  
`CharT, _Traits > &__os, const piecewise\_constant\_distribution< _RealType >`  
`&__x)`
- `template<typename _CharT , typename _Traits , typename _Tp >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _-`  
`Traits > &&__os, const _Tp &__x)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _-`  
`CharT, _Traits > &__os, const student\_t\_distribution< _RealType > &__x)`
- `template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>`  
`std::basic\_ostream< _Ch, _Tr > & operator<< (std::basic\_ostream< _Ch, _Tr`  
`> &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _-`  
`CharT, _Traits > &__os, const normal\_distribution< _RealType > &__x)`
- `template<typename _CharT , typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT,`  
`_Traits > &__os, const error\_code &__e)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _-`  
`CharT, _Traits > &, const std::uniform\_real\_distribution< _RealType > &)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _-`  
`CharT, _Traits > &, const std::weibull\_distribution< _RealType > &)`
- `template<typename _CharT , typename _Traits , typename _Alloc >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _-`  
`Traits > &__os, const basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > operator<< (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > operator<< (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`  
`typename > class _Base>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _  
Traits > &__os, const gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _  
Base > &__str)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`  
`type >::result_type > operator<< (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &`  
`__v)`
- `template<class _CharT, class _Traits >`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT,`  
`_Traits > &__out, thread::id __id)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _  
CharT, _Traits > &__os, const binomial\_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream<`  
`\_CharT, _Traits > &__os, const piecewise\_linear\_distribution< _RealType >`  
`&__x)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`  
`, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_`  
`ostream< _CharT, _Traits > &__os, const std::independent\_bits\_engine<`  
`\_RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _`  
`CharT, _Traits > &__os, const poisson\_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream<`  
`\_CharT, _Traits > &__os, const negative\_binomial\_distribution< _IntType >`  
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _`  
`CharT, _Traits > &, const std::geometric\_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _`  
`CharT, _Traits > &__os, const gamma\_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream< _-`  
`CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`  
`UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream<`  
`_CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w,`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`  
`_Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream< _-`  
`CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`  
`_Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_-`  
`ostream< _CharT, _Traits > &__os, const discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _-`  
`Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > operator<<< (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > operator<<< (const valarray< _Tp`  
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > operator<<< (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream< _-`  
`CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream< _-`  
`CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`  
`, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream<`  
`_CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a,`  
`__c, __m > &__lcr)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`  
`_Traits > & __os, _Resetiosflags __f)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator<= (const reverse_iterator< _IteratorL > & __x, const reverse_`  
`iterator< _IteratorR > & __y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator<= (const move_iterator< _IteratorL > & __x, const move_`  
`iterator< _IteratorR > & __y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc >`  
`& __y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > & __x, const`  
`map< _Key, _Tp, _Compare, _Alloc > & __y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > & __y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > & __x, const mul-`  
`tiset< _Key, _Compare, _Alloc > & __y)`
- `template<class _T1, class _T2 >`  
`bool operator<= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq >`  
`& __y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const set< _Key, _Compare, _Alloc > & __x, const set< _`  
`Key, _Compare, _Alloc > & __y)`
- `template<typename _Iterator >`  
`bool operator<= (const reverse_iterator< _Iterator > & __x, const reverse_`  
`iterator< _Iterator > & __y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`  
`bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _`  
`Alloc > & __x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`  
`> & __y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > operator<= (const _Tp & __t, const`  
`valarray< _Tp > & __v)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator<= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename _Dom1::value_type >::result_type >`  
`operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`  
`operator<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`  
`operator<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`  
`operator<= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename... _TElements, typename... _UElements>`  
`bool operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`  
`operator<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `bool operator<= (thread::id __x, thread::id __y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator<= (const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator<= (const \_Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const \_Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< \_BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp >`  
`_Expr< \_BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp >`  
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _Iterator >`  
`bool operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _Tp >`  
`bool operator== (const \_Fwd\_list\_iterator< _Tp > &__x, const \_Fwd\_list\_const\_iterator< _Tp > &__y)`
- `template<typename _IntType >`  
`bool operator== (const std::uniform\_int\_distribution< _IntType > &__d1, const std::uniform\_int\_distribution< _IntType > &__d2)`
- `template<typename _Val >`  
`bool operator== (const \_List\_iterator< _Val > &__x, const \_List\_const\_iterator< _Val > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`bool operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _RealType >`  
`bool operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator== (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc , bool __cache_hash_code>`  
`bool operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Res , typename... _Args>`  
`bool operator== (_M_clear_type *, const function< _Res(_Args...)> &__f)`

- `template<typename _IntType >`  
`bool operator== (const std::discrete\_distribution< _IntType > &__d1, const std::discrete\_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator== (const std::extreme\_value\_distribution< _RealType > &__d1, const std::extreme\_value\_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const _CharT * __lhs, const basic\_string< _CharT, _Traits, _Alloc > & __rhs)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `bool operator== (const std::bernoulli\_distribution &__d1, const std::bernoulli\_distribution &__d2)`
- `template<typename _StateT >`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Val >`  
`bool operator== (const \_Rb\_tree\_iterator< _Val > &__x, const \_Rb\_tree\_iterator< _Val > &__y)`
- `template<class _T1, class _T2 >`  
`bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _IntType >`  
`bool operator== (const std::uniform\_real\_distribution< _IntType > &__d1, const std::uniform\_real\_distribution< _IntType > &__d2)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code >`  
`bool operator== (const \_\_unordered\_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const \_\_unordered\_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator==(const vector<_Tp, _Alloc > &__x, const vector<_Tp, _Alloc > &__y)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr<_BinClos<__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun<__equal_to, typename _Dom1::value_type >::result_type > operator==(const _Expr<_Dom1, typename _Dom1::value_type > &__v, const _Expr<_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr<_BinClos<__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _fun<__equal_to, typename _Dom::value_type >::result_type > operator==(const typename _Dom::value_type &__t, const _Expr<_Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr<_BinClos<__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _fun<__equal_to, typename _Dom::value_type >::result_type > operator==(const _Expr<_Dom, typename _Dom::value_type > &__e, const valarray<typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits >`  
`bool operator==(const istreambuf_iterator<_CharT, _Traits > &__a, const istreambuf_iterator<_CharT, _Traits > &__b)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool operator==(const istream_iterator<_Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator<_Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _RealType >`  
`bool operator==(const std::weibull_distribution<_RealType > &__d1, const std::weibull_distribution<_RealType > &__d2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator==(const shared_ptr<_Tp1 > &__a, const shared_ptr<_Tp2 > &__b)`
- `template<typename... _TElements, typename... _UElements >`  
`bool operator==(const tuple<_TElements...> &__t, const tuple<_UElements...> &__u)`
- `template<typename _RealType >`  
`bool operator==(const std::normal_distribution<_RealType > &__d1, const std::normal_distribution<_RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator==(const std::cauchy_distribution<_RealType > &__d1, const std::cauchy_distribution<_RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator==(const std::exponential_distribution<_RealType > &__d1, const std::exponential_distribution<_RealType > &__d2)`
- `template<typename _IntType >`  
`bool operator==(const std::geometric_distribution<_IntType > &__d1, const std::geometric_distribution<_IntType > &__d2)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool operator==(const __unordered_multiset< _Value, _Hash, _Pred, _Alloc,`  
`__cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _`  
`Pred, _Alloc, __cache_hash_code > &__y)`
- `bool operator==(const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator==(const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator==(const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`  
`> &__y)`
- `template<typename _T1, typename _T2 >`  
`bool operator==(const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Res, typename... _Args>`  
`bool operator==(const function< _Res(_Args...)> &__f, _M_clear_type *)`
- `template<typename _RealType >`  
`bool operator==(const std::piecewise_constant_distribution< _RealType > &_`  
`_d1, const std::piecewise_constant_distribution< _RealType > &_d2)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator==(const multiset< _Key, _Compare, _Alloc > &__x, const mul-`  
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > operator==(const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator==(const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`  
`&__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`bool operator==(const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _`  
`Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp,`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator==(const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _`  
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > operator==(const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > operator==(const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `bool operator==(const error_code &__lhs, const error_code &__rhs)`

- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type operator== (const basic\_string< _CharT > &__lhs, const basic\_string< _CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator== (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator> (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< __greater, typename _Dom1::value_type >::result_type > operator>`

- (const `_Expr`< `_Dom1`, `typename _Dom1::value_type` > &\_\_v, const `_Expr`< `_Dom2`, `typename _Dom2::value_type` > &\_\_w)
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator>` (const `_Deque_iterator`< `_Tp, _RefL, _PtrL` > &\_\_x, const `_Deque_iterator`< `_Tp, _RefR, _PtrR` > &\_\_y)
  - `template<class _Dom >`  
`_Expr`< `_BinClos`< `__greater, _Expr, _Constant, _Dom, typename _Dom::value_type` >, `typename __fun`< `__greater, typename _Dom::value_type >::result_type` > `operator>` (const `_Expr`< `_Dom, typename _Dom::value_type` > &\_\_v, const `typename _Dom::value_type` &\_\_t)
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator>` (const `_Rb_tree`< `_Key, _Val, _KeyOfValue, _Compare, _Alloc` > &\_\_x, const `_Rb_tree`< `_Key, _Val, _KeyOfValue, _Compare, _Alloc` > &\_\_y)
  - `template<class _Dom >`  
`_Expr`< `_BinClos`< `__greater, _Constant, _Expr, typename _Dom::value_type, _Dom` >, `typename __fun`< `__greater, typename _Dom::value_type >::result_type` > `operator>` (const `typename _Dom::value_type` &\_\_t, const `_Expr`< `_Dom, typename _Dom::value_type` > &\_\_v)
  - `template<class _T1, class _T2 >`  
`bool operator>` (const `pair`< `_T1, _T2` > &\_\_x, const `pair`< `_T1, _T2` > &\_\_y)
  - `template<class _Dom >`  
`_Expr`< `_BinClos`< `__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type` >, `typename __fun`< `__greater, typename _Dom::value_type >::result_type` > `operator>` (const `_Expr`< `_Dom, typename _Dom::value_type` > &\_\_e, const `valarray`< `typename _Dom::value_type` > &\_\_v)
  - `template<typename _Iterator >`  
`bool operator>` (const `reverse_iterator`< `_Iterator` > &\_\_x, const `reverse_iterator`< `_Iterator` > &\_\_y)
  - `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator>` (const `unique_ptr`< `_Tp, _Tp_Deleter` > &\_\_x, const `unique_ptr`< `_Up, _Up_Deleter` > &\_\_y)
  - `template<typename _Tp, typename _Alloc >`  
`bool operator>` (const `deque`< `_Tp, _Alloc` > &\_\_x, const `deque`< `_Tp, _Alloc` > &\_\_y)
  - `template<class _Dom >`  
`_Expr`< `_BinClos`< `__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom` >, `typename __fun`< `__greater, typename _Dom::value_type >::result_type` > `operator>` (const `valarray`< `typename _Dom::value_type` > &\_\_v, const `_Expr`< `_Dom, typename _Dom::value_type` > &\_\_e)
  - `template<typename _Tp, typename _Alloc >`  
`bool operator>` (const `vector`< `_Tp, _Alloc` > &\_\_x, const `vector`< `_Tp, _Alloc` > &\_\_y)

- `bool operator> (thread::id __x, thread::id __y)`
- `template<typename... _TElements, typename... _UElements>  
bool operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _IteratorL, typename _IteratorR >  
bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >  
bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >  
bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp, typename _Alloc >  
bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >  
bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >  
bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Iterator >`  
`bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter >`  
`bool operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _T1, class _T2 >`  
`bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type >`  
`operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type >`  
`operator>= (const typename _`

```

Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type >
&_v)
• template<class _Dom >
  _Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename
  _Dom::value_type >, typename __fun< __greater_equal, typename _-
  Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-
  name _Dom::value_type > &_e, const valarray< typename _Dom::value_type
  > &_v)
• bool operator>= (thread::id __x, thread::id __y)
• template<class _Dom >
  _Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename
  _Dom::value_type >, typename __fun< __greater_equal, typename _-
  Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-
  name _Dom::value_type > &_v, const typename _Dom::value_type &_t)
• template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , type-
  name _Alloc >
bool operator>= (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _-
  Alloc > &_x, const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc
  > &_y)
• template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &_x, const
map< _Key, _Tp, _Compare, _Alloc > &_y)
• template<typename... _TElements, typename... _UElements>
bool operator>= (const tuple< _TElements...> &_t, const tuple< _-
  UElements...> &_u)
• template<class _Dom >
  _Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _-
  Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _-
  Dom::value_type >::result_type > operator>= (const valarray< typename _-
  Dom::value_type > &_v, const _Expr< _Dom, typename _Dom::value_type
  > &_e)
• template<typename _Tp , typename _Seq >
bool operator>= (const stack< _Tp, _Seq > &_x, const stack< _Tp, _Seq >
  &_y)
• template<typename _Tp >
  _Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-
  name __fun< __greater_equal, _Tp >::result_type > operator>= (const _Tp
  &_t, const valarray< _Tp > &_v)
• template<typename _Tp , typename _Seq >
bool operator>= (const queue< _Tp, _Seq > &_x, const queue< _Tp, _Seq >
  &_y)
• template<typename _Tp , typename _Alloc >
bool operator>= (const forward\_list< _Tp, _Alloc > &_lx, const forward\_-
list< _Tp, _Alloc > &_ly)

```

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator>= (const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, shuffle\_order\_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_Setiosflags __f)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename > class _Base>`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_Get\_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_Setfill< _CharT > __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &, std::uniform\_int\_distribution< _IntType > &)`
- `template<class _Dom >`  
`\_Expr< \_BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, \_Setbase __f)`

- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`  
`operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, student\_t\_distribution< _RealType > &__x)`
- `template<typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, std::bernoulli\_distribution &__x)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &, std::uniform\_real\_distribution< _RealType > &)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, chi\_squared\_distribution< _RealType > &__x)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, gamma\_distribution< _RealType > &__x)`
- `template<typename _RealType , typename _CharT , typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &, std::cauchy\_distribution< _RealType > &)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`  
`type >::result_type > operator>> (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`  
`>::result_type > operator>> (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`  
`>::result_type > operator>> (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp , typename _CharT , class _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _`  
`Traits > &__is, complex< _Tp > &__x)`
- `template<typename _CharT , typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _`  
`Traits > &__is, \_Resetiosflags __f)`

- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, negative\_binomial\_distribution< _IntType > &__x)`
- `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__is, basic\_`  
`string< char > &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, piecewise\_constant\_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &, std::geometric\_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, piecewise\_linear\_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`  
`Traits > &__is, \_Setw __f)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _`  
`UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, mersenne\_twister\_engine< _UIntType, __w, __n, __m,`  
`__r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &, std::exponential\_distribution< _RealType > &)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`  
`_Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, subtract\_with\_carry\_engine< _UIntType, __w, __s, __r`  
`> &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, poisson\_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`  
`_Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, discard\_block\_engine< _RandomNumberEngine, __p,`  
`__r > &__x)`
- `template<typename _CharT, typename _Traits, typename _Tp >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`  
`Traits > &&__is, _Tp &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > & __is, fisher_f_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > & __is, discrete_distribution< _IntType > & __x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp`  
`> & __v, const valarray< _Tp > & __w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp`  
`> & __v, const _Tp & __t)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > & __is, binomial_distribution< _IntType > & __x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __shift_right, _Tp >::result_type > operator>> (const _Tp & _-`  
`_t, const valarray< _Tp > & __v)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > & __is, normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > & __is, lognormal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`  
`>::result_type > operator^ (const typename _Dom::value_type & __t, const _`  
`Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`  
`_Tp > & __v, const valarray< _Tp > & __w)`
- `_Ios_Iostate operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`

- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_`  
`type >::result_type > operator^ (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`  
`>::result_type > operator^ (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `_Ios_Openmode operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_`  
`type >::result_type > operator^ (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __bitwise_xor, typename _Dom1::value_type >::result_type >`  
`operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `_Ios_Openmode & operator^ = (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator^ = (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & operator^ = (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`  
`>::result_type > operator| (const typename _Dom::value_type &__t, const _`  
`Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __bitwise_or, typename _Dom1::value_type >::result_type >`  
`operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_`  
`type >::result_type > operator| (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `_Ios_Iostate operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`  
`>::result_type > operator| (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_`  
`type >::result_type > operator| (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `_Ios_Openmode operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > operator| (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `_Ios_Fmtflags & operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > operator|| (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`  
`operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > operator|| (const typename _Dom::value_type &__t, const _`  
`Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > operator|| (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > operator|| (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > operator|| (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `_Ios_Fmtflags operator~ (_Ios_Fmtflags __a)`
- `_Ios_Openmode operator~ (_Ios_Openmode __a)`
- `_Ios_Iostate operator~ (_Ios_Iostate __a)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`  
`void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`  
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`

- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator partial\_sort\_copy (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`  
`result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator partial\_sort\_copy (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`  
`result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial\_sum (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator partial\_sum (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _-`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`  
`name _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > partition\_copy (_InputIterator __-`  
`first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`  
`out_false, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _-`  
`Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator partition\_point (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _Tp >`  
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _RAIter >`  
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _`  
`Dom >, typename _Dom::value_type > pow (const typename _Dom::value_`  
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _`  
`Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_`  
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow`  
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `float pow (float __x, float __y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`  
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`  
`>::__type, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow`  
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`  
`Dom >, typename _Dom::value_type > pow (const valarray< typename _`  
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow`  
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _BidirectionalIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _BidirectionalIterator >::__value, _-`  
`BidirectionalIterator >::__type prev (_BidirectionalIterator __x, typename`  
`iterator_traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BIter >`  
`bool prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`last, _Compare __comp)`
- `template<typename _Tp >`  
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp __x)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__-`  
`x)(_Arg1, _Arg2))`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RAIter >`  
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __-`  
`intl=false)`
- `template<typename _RAIter, typename _Generator >`  
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _RandomNumberGenerator &&__rand)`
- `template<typename _RandomAccessIterator >`  
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`

- `template<typename _RAIter >`  
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _Tp >`  
`_Tp real (const complex< _Tp > &__z)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _FIter, typename _Tp >`  
`_FIter remove (_FIter, _FIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, const _Tp &__value)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _Predicate __pred)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`  
`_OIter remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last,`  
`_Predicate __pred)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`  
`old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter replace_copy (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _-`  
`Tp >`  
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter replace_copy_if (_IIter, _IIter, _OIter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate _-`  
`__pred, const _Tp &__new_value)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &)`

- `_Resetiosflags` `resetiosflags` (`ios_base::fmtflags __mask`)
- `void rethrow_exception` (`exception_ptr`) `__attribute__((__noreturn__))`
- `void rethrow_if_nested` (`const nested_exception &__ex`)
- `template<typename _Ex >`  
`void rethrow_if_nested` (`const _Ex &__ex`)
- `template<typename _Tp >`  
`void return_temporary_buffer` (`_Tp *__p`)
- `template<typename _BidirectionalIterator >`  
`void reverse` (`_BidirectionalIterator __first, _BidirectionalIterator __last`)
- `template<typename _BIter >`  
`void reverse` (`_BIter, _BIter`)
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator reverse_copy` (`_BidirectionalIterator __first, _-`  
`_BidirectionalIterator __last, _OutputIterator __result`)
- `template<typename _BIter, typename _OIter >`  
`_OIter reverse_copy` (`_BIter, _BIter, _OIter`)
- `ios_base & right` (`ios_base &__base`)
- `template<typename _FIter >`  
`void rotate` (`_FIter, _FIter, _FIter`)
- `template<typename _ForwardIterator >`  
`void rotate` (`_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last`)
- `template<typename _FIter, typename _OIter >`  
`_OIter rotate_copy` (`_FIter, _FIter, _FIter, _OIter`)
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator rotate_copy` (`_ForwardIterator __first, _ForwardIterator __-`  
`middle, _ForwardIterator __last, _OutputIterator __result`)
- `ios_base & scientific` (`ios_base &__base`)
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`  
`_FIter1 search` (`_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate`)
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 search` (`_ForwardIterator1 __first1, _ForwardIterator1 __-`  
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`  
`__predicate`)
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 search` (`_ForwardIterator1 __first1, _ForwardIterator1 __-`  
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2`)
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search` (`_FIter1, _FIter1, _FIter2, _FIter2`)
- `template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`_FIter search_n` (`_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate`)
- `template<typename _FIter, typename _Size, typename _Tp >`  
`_FIter search_n` (`_FIter, _FIter, _Size, const _Tp &`)

- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _-`  
`Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _-`  
`BinaryPredicate >`  
`_ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _-`  
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter set\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`  
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter set\_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set\_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `new\_handler set\_new\_handler (new\_handler) throw ()`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set\_symmetric\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter set\_symmetric\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`  
`_OutputIterator __result)`

- [terminate\\_handler](#) [set\\_terminate](#) ([terminate\\_handler](#)) throw ()
- [unexpected\\_handler](#) [set\\_unexpected](#) ([unexpected\\_handler](#)) throw ()
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- [\\_Setbase](#) [setbase](#) (int \_\_base)
- `template<typename _CharT >`  
`_Setfill< _CharT > setfill (_CharT __c)`
- [\\_Setiosflags](#) [setiosflags](#) ([ios\\_base::fmtflags](#) \_\_mask)
- [\\_Setprecision](#) [setprecision](#) (int \_\_n)
- [\\_Setw](#) [setw](#) (int \_\_n)
- [ios\\_base](#) & [showbase](#) ([ios\\_base](#) &\_\_base)
- [ios\\_base](#) & [showpoint](#) ([ios\\_base](#) &\_\_base)
- [ios\\_base](#) & [showpos](#) ([ios\\_base](#) &\_\_base)
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`  
`UniformRandomNumberGenerator &__g)`
- `template<typename _RAIter, typename _UGenerator >`  
`void shuffle (_RAIter, _RAIter, _UGenerator &)`
- long double [sin](#) (long double \_\_x)
- float [sin](#) (float \_\_x)
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp >`  
`&__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`complex< _Tp > sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin`  
`(_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > sinh (const complex< _Tp > &)`

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`sinh (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp`  
`> &__v)`
- `long double sinh (long double __x)`
- `float sinh (float __x)`
- `ios_base & skipws (ios_base & __base)`
- `template<typename _RAIter >`  
`void sort (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _`  
`Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter >`  
`void sort_heap (_RAIter, _RAIter)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp`  
`> &__v)`
- `long double sqrt (long double __x)`
- `float sqrt (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`sqrt (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > sqrt (const complex< _Tp > &)`

- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator stable\_partition (_ForwardIterator __first, _ForwardIterator __-`  
`__last, _Predicate __pred)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter stable\_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter >`  
`void stable\_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void stable\_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`__last)`
- `template<typename _Tp, typename _Tp1 >`  
`shared\_ptr<_Tp > static\_pointer\_cast (const shared\_ptr<_Tp1 > &__r)`
- `double stof (const string &__str, size_t *__idx=0)`
- `float stof (const string &__str, size_t *__idx=0)`
- `int stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double stold (const string &__str, size_t *__idx=0)`
- `long long stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long stoull (const string &__str, size_t *__idx=0, int __base=10)`
- `char * strchr (char *__s, int __n)`
- `char * strpbrk (char *__s1, const char *__s2)`
- `char * strrchr (char *__s, int __n)`
- `char * strstr (char *__s1, const char *__s2)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_-`  
`code>`  
`void swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __-`  
`cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp >`  
`void swap (_Tp &__a, _Tp &__b)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _-`  
`Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Mutex >`  
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_-`  
`code>`

- ```
void swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_-
hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __-
cache_hash_code > &__y)
```
- `template<typename... _Elements>`  
`void swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
  - `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _-`  
`CharT, _Traits, _Alloc > &__rhs)`
  - `template<typename _Res >`  
`void swap (promise< _Res > &__x, promise< _Res > &__y)`
  - `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _-`  
`Compare, _Alloc > &__y)`
  - `template<typename _Tp, typename _Seq >`  
`void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`
  - `template<typename _Res, typename... _Args>`  
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__-`  
`y)`
  - `template<typename _Tp, size_t _Nm>`  
`void swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
  - `template<typename _Tp, typename _Sequence, typename _Compare >`  
`void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_-`  
`queue< _Tp, _Sequence, _Compare > &__y)`
  - `void swap (thread &__x, thread &__y)`
  - `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
  - `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
  - `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
  - `template<typename _Tp >`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
  - `template<typename _Tp >`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
  - `template<class _T1, class _T2 >`  
`void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)`
  - `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`
  - `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Res, typename... _ArgTypes>`  
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Tp_Deleter >`  
`void swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _Tp_Deleter > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `_GLIBCXX_CONST const error_category & system_category () throw ()`
- long double **tan** (long double \_\_x)
- float **tan** (float \_\_x)

- `template<typename _Tp >  
_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `template<typename _Tp >  
complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp >  
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tan  
(_Tp __x)`
- `template<class _Dom >  
_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan  
(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
complex< _Tp > tanh (const complex< _Tp > &)`
- `template<typename _Tp >  
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type  
tanh (_Tp __x)`
- `long double tanh (long double __x)`
- `template<class _Dom >  
_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >  
tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float tanh (float __x)`
- `template<typename _Tp >  
_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp >  
&__v)`
- `void terminate () __attribute__((__noreturn__)) throw ()`
- `const _Rb_tree_node_base *__root throw ()`
- `template<typename _Ex >  
void throw\_with\_nested (_Ex __ex)`
- `template<typename... _Elements >  
tuple< _Elements &...> tie (_Elements &...__args)`
- `string to\_string (long double __val)`
- `string to\_string (unsigned long __val)`
- `string to\_string (float __val)`
- `string to\_string (long long __val)`
- `string to\_string (long __val)`
- `string to\_string (int __val)`
- `string to\_string (unsigned __val)`
- `string to\_string (unsigned long long __val)`
- `string to\_string (double __val)`
- `wstring to\_wstring (unsigned __val)`
- `wstring to\_wstring (unsigned long __val)`
- `wstring to\_wstring (int __val)`
- `wstring to\_wstring (unsigned long long __val)`
- `wstring to\_wstring (double __val)`

- [wstring to\\_wstring](#) (long long \_\_val)
- [wstring to\\_wstring](#) (long double \_\_val)
- [wstring to\\_wstring](#) (float \_\_val)
- [wstring to\\_wstring](#) (long \_\_val)
- `template<typename _CharT >`  
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter , typename _OIter , typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1 , typename _Iter2 , typename _OIter , typename _BinaryOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , type-`  
`name _BinaryOperation >`  
`_OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`  
`op)`
- `template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >`  
`_OutputIterator transform (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Lock1 , typename _Lock2 , typename... _Lock3 >`  
`int try\_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`
- `template<typename... _TElements, typename... _UElements >`  
`tuple< _TElements..., _UElements...> tuple\_cat (const tuple< _TElements...>`  
`&__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements >`  
`tuple< _TElements..., _UElements...> tuple\_cat (const tuple< _TElements...>`  
`&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements >`  
`tuple< _TElements..., _UElements...> tuple\_cat (tuple< _TElements...>`  
`&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements >`  
`tuple< _TElements..., _UElements...> tuple\_cat (tuple< _TElements...>`  
`&&__t, const tuple< _UElements...> &__u)`
- `bool uncaught\_exception () __attribute__((__pure__)) throw ()`
- `void unexpected () __attribute__((__noreturn__))`
- `template<typename _InputIterator , typename _ForwardIterator >`  
`_ForwardIterator uninitialized\_copy (_InputIterator __first, _InputIterator __-`  
`last, _ForwardIterator __result)`
- `template<typename _InputIterator , typename _Size , typename _ForwardIterator >`  
`_ForwardIterator uninitialized\_copy\_n (_InputIterator __first, _Size __n, _-`  
`ForwardIterator __result)`
- `template<typename _ForwardIterator , typename _Tp >`  
`void uninitialized\_fill (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__x)`

- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`void uninitialized\_fill\_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _Filter >`  
`_Filter unique (_Filter, _Filter)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _IIter, typename _OIter, typename _BinaryPredicate >`  
`_OIter unique\_copy (_IIter, _IIter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator unique\_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator unique\_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result)`
- `template<typename _IIter, typename _OIter >`  
`_OIter unique\_copy (_IIter, _IIter, _OIter)`
- `ios\_base & unitbuf (ios\_base &__base)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator upper\_bound (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__val, _Compare __comp)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter upper\_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter upper\_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator upper\_bound (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__val)`
- `ios\_base & uppercase (ios\_base &__base)`
- `template<typename _Facet >`  
`const _Facet & use\_facet (const locale &__loc)`
- `wchar_t * wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & ws (basic\_istream< _CharT, _Traits > &__-`  
`is)`

- `template<size_t _Nb>`  
`bitset<_Nb> operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
  
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
  
- `template<typename _Tp>`  
`complex<_Tp> operator+ (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator+ (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator+ (const _Tp &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`complex<_Tp> operator- (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator- (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator- (const _Tp &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`complex<_Tp> operator* (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator* (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator* (const _Tp &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`complex<_Tp> operator/ (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`complex<_Tp> operator/ (const complex<_Tp> &__x, const _Tp &__y)`

- `template<typename _Tp >`  
`complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > ref (_Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > cref (const _Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t)`
- `template<typename _CharT , typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<typename _CharT , typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`

- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

## Variables

- `enable_if< (!is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type _invoke (_Functor &__f, _Args &&...__args)`
- `static ios_base::Init __iointit`
- `function< void()> __once_functor`
- `std::binder1st _GLIBCXX_DEPRECATED_ATTR`

- const [adopt\\_lock\\_t](#) **adopt\_lock**
- const [defer\\_lock\\_t](#) **defer\_lock**
- const [error\\_category](#) \*const [future\\_category](#)
- [error\\_code](#) **make\_error\_code** (errc)
- [error\\_condition](#) **make\_error\_condition** (errc)
- const [nothrow\\_t](#) **nothrow**
- const [try\\_to\\_lock\\_t](#) **try\_to\_lock**

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the *HOWTO* linked to above.

- [istream](#) `cin`
- [ostream](#) `cout`
- [ostream](#) `cerr`
- [ostream](#) `clog`
- [wistream](#) `wcin`
- [wostream](#) `wcout`
- [wostream](#) `wcerr`
- [wostream](#) `wclog`

## 4.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

## 4.11.2 Typedef Documentation

### 4.11.2.1 typedef void(\* std::new\_handler)()

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 75 of file `new`.

### 4.11.2.2 typedef long long std::streamoff

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 94 of file postypes.h.

#### 4.11.2.3 `typedef fpos<mbstate_t> std::streampos`

File position for char streams.

Definition at line 228 of file postypes.h.

#### 4.11.2.4 `typedef ptrdiff_t std::streamsize`

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file postypes.h.

#### 4.11.2.5 `typedef fpos<mbstate_t> std::u16streampos`

File position for char16\_t streams.

Definition at line 234 of file postypes.h.

#### 4.11.2.6 `typedef fpos<mbstate_t> std::u32streampos`

File position for char32\_t streams.

Definition at line 236 of file postypes.h.

#### 4.11.2.7 `typedef fpos<mbstate_t> std::wstreampos`

File position for wchar\_t streams.

Definition at line 230 of file postypes.h.

### 4.11.3 Enumeration Type Documentation

#### 4.11.3.1 anonymous enum

##### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more. This controls some aspect of the sort routines.

Definition at line 2167 of file stl\_algo.h.

### 4.11.3.2 enum `std::float_denorm_style`

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the `std::numeric_limits` class.

#### Enumerator:

*denorm\_indeterminate* Indeterminate at compile time whether denormalized values are allowed.

*denorm\_absent* The type does not allow denormalized values.

*denorm\_present* The type allows denormalized values.

Definition at line 170 of file `limits`.

### 4.11.3.3 enum `std::float_round_style`

Describes the rounding style for floating-point types.

This is used in the `std::numeric_limits` class.

#### Enumerator:

*round\_indeterminate* Self-explanatory.

*round\_toward\_zero* Self-explanatory.

*round\_to\_nearest* To the nearest representable value.

*round\_toward\_infinity* Self-explanatory.

*round\_toward\_neg\_infinity* Self-explanatory.

Definition at line 155 of file `limits`.

## 4.11.4 Function Documentation

### 4.11.4.1 `template<typename _RandomAccessIterator > void std::__final_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2172 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

**4.11.4.2** `template<typename _RandomAccessIterator, typename _Compare >  
void std::__final_insertion_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2187 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

**4.11.4.3** `template<typename _InputIterator, typename _Tp > _InputIterator  
std::__find ( _InputIterator __first, _InputIterator __last, const _Tp  
& __val, input_iterator_tag ) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 128 of file `stl_algo.h`.

Referenced by `find()`.

**4.11.4.4** `template<typename _RandomAccessIterator, typename _Tp >  
_RandomAccessIterator std::__find ( _RandomAccessIterator  
__first, _RandomAccessIterator __last, const _Tp & __val,  
random_access_iterator_tag )`

This is an overload used by `find()` for the RAI case.

Definition at line 150 of file `stl_algo.h`.

**4.11.4.5** `template<typename _InputIterator, typename _Predicate >  
_InputIterator std::__find_if ( _InputIterator __first, _InputIterator  
__last, _Predicate __pred, input_iterator_tag ) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 139 of file `stl_algo.h`.

Referenced by `find_if()`.

**4.11.4.6** `template<typename _RandomAccessIterator, typename _Predicate >  
_RandomAccessIterator std::__find_if ( _RandomAccessIterator  
__first, _RandomAccessIterator __last, _Predicate __pred,  
random_access_iterator_tag )`

This is an overload used by `find_if()` for the RAI case.

Definition at line 198 of file `stl_algo.h`.

```
4.11.4.7 template<typename _InputIterator , typename _Predicate >
    _InputIterator std::__find_if_not ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred, input_iterator_tag )
    [inline]
```

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 247 of file `stl_algo.h`.

Referenced by `find_if_not()`.

```
4.11.4.8 template<typename _RandomAccessIterator , typename _Predicate >
    _RandomAccessIterator std::__find_if_not ( _RandomAccessIterator
    __first, _RandomAccessIterator __last, _Predicate __pred,
    random_access_iterator_tag )
```

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 258 of file `stl_algo.h`.

```
4.11.4.9 template<typename _EuclideanRingElement >
    _EuclideanRingElement std::__gcd ( _EuclideanRingElement __m,
    _EuclideanRingElement __n )
```

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1487 of file `stl_algo.h`.

```
4.11.4.10 template<typename _RandomAccessIterator > void
    std::__heap_select ( _RandomAccessIterator __first,
    _RandomAccessIterator __middle, _RandomAccessIterator __last )
```

This is a helper function for the sort routines.

Definition at line 1895 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

```
4.11.4.11 template<typename _RandomAccessIterator , typename _Compare
    > void std::__heap_select ( _RandomAccessIterator __first,
    _RandomAccessIterator __middle, _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the sort routines.

Definition at line 1908 of file `stl_algo.h`.

References `make_heap()`.

**4.11.4.12** `template<typename _ForwardIterator, typename  
_Predicate, typename _Distance > _ForwardIterator  
std::__inplace_stable_partition ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, _Distance __len )`

This is a helper function...

Definition at line 1772 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

**4.11.4.13** `template<typename _RandomAccessIterator > void  
std::__inplace_stable_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last )`

This is a helper function for the stable sorting routines.

Definition at line 3353 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

**4.11.4.14** `template<typename _RandomAccessIterator, typename _Compare  
> void std::__inplace_stable_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the stable sorting routines.

Definition at line 3372 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.

**4.11.4.15** `template<typename _RandomAccessIterator > void  
std::__insertion_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2095 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

**4.11.4.16** `template<typename _RandomAccessIterator, typename _Compare  
> void std::__insertion_sort ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2118 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

**4.11.4.17** `template<typename _RandomAccessIterator, typename  
_Size, typename _Compare > void std::__introsort_loop (   
_RandomAccessIterator __first, _RandomAccessIterator __last,  
_Size __depth_limit, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2289 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

**4.11.4.18** `template<typename _RandomAccessIterator, typename _Size  
> void std::__introsort_loop ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Size __depth_limit )`

This is a helper function for the sort routine.

Definition at line 2267 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

**4.11.4.19** `template<typename _Size > _Size std::__lg ( _Size __n )  
[inline]`

This is a helper function for the sort routines and for [random.tcc](#).

Definition at line 992 of file `stl_algbase.h`.

Referenced by `nth_element()`, `std::linear_congruential_engine< _UIntType, __a, __c,  
__m >::seed()`, and `sort()`.

**4.11.4.20** `template<typename _BidirectionalIterator , typename _Distance , typename _Pointer > void std::__merge_adaptive ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size )`

This is a helper function for the merge routines.

Definition at line 2829 of file `stl_algo.h`.

References `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

**4.11.4.21** `template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare > void std::__merge_adaptive ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp )`

This is a helper function for the merge routines.

Definition at line 2891 of file `stl_algo.h`.

References `__merge_adaptive()`, `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

**4.11.4.22** `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename _Compare > _BidirectionalIterator3 std::__merge_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp )`

This is a helper function for the merge routines.

Definition at line 2761 of file `stl_algo.h`.

**4.11.4.23** `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 > _BidirectionalIterator3 std::merge_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result )`

This is a helper function for the merge routines.

Definition at line 2726 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

**4.11.4.24** `template<typename _BidirectionalIterator , typename _Distance > void std::merge_without_buffer ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2 )`

This is a helper function for the merge routines.

Definition at line 2954 of file `stl_algo.h`.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

**4.11.4.25** `template<typename _BidirectionalIterator , typename _Distance , typename _Compare > void std::merge_without_buffer ( _BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp )`

This is a helper function for the merge routines.

Definition at line 2998 of file `stl_algo.h`.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

**4.11.4.26** `template<typename _Iterator > void std::__move_median_first ( _Iterator __a, _Iterator __b, _Iterator __c )`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 76 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

**4.11.4.27** `template<typename _Iterator , typename _Compare > void  
std::__move_median_first ( _Iterator __a, _Iterator __b, _Iterator  
__c, _Compare __comp )`

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__a`.

Definition at line 100 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.28** `template<typename _BidirectionalIterator , typename _Predicate  
> _BidirectionalIterator std::__partition ( _BidirectionalIterator  
__first, _BidirectionalIterator __last, _Predicate __pred,  
bidirectional_iterator_tag )`

This is a helper function...

Definition at line 1742 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.29** `template<typename _ForwardIterator , typename _Predicate >  
_ForwardIterator std::__partition ( _ForwardIterator __first,  
_ForwardIterator __last, _Predicate __pred, forward_iterator_tag  
)`

This is a helper function...

Definition at line 1717 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

**4.11.4.30** `template<typename _BidirectionalIterator > void std::__reverse  
( _BidirectionalIterator __first, _BidirectionalIterator __last,  
bidirectional_iterator_tag )`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1387 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, and `reverse()`.

**4.11.4.31** `template<typename _RandomAccessIterator > void std::__reverse (`  
`_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`random_access_iterator_tag )`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1407 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.32** `template<typename _BidirectionalIterator > void std::__rotate (`  
`_BidirectionalIterator __first, _BidirectionalIterator __middle,`  
`_BidirectionalIterator __last, bidirectional_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1537 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

**4.11.4.33** `template<typename _RandomAccessIterator > void std::__rotate (`  
`_RandomAccessIterator __first, _RandomAccessIterator __middle,`  
`_RandomAccessIterator __last, random_access_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1567 of file `stl_algo.h`.

References `iter_swap()`, `swap()`, and `swap_ranges()`.

**4.11.4.34** `template<typename _ForwardIterator > void std::__rotate`  
`( _ForwardIterator __first, _ForwardIterator __middle,`  
`_ForwardIterator __last, forward_iterator_tag )`

This is a helper function for the rotate algorithm.

Definition at line 1501 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_ -`  
`object()`, and `rotate()`.

**4.11.4.35** `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance > _BidirectionalIterator1 std::rotate_adaptive ( _BidirectionalIterator1 _first, _BidirectionalIterator1 _middle, _BidirectionalIterator1 _last, _Distance _len1, _Distance _len2, _BidirectionalIterator2 _buffer, _Distance _buffer_size )`

This is a helper function for the merge routines.

Definition at line 2797 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

**4.11.4.36** `template<typename _ForwardIterator , typename _Integer , typename _Tp > _ForwardIterator std::search_n ( _ForwardIterator _first, _ForwardIterator _last, _Integer _count, const _Tp & _val, std::forward_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 324 of file `stl_algo.h`.

Referenced by `search_n()`.

**4.11.4.37** `template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate > _ForwardIterator std::search_n ( _ForwardIterator _first, _ForwardIterator _last, _Integer _count, const _Tp & _val, _BinaryPredicate _binary_pred, std::forward_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 410 of file `stl_algo.h`.

**4.11.4.38** `template<typename _RandomAccessIter , typename _Integer , typename _Tp > _RandomAccessIter std::search_n ( _RandomAccessIter _first, _RandomAccessIter _last, _Integer _count, const _Tp & _val, std::random_access_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 356 of file `stl_algo.h`.

**4.11.4.39** `template<typename _RandomAccessIter , typename _Integer ,  
 typename _Tp , typename _BinaryPredicate > _RandomAccessIter  
 std::__search_n ( _RandomAccessIter __first, _RandomAccessIter  
 __last, _Integer __count, const _Tp & __val, _BinaryPredicate  
 __binary_pred, std::random_access_iterator_tag )`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 449 of file `stl_algo.h`.

**4.11.4.40** `template<typename _ForwardIterator , typename _Pointer ,  
 typename _Predicate , typename _Distance > _ForwardIterator  
 std::__stable_partition_adaptive ( _ForwardIterator __first,  
 _ForwardIterator __last, _Predicate __pred, _Distance __len,  
 _Pointer __buffer, _Distance __buffer_size )`

This is a helper function...

Definition at line 1797 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

**4.11.4.41** `template<typename _RandomAccessIterator , typename _Compare  
 > void std::__unguarded_insertion_sort ( _RandomAccessIterator  
 __first, _RandomAccessIterator __last, _Compare __comp )  
 [inline]`

This is a helper function for the sort routine.

Definition at line 2153 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

**4.11.4.42** `template<typename _RandomAccessIterator > void  
 std::__unguarded_insertion_sort ( _RandomAccessIterator __first,  
 _RandomAccessIterator __last ) [inline]`

This is a helper function for the sort routine.

Definition at line 2140 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

**4.11.4.43** `template<typename _RandomAccessIterator > void  
std::_unguarded_linear_insert ( _RandomAccessIterator __last )`

This is a helper function for the sort routine.

Definition at line 2058 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

**4.11.4.44** `template<typename _RandomAccessIterator , typename _Compare  
> void std::_unguarded_linear_insert ( _RandomAccessIterator  
__last, _Compare __comp )`

This is a helper function for the sort routine.

Definition at line 2076 of file `stl_algo.h`.

**4.11.4.45** `template<typename _RandomAccessIterator , typename  
_Tp > _RandomAccessIterator std::_unguarded_partition (   
_RandomAccessIterator __first, _RandomAccessIterator __last,  
const _Tp & __pivot )`

This is a helper function...

Definition at line 2203 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

**4.11.4.46** `template<typename _RandomAccessIterator , typename  
_Tp , typename _Compare > _RandomAccessIterator  
std::_unguarded_partition ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, const _Tp & __pivot, _Compare  
__comp )`

This is a helper function...

Definition at line 2223 of file `stl_algo.h`.

References `iter_swap()`.

**4.11.4.47** `template<typename _RandomAccessIterator >  
_RandomAccessIterator std::__unguarded_partition_pivot (  
_RandomAccessIterator __first, _RandomAccessIterator __last )  
[inline]`

This is a helper function...

Definition at line 2244 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

**4.11.4.48** `template<typename _RandomAccessIterator , typename _Compare  
> _RandomAccessIterator std::__unguarded_partition_pivot (  
_RandomAccessIterator __first, _RandomAccessIterator __last,  
_Compare __comp ) [inline]`

This is a helper function...

Definition at line 2256 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

**4.11.4.49** `template<typename _ForwardIterator , typename _OutputIterator  
> _OutputIterator std::__unique_copy ( _ForwardIterator  
__first, _ForwardIterator __last, _OutputIterator __result,  
forward_iterator_tag , output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1239 of file `stl_algo.h`.

Referenced by `unique_copy()`.

**4.11.4.50** `template<typename _InputIterator , typename _OutputIterator  
> _OutputIterator std::__unique_copy ( _InputIterator __first,  
_InputIterator __last, _OutputIterator __result, input_iterator_tag  
, output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1262 of file `stl_algo.h`.

**4.11.4.51** `template<typename _InputIterator , typename _OutputIterator ,  
typename _BinaryPredicate > _OutputIterator std::__unique_copy (  
_InputIterator __first, _InputIterator __last, _OutputIterator  
__result, _BinaryPredicate __binary_pred, input_iterator_tag ,  
output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1335 of file `stl_algo.h`.

**4.11.4.52** `template<typename _InputIterator , typename _ForwardIterator ,  
typename _BinaryPredicate > _ForwardIterator std::__unique_copy  
( _InputIterator __first, _InputIterator __last, _ForwardIterator  
__result, _BinaryPredicate __binary_pred, input_iterator_tag ,  
forward_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1364 of file `stl_algo.h`.

**4.11.4.53** `template<typename _InputIterator , typename _ForwardIterator  
> _ForwardIterator std::__unique_copy ( _InputIterator  
__first, _InputIterator __last, _ForwardIterator __result,  
input_iterator_tag , forward_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1285 of file `stl_algo.h`.

**4.11.4.54** `template<typename _ForwardIterator , typename _OutputIterator ,  
typename _BinaryPredicate > _OutputIterator std::__unique_copy ( _ForwardIterator  
__first, _ForwardIterator __last, _OutputIterator  
__result, _BinaryPredicate __binary_pred, forward_iterator_tag ,  
output_iterator_tag )`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1306 of file `stl_algo.h`.

**4.11.4.55** `template<typename _T1, typename _T2 > void std::_Construct (`  
`_T1 * __p, _T2 && __value ) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 73 of file `stl_construct.h`.

**4.11.4.56** `template<typename _Tp > void std::_Destroy ( _Tp * __pointer )`  
`[inline]`

Destroy the object pointed to by a pointer type.

Definition at line 88 of file `stl_construct.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_fill_initialize()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::reserve()`, and `std::vector< result_type >::~~vector()`.

**4.11.4.57** `template<typename _ForwardIterator > void std::_Destroy (`  
`_FowardIterator __first, _ForwardIterator __last ) [inline]`

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 118 of file `stl_construct.h`.

**4.11.4.58** `template<typename _InputIterator, typename _Tp > _Tp`  
`std::accumulate ( _InputIterator __first, _InputIterator __last, _Tp`  
`__init ) [inline]`

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is `init`. The values are processed in order.

#### Parameters

*first* Start of range.

*last* End of range.

*init* Starting value to add other values to.

#### Returns

The final sum.

Definition at line 116 of file stl\_numeric.h.

**4.11.4.59** `template<typename _InputIterator , typename _Tp , typename  
_BinaryOperation > _Tp std::accumulate ( _InputIterator __first,  
_InputIterator __last, _Tp __init, _BinaryOperation __binary_op )  
[inline]`

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object *binary\_op*. The initial value is *init*. The values are processed in order.

#### Parameters

*first* Start of range.

*last* End of range.

*init* Starting value to add other values to.

*binary\_op* Function object to accumulate with.

#### Returns

The final sum.

Definition at line 142 of file stl\_numeric.h.

**4.11.4.60** `template<typename _InputIterator , typename _OutputIterator >  
_OutputIterator std::adjacent_difference ( _InputIterator __first,  
_InputIterator __last, _OutputIterator __result )`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using operator-() and writes the result to *result*.

#### Parameters

*first* Start of input range.

*last* End of input range.

*result* Output to write sums to.

#### Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 312 of file stl\_numeric.h.

```
4.11.4.61 template<typename _InputIterator , typename _OutputIterator
, typename _BinaryOperation > _OutputIterator
std::adjacent_difference ( _InputIterator __first, _InputIterator
__last, _OutputIterator __result, _BinaryOperation __binary_op )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using the function object *binary\_op* and writes the result to *result*.

#### Parameters

*first* Start of input range.

*last* End of input range.

*result* Output to write sums to.

#### Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 354 of file `stl_numeric.h`.

```
4.11.4.62 template<typename _InputIterator , typename _Distance > void
std::advance ( _InputIterator & __i, _Distance __n ) [inline]
```

A generalization of pointer arithmetic.

#### Parameters

*i* An input iterator.

*n* The *delta* by which to change *i*.

#### Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, *n* may be negative, in which case *i* is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 169 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::M_range_initialize()`, `equal_range()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

**4.11.4.63** `template<typename _Result , typename _Functor ,  
typename... _ArgTypes> _Bind_result<_Result, typename  
_Maybe_wrap_member_pointer<_Functor>::type (_ArgTypes...)>  
std::bind ( _Functor __f, _ArgTypes... __args ) [inline]`

Function template for `std::bind`.

Definition at line 1402 of file `functional`.

**4.11.4.64** `ios_base& std::boolalpha ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 794 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noboolalpha()`.

**4.11.4.65** `template<typename _Tp > reference_wrapper<const _Tp> std::cref  
( const _Tp & __t ) [inline]`

Denotes a const reference should be taken to a variable.

Definition at line 445 of file `functional`.

Referenced by `cref()`.

**4.11.4.66** `template<typename _Tp > reference_wrapper<const _Tp> std::cref  
( reference_wrapper<_Tp > __t ) [inline]`

Partial specialization.

Definition at line 457 of file `functional`.

References `cref()`.

**4.11.4.67** `ios_base& std::dec ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 932 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by operator `>>()`.

**4.11.4.68** `template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance ( _InputIterator __first, _InputIterator __last ) [inline]`

A generalization of pointer arithmetic.

#### Parameters

*first* An input iterator.

*last* An input iterator.

#### Returns

The distance between them.

Returns `n` such that `first + n == last`. This requires that `last` must be reachable from `first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 111 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::_M_range_initialize()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `partition_point()`, `__gnu_cxx::random_sample_n()`, and `upper_bound()`.

**4.11.4.69** `template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::endl ( basic_ostream<_CharT, _Traits > & __os ) [inline]`

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this subject.

Definition at line 541 of file `ostream`.

References `flush()`, `std::basic_ostream<_CharT, _Traits>::put()`, and `std::basic_ios<_CharT, _Traits>::widen()`.

**4.11.4.70** `template<typename _CharT , typename _Traits >  
basic_ostream<_CharT, _Traits>& std::ends ( basic_ostream<  
_CharT, _Traits > & __os ) [inline]`

Write a null character into the output sequence.

*Null character* is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

Definition at line 552 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits >::put()`.

**4.11.4.71** `ios_base& std::fixed ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 957 of file `ios_base.h`.

References `std::ios_base::setf()`.

**4.11.4.72** `template<typename _CharT , typename _Traits >  
basic_ostream<_CharT, _Traits>& std::flush ( basic_ostream<  
_CharT, _Traits > & __os ) [inline]`

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 562 of file `ostream`.

References `std::basic_ostream<_CharT, _Traits >::flush()`.

Referenced by `endl()`.

**4.11.4.73** `template<typename _Tp > enable_if<!is_lvalue_reference<_  
_Tp>::value, _Tp&&>::type std::forward ( typename std::identity<  
_Tp >::type & __t ) [inline]`

forward (as per N2835) /// Forward lvalues as rvalues.

Definition at line 53 of file `move.h`.

**4.11.4.74** `template<typename _Tp > enable_if<!is_lvalue_reference<_  
_Tp>::value, _Tp&&>::type std::forward ( typename std::identity<  
_Tp >::type && __t ) [inline]`

Forward rvalues as rvalues.

Definition at line 59 of file move.h.

```
4.11.4.75 template<typename _MoneyT > _Get_money<_MoneyT>
std::get_money ( _MoneyT & __mon, bool __intl = false )
[inline]
```

Extended manipulator for extracting money.

#### Parameters

*mon* Either long double or a specialization of `basic_string`.

*intl* A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *mon*.

Definition at line 256 of file iomanip.

```
4.11.4.76 template<typename _Tp > pair<_Tp*, ptrdiff_t>
std::get_temporary_buffer ( ptrdiff_t __len )
```

Allocates a temporary buffer.

#### Parameters

*len* The number of objects of type `Tp`.

#### Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(Tp)`), or a pair of 0 values if no storage can be obtained*. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

```
4.11.4.77 template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream< _CharT, _Traits > & std::getline ( basic_istream<
_CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
& __str, _CharT __delim )
```

Read a line from stream into a string.

#### Parameters

- is* Input stream.
- str* Buffer to store into.
- delim* Character marking end of line.

#### Returns

Reference to the input stream.

Stores characters from *is* into *str* until *delim* is found, the end of the stream is encountered, or *str.max\_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If *delim* was encountered, it is extracted but not stored into *str*.

Definition at line 1068 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

```
4.11.4.78 template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream< _CharT, _Traits>& std::getline ( basic_istream<
_CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
& __str ) [inline]
```

Read a line from stream into a string.

#### Parameters

- is* Input stream.
- str* Buffer to store into.

#### Returns

Reference to the input stream.

Stores characters from *is* into *str* until ‘

’ is found, the end of the stream is encountered, or *str.max\_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*.

Any previous contents of *str* are erased. If end of line was encountered, it is extracted but not stored into *str*.

Definition at line 2641 of file basic\_string.h.

References `std::basic_ios<_CharT, _Traits >::widen()`.

**4.11.4.79** `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline ( basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim )`

Read a line from stream into a string.

#### Parameters

- `__is` Input stream.
- `__str` Buffer to store into.
- `__delim` Character marking end of line.

#### Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

Definition at line 622 of file vstring.tcc.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::erase()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::max_size()`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**4.11.4.80** `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::getline ( basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Read a line from stream into a string.

#### Parameters

- `__is` Input stream.

*\_\_str* Buffer to store into.

### Returns

Reference to the input stream.

Stores characters from *is* into *\_\_str* until ‘

’ is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into *\_\_str*. Any previous contents of *\_\_str* are erased. If end of line was encountered, it is extracted but not stored into *\_\_str*.

Definition at line 2447 of file `vstring.h`.

References `std::basic_ios<_CharT, _Traits>::widen()`.

#### 4.11.4.81 `template<typename _Facet > bool std::has_facet ( const locale & __loc ) throw ()`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

### Parameters

*Facet* The facet type to test the presence of.

*locale* The locale to test.

### Returns

true if locale contains a facet of type `Facet`, else false.

Definition at line 91 of file `locale_classes.tcc`.

#### 4.11.4.82 `ios_base& std::hex ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 940 of file `ios_base.h`.

References `std::ios_base::setf()`.

```
4.11.4.83 template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _Tp > _Tp std::inner_product ( _InputIterator1 __first1,
            _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init )
            [inline]
```

Compute inner product of two ranges.

Starting with an initial value of *init*, multiplies successive elements from the two ranges and adds each product into the accumulated value using operator+(). The values in the ranges are processed in order.

#### Parameters

*first1* Start of range 1.

*last1* End of range 1.

*first2* Start of range 2.

*init* Starting value to add other values to.

#### Returns

The final inner product.

Definition at line 170 of file stl\_numeric.h.

```
4.11.4.84 template<typename _InputIterator1 , typename _InputIterator2
            , typename _Tp , typename _BinaryOperation1 , typename
            _BinaryOperation2 > _Tp std::inner_product ( _InputIterator1
            __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
            __init, _BinaryOperation1 __binary_op1, _BinaryOperation2
            __binary_op2 ) [inline]
```

Compute inner product of two ranges.

Starting with an initial value of *init*, applies *binary\_op2* to successive elements from the two ranges and accumulates each result into the accumulated value using *binary\_op1*. The values in the ranges are processed in order.

#### Parameters

*first1* Start of range 1.

*last1* End of range 1.

*first2* Start of range 2.

*init* Starting value to add other values to.

*binary\_op1* Function object to accumulate with.

*binary\_op2* Function object to apply to pairs of input values.

**Returns**

The final inner product.

Definition at line 202 of file `stl_numeric.h`.

**4.11.4.85** `ios_base& std::internal ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 907 of file `ios_base.h`.

References `std::ios_base::setf()`.

**4.11.4.86** `template<typename _ForwardIterator , typename _Tp > void  
std::iota ( _ForwardIterator __first, _ForwardIterator __last, _Tp  
__value )`

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

**Parameters**

*first* Start of range.

*last* End of range.

*value* Starting value.

**Returns**

Nothing.

Definition at line 81 of file `stl_numeric.h`.

**4.11.4.87** `template<typename _CharT > bool std::isalnum ( _CharT __c,  
const locale & __loc ) [inline]`

Convenience interface to `cctype.is(cctype_base::alnum, __c)`.

**4.11.4.88** `template<typename _CharT > bool std::isalpha ( _CharT __c,  
const locale & __loc ) [inline]`

Convenience interface to `cctype.is(cctype_base::alpha, __c)`.

**4.11.4.89** `template<typename _CharT > bool std::isctrl ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

**4.11.4.90** `template<typename _CharT > bool std::isdigit ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

**4.11.4.91** `template<typename _CharT > bool std::isgraph ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

**4.11.4.92** `template<typename _CharT > bool std::islower ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

**4.11.4.93** `template<typename _CharT > bool std::isprint ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

**4.11.4.94** `template<typename _CharT > bool std::ispunct ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

**4.11.4.95** `template<typename _CharT > bool std::isspace ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

**4.11.4.96** `template<typename _CharT > bool std::isupper ( _CharT __c, const locale & __loc ) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

**4.11.4.97** `template<typename _CharT > bool std::isxdigit ( _CharT __c,  
const locale & __loc ) [inline]`

Convenience interface to `cctype.is(cctype_base::xdigit, __c)`.

**4.11.4.98** `ios_base& std::left ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 915 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by operator<<().

**4.11.4.99** `ios_base& std::noboolalpha ( ios_base & __base ) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 802 of file `ios_base.h`.

References `boolalpha()`, and `std::ios_base::unsetf()`.

**4.11.4.100** `ios_base& std::noshowbase ( ios_base & __base ) [inline]`

Calls `base.unsetf(ios_base::showbase)`.

Definition at line 818 of file `ios_base.h`.

References `showbase()`, and `std::ios_base::unsetf()`.

**4.11.4.101** `ios_base& std::noshowpoint ( ios_base & __base ) [inline]`

Calls `base.unsetf(ios_base::showpoint)`.

Definition at line 834 of file `ios_base.h`.

References `showpoint()`, and `std::ios_base::unsetf()`.

**4.11.4.102** `ios_base& std::noshowpos ( ios_base & __base ) [inline]`

Calls `base.unsetf(ios_base::showpos)`.

Definition at line 850 of file `ios_base.h`.

References `showpos()`, and `std::ios_base::unsetf()`.

**4.11.4.103 ios\_base& std::noskipws ( ios\_base & *\_\_base* ) [inline]**

Calls base.unsetf(ios\_base::skipws).

Definition at line 866 of file ios\_base.h.

References skipws(), and std::ios\_base::unsetf().

**4.11.4.104 ios\_base& std::nounitbuf ( ios\_base & *\_\_base* ) [inline]**

Calls base.unsetf(ios\_base::unitbuf).

Definition at line 898 of file ios\_base.h.

References unitbuf(), and std::ios\_base::unsetf().

**4.11.4.105 ios\_base& std::nouppercase ( ios\_base & *\_\_base* ) [inline]**

Calls base.unsetf(ios\_base::uppercase).

Definition at line 882 of file ios\_base.h.

References std::ios\_base::unsetf(), and uppercase().

**4.11.4.106 ios\_base& std::oct ( ios\_base & *\_\_base* ) [inline]**

Calls base.setf(ios\_base::oct, ios\_base::basefield).

Definition at line 948 of file ios\_base.h.

References std::ios\_base::setf().

**4.11.4.107 template<typename *\_Tp* , typename *\_Alloc* > bool std::operator!=( const deque< *\_Tp*, *\_Alloc* > & *\_\_x*, const deque< *\_Tp*, *\_Alloc* > & *\_\_y* ) [inline]**

Based on operator==.

Definition at line 1853 of file stl\_deque.h.

**4.11.4.108 template<typename *\_CharT* , typename *\_Traits* , typename *\_Alloc* > bool std::operator!=( const basic\_string< *\_CharT*, *\_Traits*, *\_Alloc* > & *\_\_lhs*, const basic\_string< *\_CharT*, *\_Traits*, *\_Alloc* > & *\_\_rhs* ) [inline]**

Test difference of two strings.

**Parameters**

*lhs* First string.

*rhs* Second string.

**Returns**

True if *lhs.compare(rhs) != 0*. False otherwise.

Definition at line 2380 of file `basic_string.h`.

**4.11.4.109** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator!=( const _CharT * __lhs, const basic_string<  
_CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test difference of C string and string.

**Parameters**

*lhs* C string.

*rhs* String.

**Returns**

True if *rhs.compare(lhs) != 0*. False otherwise.

Definition at line 2392 of file `basic_string.h`.

**4.11.4.110** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator!=( const basic_string< _CharT, _Traits,  
_Alloc > & __lhs, const _CharT * __rhs ) [inline]`

Test difference of string and C string.

**Parameters**

*lhs* String.

*rhs* C string.

**Returns**

True if *lhs.compare(rhs) != 0*. False otherwise.

Definition at line 2404 of file `basic_string.h`.

---

**4.11.4.111** `template<typename _Tp, typename _Alloc > bool std::operator!=( const list<_Tp, _Alloc > & __x, const list<_Tp, _Alloc > & __y ) [inline]`

Based on operator==.

Definition at line 1519 of file stl\_list.h.

**4.11.4.112** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!=( const map<_Key, _Tp, _Compare, _Alloc > & __x, const map<_Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

Based on operator==.

Definition at line 861 of file stl\_map.h.

**4.11.4.113** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!=( const multimap<_Key, _Tp, _Compare, _Alloc > & __x, const multimap<_Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

Based on operator==.

Definition at line 792 of file stl\_multimap.h.

**4.11.4.114** `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator!=( const multiset<_Key, _Compare, _Alloc > & __x, const multiset<_Key, _Compare, _Alloc > & __y ) [inline]`

Returns !(x == y).

Definition at line 687 of file stl\_multiset.h.

**4.11.4.115** `template<typename _Res, typename... _Args> bool std::operator!=( const function<_Res(_Args...)> & __f, _M_clear_type * ) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

#### Returns

false if the wrapper has no target, true otherwise

This function will not throw an exception.

Definition at line 2200 of file functional.

**4.11.4.116** `template<class _T1 , class _T2 > bool std::operator!=( const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y ) [inline]`

Uses `operator==` to find the result.

Definition at line 167 of file stl\_pair.h.

**4.11.4.117** `template<typename _Res , typename... _Args> bool std::operator!=( _M_clear_type *, const function< _Res(_Args...)> & __f ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2206 of file functional.

**4.11.4.118** `template<typename _Tp , typename _Seq > bool std::operator!=( const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y ) [inline]`

Based on `operator==`.

Definition at line 294 of file stl\_queue.h.

**4.11.4.119** `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator!=( const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Returns `!(x == y)`.

Definition at line 700 of file stl\_set.h.

**4.11.4.120** `template<typename _Tp , typename _Seq > bool std::operator!=( const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y ) [inline]`

Based on `operator==`.

Definition at line 259 of file stl\_stack.h.

**4.11.4.121** `template<typename _Tp > bool std::operator!=( const  
_Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator<  
_Tp > & __y ) [inline]`

Forward list iterator inequality comparison.

Definition at line 264 of file forward\_list.h.

**4.11.4.122** `template<typename _Tp , typename _Alloc > bool std::operator!=(  
const forward_list< _Tp, _Alloc > & __lx, const forward_list<  
_Tp, _Alloc > & __ly ) [inline]`

Based on operator==.

Definition at line 1258 of file forward\_list.h.

**4.11.4.123** `template<typename _Tp , typename _Alloc > bool std::operator!=(  
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &  
__y ) [inline]`

Based on operator==.

Definition at line 1196 of file stl\_vector.h.

**4.11.4.124** `template<size_t _Nb> bitset<_Nb> std::operator&( const bitset<  
_Nb > & __x, const bitset< _Nb > & __y ) [inline]`

Global bitwise operations on bitsets.

#### Parameters

*x* A bitset.

*y* A bitset of the same size as *x*.

#### Returns

A new bitset.

These should be self-explanatory.

Definition at line 1366 of file bitset.

**4.11.4.125** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (   
const basic_string< _CharT, _Traits, _Alloc > & __lhs, const  
basic_string< _CharT, _Traits, _Alloc > & __rhs )`

Concatenate two strings.

#### Parameters

*lhs* First string.

*rhs* Last string.

#### Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 2263 of file basic\_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc >::append()`.

**4.11.4.126** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> std::operator+ ( const  
_CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &  
__rhs )`

Concatenate C string and string.

#### Parameters

*lhs* First string.

*rhs* Last string.

#### Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 692 of file basic\_string.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**4.11.4.127** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> std::operator+ ( const  
basic_string<_CharT, _Traits, _Alloc > & __lhs, const _CharT *  
__rhs ) [inline]`

Concatenate string and C string.

**Parameters**

*lhs* First string.

*rhs* Last string.

**Returns**

New string with *lhs* followed by *rhs*.

Definition at line 2300 of file basic\_string.h.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::append().

**4.11.4.128** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> std::operator+ ( const  
basic_string<_CharT, _Traits, _Alloc > & __lhs, _CharT __rhs )  
[inline]`

Concatenate string and character.

**Parameters**

*lhs* First string.

*rhs* Last string.

**Returns**

New string with *lhs* followed by *rhs*.

Definition at line 2316 of file basic\_string.h.

**4.11.4.129** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc > std::operator+ ( _CharT  
__lhs, const basic_string<_CharT, _Traits, _Alloc > & __rhs )`

Concatenate character and string.

**Parameters**

*lhs* First string.

*rhs* Last string.

**Returns**

New string with *lhs* followed by *rhs*.

Definition at line 708 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::size().

**4.11.4.130** `template<typename _Tp, typename _Alloc > bool std::operator< ( const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y ) [inline]`

Deque ordering relation.

#### Parameters

*x* A deque.

*y* A deque of the same type as *x*.

#### Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1845 of file `stl_deque.h`.

References `lexicographical_compare()`.

**4.11.4.131** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator< ( const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

Multimap ordering relation.

#### Parameters

*x* A multimap.

*y* A multimap of the same type as *x*.

#### Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 785 of file `stl_multimap.h`.

**4.11.4.132** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator<( const basic_string< _CharT, _Traits, _Alloc  
> & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs  
) [inline]`

Test if string precedes string.

#### Parameters

*lhs* First string.

*rhs* Second string.

#### Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2417 of file basic\_string.h.

**4.11.4.133** `template<typename _Tp, typename _Alloc > bool std::operator<( const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y ) [inline]`

List ordering relation.

#### Parameters

*x* A list.

*y* A list of the same type as *x*.

#### Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1512 of file stl\_list.h.

References `lexicographical_compare()`.

**4.11.4.134** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator<( const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

Map ordering relation.

**Parameters**

- x* A map.
- y* A map of the same type as *x*.

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 854 of file `stl_map.h`.

```
4.11.4.135 template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator< ( const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y )
[inline]
```

Multiset ordering relation.

**Parameters**

- x* A multiset.
- y* A multiset of the same type as *x*.

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 680 of file `stl_multiset.h`.

```
4.11.4.136 template<class _T1 , class _T2 > bool std::operator< ( const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y ) [inline]
```

<<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>>

Definition at line 160 of file `stl_pair.h`.

**4.11.4.137** `template<typename _Tp, typename _Seq > bool std::operator< ( const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y ) [inline]`

Queue ordering relation.

#### Parameters

- x* A queue.
- y* A queue of the same type as *x*.

#### Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 288 of file `stl_queue.h`.

**4.11.4.138** `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator< ( const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Set ordering relation.

#### Parameters

- x* A set.
- y* A set of the same type as *x*.

#### Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 693 of file `stl_set.h`.

**4.11.4.139** `template<typename _Tp, typename _Alloc > bool std::operator< ( const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Forward list ordering relation.

**Parameters**

*lx* A forward\_list.

*ly* A forward\_list of the same type as *lx*.

**Returns**

True iff *lx* is lexicographically less than *ly*.

This is a total ordering relation. It is linear in the size of the forward lists. The elements must be comparable with <.

See std::lexicographical\_compare() for how the determination is made.

Definition at line 1250 of file forward\_list.h.

References lexicographical\_compare().

**4.11.4.140** `template<typename _Tp, typename _Alloc > bool std::operator< ( const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y ) [inline]`

Vector ordering relation.

**Parameters**

*x* A vector.

*y* A vector of the same type as *x*.

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with <.

See std::lexicographical\_compare() for how the determination is made.

Definition at line 1189 of file stl\_vector.h.

References lexicographical\_compare().

**4.11.4.141** `template<typename _Tp, typename _Seq > bool std::operator< ( const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y ) [inline]`

Stack ordering relation.

**Parameters**

- x* A stack.
- y* A stack of the same type as *x*.

**Returns**

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 253 of file `stl_stack.h`.

**4.11.4.142** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator<( const _CharT * __lhs, const basic_string<  
_CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if C string precedes string.

**Parameters**

- lhs* C string.
- rhs* String.

**Returns**

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2441 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.143** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator<( const basic_string< _CharT, _Traits, _Alloc  
> & __lhs, const _CharT * __rhs ) [inline]`

Test if string precedes C string.

**Parameters**

- lhs* String.
- rhs* C string.

**Returns**

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2429 of file `basic_string.h`.

**4.11.4.144** `template<class _Traits > basic_ostream<char, _Traits>&  
std::operator<<( basic_ostream< char, _Traits > & __out, signed  
char __c ) [inline]`

Character inserters.

**Parameters**

*out* An output stream.

*c* A character.

**Returns**

*out*

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 466 of file `ostream`.

**4.11.4.145** `template<class _CharT , class _Traits , size_t _Nb>  
std::basic_ostream<_CharT, _Traits>& std::operator<<(   
std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb  
> & __x )`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1471 of file `bitset`.

References `std::__ctype_abstract_base< _CharT >::widen()`.

```
4.11.4.146 template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
basic_ostream<_CharT, _Traits > & __out, const _CharT * __s )
[inline]
```

String inserters.

#### Parameters

*out* An output stream.

*s* A character string.

#### Returns

*out*

#### Precondition

*s* must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 491 of file ostream.

```
4.11.4.147 template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
basic_ostream<_CharT, _Traits > & __out, _CharT __c )
[inline]
```

Character inserters.

#### Parameters

*out* An output stream.

*c* A character.

#### Returns

*out*

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 449 of file `ostream`.

```
4.11.4.148 template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
basic_ostream<_CharT, _Traits > & __out, char __c )
[inline]
```

Character inserters.

#### Parameters

*out* An output stream.

*c* A character.

#### Returns

*out*

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 454 of file `ostream`.

```
4.11.4.149 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< ( basic_ostream< char, _Traits > & __out, char
__c ) [inline]
```

Character inserters.

#### Parameters

*out* An output stream.

*c* A character.

#### Returns

*out*

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If `c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 460 of file `ostream`.

```
4.11.4.150 template<typename _CharT , typename _Traits , typename
    _Alloc , template< typename, typename, typename > class
    _Base> basic_ostream<_CharT, _Traits>& std::operator<<
    ( basic_ostream< _CharT, _Traits > & __os, const
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &
    __str ) [inline]
```

Write string to a stream.

#### Parameters

`__os` Output stream.  
`__str` String to write out.

#### Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2401 of file `vstring.h`.

```
4.11.4.151 template<typename _CharT , typename _Traits > basic_ostream<
    _CharT, _Traits > & std::operator<< ( basic_ostream< _CharT,
    _Traits > & __out, const char * __s )
```

String inserters.

#### Parameters

`out` An output stream.  
`s` A character string.

#### Returns

`out`

**Precondition**

*s* must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 321 of file `ostream.tcc`.

References `std::ios_base::badbit`.

```
4.11.4.152 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<( basic_ostream< char, _Traits > & __out, const
char * __s ) [inline]
```

String inserters.

**Parameters**

*out* An output stream.

*s* A character string.

**Returns**

*out*

**Precondition**

*s* must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 508 of file `ostream`.

```
4.11.4.153 template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<( basic_ostream< char, _Traits > & __out, const
signed char * __s ) [inline]
```

String inserters.

**Parameters**

*out* An output stream.

*s* A character string.

### Returns

*out*

### Precondition

*s* must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 521 of file ostream.

**4.11.4.154** `template<class _Traits > basic_ostream<char, _Traits>& std::operator<<( basic_ostream< char, _Traits > & __out, const unsigned char * __s ) [inline]`

String inserters.

### Parameters

*out* An output stream.

*s* A character string.

### Returns

*out*

### Precondition

*s* must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 526 of file ostream.

**4.11.4.155** `template<typename _CharT , typename _Traits , typename _Tp > basic_ostream<_CharT, _Traits>& std::operator<<( basic_ostream< _CharT, _Traits > && __os, const _Tp & __x ) [inline]`

Generic inserter for rvalue stream.

**Parameters**

- os* An input stream.
- x* A reference to the object being inserted.

**Returns**

*os*

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 579 of file ostream.

```
4.11.4.156 template<typename _CharT , typename _Traits , typename
    _Alloc > basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __os, const basic_string<
    _CharT, _Traits, _Alloc > & __str ) [inline]
```

Write string to a stream.

**Parameters**

- os* Output stream.
- str* String to write out.

**Returns**

Reference to the output stream.

Output characters of *str* into *os* following the same rules as for writing a C string.

Definition at line 2600 of file basic\_string.h.

```
4.11.4.157 template<class _Traits > basic_ostream<char, _Traits>&
    std::operator<< ( basic_ostream< char, _Traits > & __out,
    unsigned char __c ) [inline]
```

Character inserters.

**Parameters**

- out* An output stream.
- c* A character.

**Returns**

*out*

Behaves like one of the formatted arithmetic inserters described in [std::basic\\_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If `c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 471 of file `ostream`.

```
4.11.4.158 template<typename _Tp , typename _Alloc > bool std::operator<=
( const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc >
& __y ) [inline]
```

Based on `operator<`.

Definition at line 1867 of file `stl_deque.h`.

```
4.11.4.159 template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator<= ( const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs ) [inline]
```

Test if string doesn't follow string.

#### Parameters

*lhs* First string.

*rhs* Second string.

#### Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2491 of file `basic_string.h`.

```
4.11.4.160 template<typename _CharT , typename _Traits , typename
_Alloc > bool std::operator<= ( const _CharT * __lhs, const
basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string doesn't follow string.

#### Parameters

*lhs* C string.

*rhs* String.

**Returns**

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2515 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**4.11.4.161** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator<= ( const basic_string< _CharT, _Traits,  
_Alloc > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't follow C string.

**Parameters**

*lhs* String.

*rhs* C string.

**Returns**

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2503 of file `basic_string.h`.

**4.11.4.162** `template<typename _Tp, typename _Alloc > bool std::operator<=  
( const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y  
) [inline]`

Based on `operator<`.

Definition at line 1531 of file `stl_list.h`.

**4.11.4.163** `template<typename _Key, typename _Tp, typename _Compare,  
typename _Alloc > bool std::operator<= ( const map< _Key, _Tp,  
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,  
_Alloc > & __y ) [inline]`

Based on `operator<`.

Definition at line 875 of file `stl_map.h`.

**4.11.4.164** `template<typename _Key , typename _Tp , typename _Compare ,  
typename _Alloc > bool std::operator<= ( const multimap< _Key,  
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,  
_Compare, _Alloc > & __y ) [inline]`

Based on `operator<`.

Definition at line 806 of file `stl_multimap.h`.

**4.11.4.165** `template<typename _Key , typename _Compare , typename _Alloc  
> bool std::operator<= ( const multiset< _Key, _Compare, _Alloc  
> & __x, const multiset< _Key, _Compare, _Alloc > & __y )  
[inline]`

Returns `!(y < x)`.

Definition at line 701 of file `stl_multiset.h`.

**4.11.4.166** `template<class _T1 , class _T2 > bool std::operator<= ( const  
pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y )  
[inline]`

Uses `operator<` to find the result.

Definition at line 179 of file `stl_pair.h`.

**4.11.4.167** `template<typename _Tp , typename _Seq > bool std::operator<= (   
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &  
__y ) [inline]`

Based on `operator<`.

Definition at line 306 of file `stl_queue.h`.

**4.11.4.168** `template<typename _Key , typename _Compare , typename _Alloc  
> bool std::operator<= ( const set< _Key, _Compare, _Alloc > &  
__x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Returns `!(y < x)`.

Definition at line 714 of file `stl_set.h`.

**4.11.4.169** `template<typename _Tp, typename _Seq > bool std::operator<= ( const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y ) [inline]`

Based on operator<.

Definition at line 271 of file `stl_stack.h`.

**4.11.4.170** `template<typename _Tp, typename _Alloc > bool std::operator<= ( const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Based on operator<.

Definition at line 1279 of file `forward_list.h`.

**4.11.4.171** `template<typename _Tp, typename _Alloc > bool std::operator<= ( const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y ) [inline]`

Based on operator<.

Definition at line 1208 of file `stl_vector.h`.

**4.11.4.172** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator==( const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test equivalence of C string and string.

#### Parameters

*lhs* C string.

*rhs* String.

#### Returns

True if `rhs.compare(lhs) == 0`. False otherwise.

Definition at line 2355 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

```
4.11.4.173 template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator==( const map< _Key, _Tp,
    _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
    _Alloc > & __y ) [inline]
```

Map equality comparison.

#### Parameters

- x* A map.
- y* A map of the same type as *x*.

#### Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 837 of file `stl_map.h`.

```
4.11.4.174 template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator==( const basic_string< _CharT, _Traits,
    _Alloc > & __lhs, const _CharT * __rhs ) [inline]
```

Test equivalence of string and C string.

#### Parameters

- lhs* String.
- rhs* C string.

#### Returns

True if `lhs.compare(rhs) == 0`. False otherwise.

Definition at line 2367 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

```
4.11.4.175 template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator==( const multimap< _Key,
    _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
    _Compare, _Alloc > & __y ) [inline]
```

Multimap equality comparison.

**Parameters**

- x* A multimap.
- y* A multimap of the same type as *x*.

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 768 of file `stl_multimap.h`.

```
4.11.4.176 template<typename _Res , typename... _Args> bool std::operator==
( _M_clear_type *, const function< _Res(_Args...)> & _f )
[inline]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2188 of file `functional`.

```
4.11.4.177 template<typename _Tp , typename _Seq > bool std::operator== (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y ) [inline]
```

Queue equality comparison.

**Parameters**

- x* A queue.
- y* A queue of the same type as *x*.

**Returns**

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 270 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

**4.11.4.178** `template<typename _Key , typename _Compare , typename _Alloc  
> bool std::operator==( const set< _Key, _Compare, _Alloc > &  
__x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Set equality comparison.

#### Parameters

*x* A set.

*y* A set of the same type as *x*.

#### Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 676 of file `stl_set.h`.

**4.11.4.179** `template<typename _Tp , typename _Seq > bool std::operator==(  
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y  
) [inline]`

Stack equality comparison.

#### Parameters

*x* A stack.

*y* A stack of the same type as *x*.

#### Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 235 of file `stl_stack.h`.

**4.11.4.180** `template<typename _Tp > bool std::operator==( const  
_Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator<  
_Tp > & __y ) [inline]`

Forward list iterator equality comparison.

Definition at line 255 of file `forward_list.h`.

```
4.11.4.181 template<typename _Tp, typename _Alloc > bool std::operator==(
    const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &
    __y ) [inline]
```

Vector equality comparison.

#### Parameters

- x* A vector.
- y* A vector of the same type as *x*.

#### Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1172 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equal()`, and `std::vector< _Tp, _Alloc >::size()`.

```
4.11.4.182 template<typename _Key, typename _Compare, typename _Alloc
    > bool std::operator==( const multiset< _Key, _Compare, _Alloc
    > & __x, const multiset< _Key, _Compare, _Alloc > & __y )
    [inline]
```

Multiset equality comparison.

#### Parameters

- x* A multiset.
- y* A multiset of the same type as *x*.

#### Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 663 of file `stl_multiset.h`.

**4.11.4.183** `template<class _T1, class _T2 > bool std::operator==( const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y ) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 154 of file `stl_pair.h`.

References `std::pair< _T1, _T2 >::first`, and `std::pair< _T1, _T2 >::second`.

**4.11.4.184** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator==( const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test equivalence of two strings.

#### Parameters

*lhs* First string.

*rhs* Second string.

#### Returns

True if `lhs.compare(rhs) == 0`. False otherwise.

Definition at line 2334 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.185** `template<typename _Tp, typename _Alloc > bool std::operator==( const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y ) [inline]`

List equality comparison.

#### Parameters

*x* A list.

*y* A list of the same type as *x*.

#### Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1483 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

**4.11.4.186** `template<typename _StateT > bool std::operator==( const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs ) [inline]`

Test if equivalent to another position.

Definition at line 216 of file postypes.h.

**4.11.4.187** `template<typename _Tp, typename _Alloc > bool std::operator==( const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y ) [inline]`

Deque equality comparison.

#### Parameters

*x* A deque.

*y* A deque of the same type as *x*.

#### Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1827 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `equal()`, and `std::deque< _Tp, _Alloc >::size()`.

**4.11.4.188** `template<typename _Tp, typename _Alloc > bool std::operator==( const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly )`

Forward list equality comparison.

#### Parameters

*lx* A forward\_list

*ly* A forward\_list of the same type as *lx*.

#### Returns

True iff the size and elements of the forward lists are equal.

This is an equivalence relation. It is linear in the size of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 361 of file forward\_list.tcc.

References `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

**4.11.4.189** `template<typename _Res , typename... _Args> bool  
std::operator==( const function< _Res(_Args...)> & __f,  
_M_clear_type * ) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

#### Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2182 of file functional.

**4.11.4.190** `template<typename _Tp , typename _Alloc > bool std::operator< (  
const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > &  
__y ) [inline]`

Based on `operator<`.

Definition at line 1860 of file stl\_deque.h.

**4.11.4.191** `template<typename _Key , typename _Tp , typename _Compare ,  
typename _Alloc > bool std::operator< ( const map< _Key, _Tp,  
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,  
_Alloc > & __y ) [inline]`

Based on `operator<`.

Definition at line 868 of file stl\_map.h.

**4.11.4.192** `template<typename _Tp , typename _Seq > bool std::operator< (  
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &  
__y ) [inline]`

Based on `operator<`.

Definition at line 300 of file stl\_queue.h.

**4.11.4.193** `template<typename _CharT , typename _Traits , typename _Alloc  
> bool std::operator> ( const basic_string< _CharT, _Traits, _Alloc  
> & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs  
) [inline]`

Test if string follows string.

#### Parameters

*lhs* First string.

*rhs* Second string.

#### Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2454 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.194** `template<typename _Key , typename _Compare , typename _Alloc  
> bool std::operator> ( const set< _Key, _Compare, _Alloc > &  
__x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Returns  $y < x$ .

Definition at line 707 of file `stl_set.h`.

**4.11.4.195** `template<class _T1 , class _T2 > bool std::operator> ( const pair<  
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y ) [inline]`

Uses `operator<` to find the result.

Definition at line 173 of file `stl_pair.h`.

**4.11.4.196** `template<typename _Tp , typename _Alloc > bool std::operator> (   
const forward_list< _Tp, _Alloc > & __lx, const forward_list<  
_Tp, _Alloc > & __ly ) [inline]`

Based on `operator<`.

Definition at line 1265 of file `forward_list.h`.

**4.11.4.197** `template<typename _Key , typename _Compare , typename _Alloc  
> bool std::operator> ( const multiset< _Key, _Compare, _Alloc  
> & __x, const multiset< _Key, _Compare, _Alloc > & __y )  
[inline]`

Returns  $y < x$ .

Definition at line 694 of file `stl_multiset.h`.

**4.11.4.198** `template<typename _CharT , typename _Traits , typename _Alloc  
> bool std::operator> ( const _CharT* __lhs, const basic_string<  
_CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if C string follows string.

#### Parameters

*lhs* C string.

*rhs* String.

#### Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2478 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.199** `template<typename _Tp , typename _Alloc > bool std::operator> (   
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y )  
[inline]`

Based on `operator<`.

Definition at line 1525 of file `stl_list.h`.

**4.11.4.200** `template<typename _Tp , typename _Alloc > bool std::operator> (   
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &  
__y ) [inline]`

Based on `operator<`.

Definition at line 1202 of file `stl_vector.h`.

**4.11.4.201** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator> ( const basic_string<_CharT, _Traits, _Alloc  
> & __lhs, const _CharT* __rhs ) [inline]`

Test if string follows C string.

#### Parameters

*lhs* String.

*rhs* C string.

#### Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2466 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

**4.11.4.202** `template<typename _Tp, typename _Seq > bool std::operator> (   
const stack<_Tp, _Seq > & __x, const stack<_Tp, _Seq > & __y  
) [inline]`

Based on `operator<`.

Definition at line 265 of file `stl_stack.h`.

**4.11.4.203** `template<typename _Key, typename _Tp, typename _Compare,   
typename _Alloc > bool std::operator> ( const multimap<_Key,   
_Tp, _Compare, _Alloc > & __x, const multimap<_Key, _Tp,   
_Compare, _Alloc > & __y ) [inline]`

Based on `operator<`.

Definition at line 799 of file `stl_multimap.h`.

**4.11.4.204** `template<typename _Tp, typename _Alloc > bool std::operator>=   
( const deque<_Tp, _Alloc > & __x, const deque<_Tp, _Alloc >   
& __y ) [inline]`

Based on `operator<`.

Definition at line 1874 of file `stl_deque.h`.

**4.11.4.205** `template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator>= ( const basic_string< _CharT, _Traits,  
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &  
__rhs ) [inline]`

Test if string doesn't precede string.

#### Parameters

*lhs* First string.

*rhs* Second string.

#### Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2528 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.206** `template<typename _Tp, typename _Alloc > bool std::operator>=  
( const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y  
) [inline]`

Based on `operator<`.

Definition at line 1537 of file `stl_list.h`.

**4.11.4.207** `template<typename _Tp, typename _Seq > bool std::operator>=  
( const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y  
) [inline]`

Based on `operator<`.

Definition at line 277 of file `stl_stack.h`.

**4.11.4.208** `template<typename _Key, typename _Compare, typename _Alloc  
> bool std::operator>= ( const set< _Key, _Compare, _Alloc > &  
__x, const set< _Key, _Compare, _Alloc > & __y ) [inline]`

Returns `!(x < y)`.

Definition at line 721 of file `stl_set.h`.

---

**4.11.4.209** `template<typename _Tp, typename _Seq > bool std::operator>= ( const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y ) [inline]`

Based on operator<.

Definition at line 312 of file stl\_queue.h.

**4.11.4.210** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator>= ( const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

Based on operator<.

Definition at line 813 of file stl\_multimap.h.

**4.11.4.211** `template<typename _Tp, typename _Alloc > bool std::operator>= ( const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y ) [inline]`

Based on operator<.

Definition at line 1214 of file stl\_vector.h.

**4.11.4.212** `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator>= ( const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y ) [inline]`

Returns  $!(x < y)$ .

Definition at line 708 of file stl\_multiset.h.

**4.11.4.213** `template<typename _Tp, typename _Alloc > bool std::operator>= ( const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Based on operator<.

Definition at line 1272 of file forward\_list.h.

**4.11.4.214** `template<class _T1, class _T2 > bool std::operator>= ( const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y ) [inline]`

Uses `operator<` to find the result.

Definition at line 185 of file `stl_pair.h`.

**4.11.4.215** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator>= ( const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't precede C string.

#### Parameters

*lhs* String.

*rhs* C string.

#### Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2540 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.216** `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator>= ( const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Test if C string doesn't precede string.

#### Parameters

*lhs* C string.

*rhs* String.

#### Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2552 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

**4.11.4.217** `template<typename _Key , typename _Tp , typename _Compare ,  
typename _Alloc > bool std::operator>= ( const map< _Key, _Tp,  
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,  
_Alloc > & __y ) [inline]`

Based on operator<.

Definition at line 882 of file stl\_map.h.

**4.11.4.218** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_istream< _CharT, _Traits > & std::operator>> (   
basic_istream< _CharT, _Traits > & __is, basic_string< _CharT,  
_Traits, _Alloc > & __str )`

Read stream into a string.

#### Parameters

*is* Input stream.

*str* Buffer to store into.

#### Returns

Reference to the input stream.

Stores characters from *is* into *str* until whitespace is found, the end of the stream is encountered, or *str.max\_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased.

Definition at line 996 of file basic\_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::ios_base::getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::ios_base::width()`.

**4.11.4.219** `template<typename _CharT , typename _Traits , typename  
_Alloc , template< typename, typename, typename  
> class _Base> basic_istream< _CharT, _Traits > &  
std::operator>> ( basic_istream< _CharT, _Traits > & __is,  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &  
__str )`

Read stream into a string.

#### Parameters

*\_\_is* Input stream.

`__str` Buffer to store into.

### Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 547 of file `vstring.tcc`.

```
4.11.4.220 template<class _CharT , class _Traits , size_t _Nb>
std::basic_istream<_CharT, _Traits>& std::operator>> (
std::basic_istream<_CharT, _Traits > & __is, bitset<_Nb > &
__x )
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept `0` and `1` characters, and will only extract as many digits as the bitset will hold.

Definition at line 1403 of file `bitset`.

References `std::basic_string<_CharT, _Traits, _Alloc >::empty()`, `std::basic_string<_CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_ios<_CharT, _Traits >::widen()`.

```
4.11.4.221 template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::operator>> ( basic_istream<_CharT,
_Traits > & __in, _CharT * __s )
```

Character string extractors.

### Parameters

*in* An input stream.

*s* A pointer to a character array.

### Returns

*in*

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is the number of elements of the largest array of `*`
- *char\_type* that can store a terminating `eos`.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 935 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::getloc()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

**4.11.4.222** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::operator>> ( basic_istream<_CharT, _Traits> & __in, _CharT & __c )`

Character extractors.

#### Parameters

- in* An input stream.
- c* A character reference.

#### Returns

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 903 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**4.11.4.223** `template<class _Traits > basic_istream<char, _Traits>&  
std::operator>>( basic_istream< char, _Traits > & __in, signed  
char & __c ) [inline]`

Character extractors.

#### Parameters

- in* An input stream.
- c* A character reference.

#### Returns

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 709 of file istream.

**4.11.4.224** `template<class _Traits > basic_istream<char, _Traits>&  
std::operator>>( basic_istream< char, _Traits > & __in,  
unsigned char * __s ) [inline]`

Character string extractors.

#### Parameters

- in* An input stream.
- s* A pointer to a character array.

#### Returns

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of \*
- *char\_type* that can store a terminating *eos*.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- $n-1$  characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 751 of file `istream`.

```
4.11.4.225 template<typename _CharT , typename _Traits , typename
_Tp > basic_istream<_CharT, _Traits>& std::operator>> (
basic_istream< _CharT, _Traits > && __is, _Tp & __x )
[inline]
```

Generic extractor for rvalue stream.

#### Parameters

- is* An input stream.
- x* A reference to the extraction target.

#### Returns

*is*

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 847 of file `istream`.

```
4.11.4.226 template<class _Traits > basic_istream<char, _Traits>&
std::operator>> ( basic_istream< char, _Traits > & __in,
unsigned char & __c ) [inline]
```

Character extractors.

#### Parameters

- in* An input stream.
- c* A character reference.

**Returns**

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 704 of file istream.

```
4.11.4.227 template<class _Traits > basic_istream<char, _Traits>&
std::operator>>( basic_istream< char, _Traits > & __in, signed
char * __s ) [inline]
```

Character string extractors.

**Parameters**

*in* An input stream.

*s* A pointer to a character array.

**Returns**

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of \*
- *char\_type* that can store a terminating `eos`.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 756 of file istream.

**4.11.4.228** `template<> basic_istream<char>& std::operator>> ( basic_istream< char > & __in, char * __s )`

Character string extractors.

#### Parameters

- in* An input stream.
- s* A pointer to a character array.

#### Returns

*in*

Behaves like one of the formatted arithmetic extractors described in [std::basic\\_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of \*
- *char\_type* that can store a terminating `eos`.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**4.11.4.229** `template<size_t _Nb> bitset<_Nb> std::operator^ ( const bitset< _Nb > & __x, const bitset< _Nb > & __y ) [inline]`

Global bitwise operations on bitsets.

#### Parameters

- x* A bitset.

*y* A bitset of the same size as *x*.

**Returns**

A new bitset.

These should be self-explanatory.

Definition at line 1384 of file bitset.

**4.11.4.230** `template<size_t _Nb> bitset<_Nb> std::operator| ( const bitset<_Nb> & __x, const bitset<_Nb> & __y ) [inline]`

Global bitwise operations on bitsets.

**Parameters**

*x* A bitset.

*y* A bitset of the same size as *x*.

**Returns**

A new bitset.

These should be self-explanatory.

Definition at line 1375 of file bitset.

**4.11.4.231** `template<typename _InputIterator, typename _OutputIterator> > _OutputIterator std::partial_sum ( _InputIterator __first, _InputIterator __last, _OutputIterator __result )`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

**Parameters**

*first* Start of input range.

*last* End of input range.

*result* Output to write sums to.

**Returns**

Iterator pointing just beyond the values written to result.

Definition at line 233 of file stl\_numeric.h.

**4.11.4.232** `template<typename _InputIterator , typename _OutputIterator ,  
typename _BinaryOperation > _OutputIterator std::partial_sum (  
_InputIterator __first, _InputIterator __last, _OutputIterator  
__result, _BinaryOperation __binary_op )`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

#### Parameters

*first* Start of input range.

*last* End of input range.

*result* Output to write sums to.

#### Returns

Iterator pointing just beyond the values written to result.

Definition at line 273 of file stl\_numeric.h.

**4.11.4.233** `template<typename _MoneyT > _Put_money<_MoneyT>  
std::put_money ( const _MoneyT & __mon, bool __intl = false )  
[inline]`

Extended manipulator for inserting money.

#### Parameters

*mon* Either long double or a specialization of `basic_string`.

*intl* A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *mon*.

Definition at line 292 of file iomanip.

**4.11.4.234** `template<typename _Tp > reference_wrapper<_Tp> std::ref (  
reference_wrapper<_Tp > __t ) [inline]`

Partial specialization.

Definition at line 451 of file functional.

References ref().

**4.11.4.235** `template<typename _Tp > reference_wrapper<_Tp> std::ref (`  
`_Tp & __t ) [inline]`

Denotes a reference should be taken to a variable.

Definition at line 439 of file functional.

Referenced by ref().

**4.11.4.236** `template<typename _InputIterator , typename _OutputIterator`  
`, typename _Tp > _OutputIterator std::replace_copy (`  
`_InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, const _Tp & __old_value, const _Tp & __new_value )`

Copy a sequence, replacing each element of one value with another value.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

*old\_value* The value to be replaced.

*new\_value* The replacement value.

#### Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[first,last)` to the output range `[result,result+(last-first))` replacing elements equal to `old_value` with `new_value`.

Definition at line 3748 of file `stl_algo.h`.

**4.11.4.237** `_Resetiosflags std::resetiosflags ( ios_base::fmtflags __mask )`  
`[inline]`

Manipulator for `setf`.

#### Parameters

*mask* A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via *stream.setf(0,mask)*.

Definition at line 63 of file iomanip.

**4.11.4.238** `template<typename _Tp > void std::return_temporary_buffer (   
_Tp * __p ) [inline]`

The companion to [get\\_temporary\\_buffer\(\)](#).

#### Parameters

*p* A buffer previously allocated by [get\\_temporary\\_buffer](#).

#### Returns

None.

Frees the memory pointed to by *p*.

Definition at line 112 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

**4.11.4.239** `ios_base& std::right ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 923 of file `ios_base.h`.

References `std::ios_base::setf()`.

**4.11.4.240** `ios_base& std::scientific ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 965 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `operator<<()`.

**4.11.4.241** `new_handler std::set_new_handler ( new_handler ) throw ()`

Takes a replacement handler as the argument, returns the previous handler.

**4.11.4.242** `_Setbase std::setbase ( int __base ) [inline]`

Manipulator for `setf`.

**Parameters**

*base* A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *base* is any other value.

Definition at line 124 of file `iomanip`.

**4.11.4.243** `template<typename _CharT > _Setfill<_CharT> std::setfill ( _CharT __c ) [inline]`

Manipulator for `fill`.

**Parameters**

*c* The new fill character.

Sent to a stream object, this manipulator calls `fill(c)` for that object.

Definition at line 162 of file `iomanip`.

**4.11.4.244** `_Setiosflags std::setiosflags ( ios_base::fmtflags __mask ) [inline]`

Manipulator for `setf`.

**Parameters**

*mask* A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *mask*.

Definition at line 93 of file `iomanip`.

**4.11.4.245** `_Setprecision std::setprecision ( int __n ) [inline]`

Manipulator for `precision`.

**Parameters**

*n* The new precision.

Sent to a stream object, this manipulator calls `precision(n)` for that object.

Definition at line 192 of file `iomanip`.

#### 4.11.4.246 `_Setw std::setw ( int __n ) [inline]`

Manipulator for `width`.

##### Parameters

*n* The new width.

Sent to a stream object, this manipulator calls `width(n)` for that object.

Definition at line 222 of file `iomanip`.

#### 4.11.4.247 `ios_base& std::showbase ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 810 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowbase()`.

#### 4.11.4.248 `ios_base& std::showpoint ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::showpoint)`.

Definition at line 826 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowpoint()`.

#### 4.11.4.249 `ios_base& std::showpos ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::showpos)`.

Definition at line 842 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowpos()`.

#### 4.11.4.250 `ios_base& std::skipws ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::skipws)`.

Definition at line 858 of file ios\_base.h.

References `std::ios_base::setf()`.

Referenced by `noskipws()`, and `operator>>()`.

**4.11.4.251** `template<typename _Tp , typename _Alloc > void std::swap ( deque< _Tp, _Alloc > & __x, deque< _Tp, _Alloc > & __y ) [inline]`

See [std::deque::swap\(\)](#).

Definition at line 1881 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::swap()`.

**4.11.4.252** `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap ( multimap< _Key, _Tp, _Compare, _Alloc > & __x, multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

See [std::multimap::swap\(\)](#).

Definition at line 820 of file stl\_multimap.h.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

**4.11.4.253** `template<typename _Key , typename _Compare , typename _Alloc > void std::swap ( multiset< _Key, _Compare, _Alloc > & __x, multiset< _Key, _Compare, _Alloc > & __y ) [inline]`

See [std::multiset::swap\(\)](#).

Definition at line 715 of file stl\_multiset.h.

References `std::multiset< _Key, _Compare, _Alloc >::swap()`.

**4.11.4.254** `template<class _T1 , class _T2 > void std::swap ( pair< _T1, _T2 > & __x, pair< _T1, _T2 > & __y ) [inline]`

See `std::pair::swap()`.

Definition at line 194 of file stl\_pair.h.

**4.11.4.255** `template<typename _Tp, typename _Alloc > void std::swap ( list<_Tp, _Alloc > & __x, list<_Tp, _Alloc > & __y ) [inline]`

See [std::list::swap\(\)](#).

Definition at line 1543 of file `stl_list.h`.

References `std::list<_Tp, _Alloc >::swap()`.

**4.11.4.256** `template<typename _Res, typename... _Args> void std::swap ( function<_Res(_Args...)> & __x, function<_Res(_Args...)> & __y ) [inline]`

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2218 of file `functional`.

Referenced by `__gnu_parallel::LoserTree< false, _Tp, _Compare >::delete_min_insert()`, `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >::delete_min_insert()`, `__gnu_parallel::parallel_nth_element()`, `__gnu_parallel::parallel_partition()`, `__gnu_parallel::qsb_divide()`, `__gnu_parallel::qsb_local_sort_with_helping()`, `__rotate()`, `__gnu_cxx::versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `std::basic_string< char >::assign()`, `__gnu_cxx::versa_string< _CharT, _Traits, _Alloc, _Base >::operator=()`, `std::vector< result_type >::operator=()`, `std::set< _Key, _Compare, _Alloc >::operator=()`, `std::multiset< _Key, _Compare, _Alloc >::operator=()`, `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`, `std::map< _Key, _Tp, _Compare, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::basic_string< char >::operator=()`, `std::vector< result_type >::swap()`, `std::list< _Tp, _Alloc >::swap()`, `std::function< _Res(_ArgTypes...)>::swap()`, and `std::forward_list< _Tp, _Alloc >::swap()`.

**4.11.4.257** `template<typename _CharT, typename _Traits, typename _Alloc > void std::swap ( basic_string< _CharT, _Traits, _Alloc > & __lhs, basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]`

Swap contents of two strings.

#### Parameters

*lhs* First string.

*rhs* Second string.

Exchanges the contents of *lhs* and *rhs* in constant time.

Definition at line 2565 of file basic\_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::swap()`.

**4.11.4.258** `template<typename _Tp, typename _Alloc > void std::swap ( forward_list<_Tp, _Alloc > & __lx, forward_list<_Tp, _Alloc > & __ly ) [inline]`

See `std::forward_list::swap()`.

Definition at line 1286 of file forward\_list.h.

References `std::forward_list<_Tp, _Alloc>::swap()`.

**4.11.4.259** `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > void std::swap ( map<_Key, _Tp, _Compare, _Alloc > & __x, map<_Key, _Tp, _Compare, _Alloc > & __y ) [inline]`

See `std::map::swap()`.

Definition at line 889 of file stl\_map.h.

References `std::map<_Key, _Tp, _Compare, _Alloc>::swap()`.

**4.11.4.260** `template<typename _Key, typename _Compare, typename _Alloc > void std::swap ( set<_Key, _Compare, _Alloc > & __x, set<_Key, _Compare, _Alloc > & __y ) [inline]`

See `std::set::swap()`.

Definition at line 728 of file stl\_set.h.

References `std::set<_Key, _Compare, _Alloc>::swap()`.

**4.11.4.261** `template<typename _Tp, typename _Alloc > void std::swap ( vector<_Tp, _Alloc > & __x, vector<_Tp, _Alloc > & __y ) [inline]`

See `std::vector::swap()`.

Definition at line 1220 of file stl\_vector.h.

References `std::vector<_Tp, _Alloc>::swap()`.

**4.11.4.262** `template<typename _CharT > _CharT std::tolower ( _CharT __c,  
const locale & __loc ) [inline]`

Convenience interface to `ctype.tolower(__c)`.

**4.11.4.263** `template<typename _CharT > _CharT std::toupper ( _CharT __c,  
const locale & __loc ) [inline]`

Convenience interface to `ctype.toupper(__c)`.

**4.11.4.264** `template<typename _InputIterator , typename _ForwardIterator >  
_ForwardIterator std::uninitialized_copy ( _InputIterator __first,  
_InputIterator __last, _ForwardIterator __result ) [inline]`

Copies the range `[first,last)` into `result`.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*result* An output iterator.

#### Returns

`result + (first - last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 106 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

**4.11.4.265** `template<typename _InputIterator , typename _Size , typename  
_ForwardIterator > _ForwardIterator std::uninitialized_copy_n (   
_InputIterator __first, _Size __n, _ForwardIterator __result )  
[inline]`

Copies the range `[first,first+n)` into `result`.

#### Parameters

*first* An input iterator.

*n* The number of elements to copy.

*result* An output iterator.

**Returns**

result + n

Like `copy_n()`, but does not require an initialized output range.

Definition at line 531 of file `stl_uninitialized.h`.

References `__iterator_category()`.

**4.11.4.266** `template<typename _ForwardIterator, typename _Tp > void  
std::uninitialized_fill ( _ForwardIterator __first, _ForwardIterator  
__last, const _Tp & __x ) [inline]`

Copies the value `x` into the range `[first,last)`.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*x* The source value.

**Returns**

Nothing.

Like `fill()`, but does not require an initialized output range.

Definition at line 163 of file `stl_uninitialized.h`.

**4.11.4.267** `template<typename _ForwardIterator, typename _Size, typename  
_Tp > void std::uninitialized_fill_n ( _ForwardIterator __first,  
_Size __n, const _Tp & __x ) [inline]`

Copies the value `x` into the range `[first,first+n)`.

**Parameters**

*first* An input iterator.

*n* The number of copies to make.

*x* The source value.

**Returns**

Nothing.

Like `fill_n()`, but does not require an initialized output range.

Definition at line 279 of file `stl_uninitialized.h`.

**4.11.4.268** `ios_base& std::unitbuf ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 890 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nounitbuf()`.

**4.11.4.269** `ios_base& std::uppercase ( ios_base & __base ) [inline]`

Calls `base.setf(ios_base::uppercase)`.

Definition at line 874 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nouppercase()`.

**4.11.4.270** `template<typename _Facet > const _Facet & std::use_facet ( const locale & __loc )`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws [std::bad\\_cast](#) if the locale doesn't contain a facet of type `Facet`.

**Parameters**

*Facet* The facet type to access.

*locale* The locale to use.

**Returns**

Reference to facet of type `Facet`.

**Exceptions**

[std::bad\\_cast](#) if locale doesn't contain a facet of type `Facet`.

Definition at line 105 of file `locale_classes.tcc`.

**4.11.4.271** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::ws ( basic_istream< _CharT, _Traits > & __is )`

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a [std::basic\\_istream::sentry](#) inside your definition of operator>>.

Definition at line 996 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

## 4.11.5 Variable Documentation

### 4.11.5.1 `enable_if<(is_pointer<_Function>::value &&is_function<typename remove_pointer<_Function>::type>::value), typename result_of<_Function(_Args...)>::type>::type std::__invoke [inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 240 of file functional.

### 4.11.5.2 `ostream std::cerr`

Linked to standard error (unbuffered).

### 4.11.5.3 `istream std::cin`

Linked to standard input.

### 4.11.5.4 `ostream std::clog`

Linked to standard error (buffered).

#### 4.11.5.5 ostream std::cout

Linked to standard output.

#### 4.11.5.6 wostream std::wcerr

Linked to standard error (unbuffered).

#### 4.11.5.7 wistream std::wcin

Linked to standard input.

#### 4.11.5.8 wostream std::wclog

Linked to standard error (buffered).

#### 4.11.5.9 wostream std::wcout

Linked to standard output.

## 4.12 std::\_\_debug Namespace Reference

GNU debug code, replaces standard behavior with debug behavior.

### Classes

- class [bitset](#)  
*Class [std::bitset](#) with additional safety/checking/debug instrumentation.*
- class [deque](#)  
*Class [std::deque](#) with safety/checking/debug instrumentation.*
- class [list](#)  
*Class [std::list](#) with safety/checking/debug instrumentation.*
- class [map](#)  
*Class [std::map](#) with safety/checking/debug instrumentation.*
- class [multimap](#)

Class [std::multimap](#) with safety/checking/debug instrumentation.

- class [multiset](#)

Class [std::multiset](#) with safety/checking/debug instrumentation.

- class [set](#)

Class [std::set](#) with safety/checking/debug instrumentation.

- class [unordered\\_map](#)

Class [std::unordered\\_map](#) with safety/checking/debug instrumentation.

- class [unordered\\_multimap](#)

Class [std::unordered\\_multimap](#) with safety/checking/debug instrumentation.

- class [unordered\\_multiset](#)

Class [std::unordered\\_multiset](#) with safety/checking/debug instrumentation.

- class [unordered\\_set](#)

Class [std::unordered\\_set](#) with safety/checking/debug instrumentation.

- class [vector](#)

Class [std::vector](#) with safety/checking/debug instrumentation.

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool operator!=(const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator!=(const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!=(const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!=(const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!=(const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__`  
`__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _`  
`Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`

- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic_ostream< _CharT, _Traits > & operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
>  
bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >  
bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
>`

- bool **operator==** (const [unordered\\_multimap](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > &\_\_x, const [unordered\\_multimap](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > &\_\_y)
- template<typename \_Key, typename \_Compare, typename \_Allocator >  
bool **operator==** (const [multiset](#)< \_Key, \_Compare, \_Allocator > &\_\_lhs, const [multiset](#)< \_Key, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Value, typename \_Hash, typename \_Pred, typename \_Alloc >  
bool **operator==** (const [unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, const [unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator==** (const [deque](#)< \_Tp, \_Alloc > &\_\_lhs, const [deque](#)< \_Tp, \_Alloc > &\_\_rhs)
  - template<typename \_Key, typename \_Compare, typename \_Allocator >  
bool **operator==** (const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_lhs, const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Key, typename \_Compare, typename \_Allocator >  
bool **operator>** (const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_lhs, const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator>** (const [list](#)< \_Tp, \_Alloc > &\_\_lhs, const [list](#)< \_Tp, \_Alloc > &\_\_rhs)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator>** (const [vector](#)< \_Tp, \_Alloc > &\_\_lhs, const [vector](#)< \_Tp, \_Alloc > &\_\_rhs)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator>** (const [deque](#)< \_Tp, \_Alloc > &\_\_lhs, const [deque](#)< \_Tp, \_Alloc > &\_\_rhs)
  - template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Allocator >  
bool **operator>** (const [map](#)< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const [map](#)< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Allocator >  
bool **operator>** (const [multimap](#)< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const [multimap](#)< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Key, typename \_Compare, typename \_Allocator >  
bool **operator>** (const [multiset](#)< \_Key, \_Compare, \_Allocator > &\_\_lhs, const [multiset](#)< \_Key, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Key, typename \_Compare, typename \_Allocator >  
bool **operator>=** (const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_lhs, const [set](#)< \_Key, \_Compare, \_Allocator > &\_\_rhs)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator>=** (const [list](#)< \_Tp, \_Alloc > &\_\_lhs, const [list](#)< \_Tp, \_Alloc > &\_\_rhs)
  - template<typename \_Tp, typename \_Alloc >  
bool **operator>=** (const [vector](#)< \_Tp, \_Alloc > &\_\_lhs, const [vector](#)< \_Tp, \_Alloc > &\_\_rhs)

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs,`  
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, mul-`  
`timap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_-`  
`set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key,`  
`_Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

### 4.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

## 4.13 `std::__detail` Namespace Reference

Implementation details not part of the namespace `std` interface.

### Functions

- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator __transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Value, bool __cache >`  
`bool operator!= (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`

- `template<typename _Value, bool __cache>`  
`bool operator!= (const _Hashtable_iterator_base< _Value, __cache > &__x,`  
`const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`  
`bool operator== (const _Node_iterator_base< _Value, __cache > &__x, const`  
`_Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`  
`bool operator== (const _Hashtable_iterator_base< _Value, __cache > &__x,`  
`const _Hashtable_iterator_base< _Value, __cache > &__y)`

## Variables

- `const unsigned long __prime_list [ ]`

### 4.13.1 Detailed Description

Implementation details not part of the namespace `std` interface.

## 4.14 `std::__parallel` Namespace Reference

GNU parallel code, replaces standard behavior with parallel behavior.

## Classes

- [struct `\_CRandNumber`](#)  
*Functor wrapper for `std::rand()`.*

## Functions

- `template<typename _IIter, typename _Tp, typename _IteratorTag >`  
`_Tp __accumulate_switch (_IIter __begin, _IIter __end, _Tp __init, _-`  
`IteratorTag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`  
`_Tp __accumulate_switch (_IIter __begin, _IIter __end, _Tp __init, _-`  
`BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`  
`_Tp __accumulate_switch (_RAIter __begin, _RAIter __end, _Tp __init, _-`  
`BinaryOperation __binary_op, random\_access\_iterator\_tag, \_\_gnu\_parallel::-`  
`Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`

- `template<typename _IIter, typename _Tp, typename _Tag >`  
`_Tp __accumulate_switch (_IIter, _IIter, _Tp, _Tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag >`  
`_Tp __accumulate_switch (_IIter, _IIter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`  
`_Tp __accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random\_  
access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_  
parallel::parallel\_unbalanced)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`  
`_Tag2 >`  
`_OIter __adjacent_difference_switch (_IIter, _IIter, _OIter, _BinaryOper, _`  
`Tag1, _Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter __adjacent_difference_switch (_IIter, _IIter, _OIter, _BinaryOper,`  
`random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::  
Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __adjacent_difference_switch (_IIter __begin, _IIter __`  
`end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _`  
`IteratorTag2)`
- `template<typename _IIter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __adjacent_difference_switch (_IIter __begin, _IIter __end, _`  
`OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_  
tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_  
tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_  
access\_iterator\_tag)`
- `template<typename _FIterator, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _`  
`IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _`  
`BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _`  
`BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _IterTag >`  
`_FIter __adjacent_find_switch (_FIter, _FIter, _IterTag)`
- `template<typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_FIter __adjacent_find_switch (_FIter, _FIter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_  
access\_iterator\_tag)`

- `template<typename _RAIter, typename _Predicate >`  
`iterator_traits< _RAIter >::difference_type __count_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type __count_if_switch ( _Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type __count_if_switch ( _Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type __count_switch ( _RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type __count_switch ( _Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type __count_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2 >`  
`__Iter __find_first_of_switch ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_RAIter __find_first_of_switch ( _RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Iter __find_first_of_switch ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIter, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch ( _Iter, _Iter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_RAIter __find_first_of_switch ( _RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch ( _Iter, _Iter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`

- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`_Iter find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _-`  
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred,`  
`random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _-`  
`IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val,`  
`random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function for_each_switch (_RAIter __begin, _RAIter __end, _Function`  
`__f, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_`  
`tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function for_each_switch (_Iter __begin, _Iter __end, _Function __f, _-`  
`IteratorTag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _-`  
`IteratorTag >`  
`_OutputIterator generate_n_switch (_OutputIterator __begin, _Size __n, _-`  
`Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen,`  
`random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=-`  
`\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`  
`void generate_switch (_FIterator __begin, _FIterator __end, _Generator __-`  
`gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen,`  
`random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=-`  
`\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIter, typename _Generator, typename _IterTag >`  
`void generate_switch (_FIter, _FIter, _Generator, _IterTag)`

- `template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename BinaryFunction1 , typename BinaryFunction2 >`  
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _Tag1 , typename _Tag2 >`  
`_Tp __inner_product_switch (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >`  
`_Tp __inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_Tp __inner_product_switch (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >`  
`bool __lexicographical_compare_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Predicate >`  
`bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IterTag1 , typename _IterTag2 >`  
`bool __lexicographical_compare_switch (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _FIter , typename _Compare , typename _IterTag >`  
`_FIter __max_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`  
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`  
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`

- 
- `_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare > _OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
  - `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 > _OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
  - `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare > _OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
  - `template<typename _FIter, typename _Compare, typename _IterTag > _FIter __min_element_switch (_FIter, _FIter, _Compare, _IterTag)`
  - `template<typename _FIterator, typename _Compare, typename _IteratorTag > _FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
  - `template<typename _RAIter, typename _Compare > _RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 > pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Predicate > pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 > pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
  - `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
  - `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 > _OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
-

- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _-`  
`Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter __partial_sum_switch (_IIter, _IIter, _OIter, _BinaryOper, random\_`  
`access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator __partition_switch (_FIterator __begin, _FIterator __end, _Predicate`  
`__pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __-`  
`pred, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _Predicate, typename _IterTag >`  
`_FIter __partition_switch (_FIter, _FIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __-`  
`pred, const _Tp &__new_value, random\_access\_iterator\_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void __replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __-`  
`pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void __replace_if_switch (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _FIter, typename _Tp, typename _IterTag >`  
`void __replace_switch (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__-`  
`old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__-`  
`old_value, const _Tp &__new_value, random\_access\_iterator\_tag, \_\_gnu\_`  
`parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _-`  
`BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename`  
`_IteratorTag >`  
`_FIter __search_n_switch (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __-`  
`count, const _Tp &__val, _BinaryPredicate __binary_pred, random\_access\_`  
`iterator\_tag)`

- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator __search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`

- `_OIter __set_difference_switch` (`_IIter1`, `_IIter1`, `_IIter2`, `_IIter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_intersection_switch` (`_IIter1 __begin1`, `_IIter1 __end1`, `_IIter2 __begin2`, `_IIter2 __end2`, `_OutputIterator __result`, `_Predicate __pred`, `_IteratorTag1`, `_IteratorTag2`, `_IteratorTag3`)
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_intersection_switch` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_RAIter2 __end2`, `_Output_RAIter __result`, `_Predicate __pred`, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#))
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_intersection_switch` (`_IIter1`, `_IIter1`, `_IIter2`, `_IIter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_symmetric_difference_switch` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_RAIter2 __end2`, `_Output_RAIter __result`, `_Predicate __pred`, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#))
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_symmetric_difference_switch` (`_IIter1 __begin1`, `_IIter1 __end1`, `_IIter2 __begin2`, `_IIter2 __end2`, `_OutputIterator __result`, `_Predicate __pred`, `_IteratorTag1`, `_IteratorTag2`, `_IteratorTag3`)
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_symmetric_difference_switch` (`_IIter1`, `_IIter1`, `_IIter2`, `_IIter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_union_switch` (`_IIter1`, `_IIter1`, `_IIter2`, `_IIter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_union_switch` (`_IIter1 __begin1`, `_IIter1 __end1`, `_IIter2 __begin2`, `_IIter2 __end2`, `_OutputIterator __result`, `_Predicate __pred`, `_IteratorTag1`, `_IteratorTag2`, `_IteratorTag3`)
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_union_switch` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_RAIter2 __end2`, `_Output_RAIter __result`, `_Predicate`

- [\\_\\_pred](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#))
- `template<typename _Iter , typename _OIter , typename _UnaryOperation , typename _IterTag1 , typename _IterTag2 >`  
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
  - `template<typename _RAIter , typename _RAOIter , typename _UnaryOperation >`  
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
  - `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation >`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
  - `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
  - `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BiOperation >`  
`_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
  - `template<typename _Iter1 , typename _Iter2 , typename _OIter , typename _BiOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
  - `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BinaryOperation >`  
`_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
  - `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
  - `template<typename _Iter , typename _OIter , typename _Predicate , typename _IterTag1 , typename _IterTag2 >`  
`_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _-`  
`RandomAccess_OIter, _Predicate, random\_access\_iterator\_tag, random\_`  
`access\_iterator\_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __unique_copy_switch (_IIter __begin, _IIter __last, _-`  
`OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`  
`RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin,`  
`_RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred,`  
`random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation`  
`__binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation`  
`__binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation`  
`__binary_op)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu\_parallel::-`  
`Parallelism)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_`  
`tag)`
- `template<typename _IIter, typename _Tp >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, \_\_gnu\_parallel::-`  
`Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Tp >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp >`  
`_Tp accumulate (_IIter __begin, _IIter __end, _Tp __init)`
- `template<typename _IIter, typename _OIter >`  
`_OIter adjacent_difference (_IIter, _IIter, _OIter)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_IIter, _IIter, _OIter, _BinaryOper)`
- `template<typename _IIter, typename _OIter >`  
`_OIter adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::sequential\_`  
`tag)`

- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _FIter >`  
`_FIter adjacent_find (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter adjacent_find (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIter, typename _BiPredicate >`  
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate)`
- `template<typename _FIter, typename _BiPredicate >`  
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp >`  
`iterator\_traits< _IIter >::difference_type count (_IIter __begin, _IIter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp >`  
`iterator\_traits< _IIter >::difference_type count (_IIter __begin, _IIter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Tp >`  
`iterator\_traits< _IIter >::difference_type count (_IIter __begin, _IIter __end, const _Tp &__value)`
- `template<typename _IIter, typename _Predicate >`  
`iterator\_traits< _IIter >::difference_type count_if (_IIter __begin, _IIter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Predicate >`  
`iterator\_traits< _IIter >::difference_type count_if (_IIter __begin, _IIter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Predicate >`  
`iterator\_traits< _IIter >::difference_type count_if (_IIter __begin, _IIter __end, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2 >`  
`bool equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`bool equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2 >`  
`bool equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`bool equal (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)`
- `template<typename _IIter, typename _Tp >`  
`_IIter find (_IIter __begin, _IIter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp >`  
`_IIter find (_IIter __begin, _IIter __end, const _Tp &__val)`
- `template<typename _IIter, typename _FIterator >`  
`_IIter find_first_of (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _FIterator, typename _BinaryPredicate >`  
`_IIter find_first_of (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIter >`  
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
  
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`  
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::\_Parallelism \_\_parallelism\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::\_Parallelism \_\_parallelism\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator, \_\_gnu\_parallel::\_Parallelism)`

- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator _-`  
`__gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator _-`  
`__gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator _-`  
`__gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __-`  
`init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp _-`  
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, type-`  
`name BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, Binary-`  
`Function2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp _-`  
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_-`  
`gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`  
`_BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __-`  
`begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
  
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _`  
`Iter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _`  
`Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _`  
`Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __result, _Compare __comp, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _FIter >`  
`_FIter min_element (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter min_element (_FIter, _FIter, \_\_gnu\_parallel::Parallelism __-`  
`parallelism_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter min_element (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter min_element (_FIter, _FIter, _Compare)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu-`  
`parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`

- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, _Predicate __pred)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu -`  
`parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare`  
`__comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare`  
`__comp)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _-`  
`Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _-`  
`Compare __comp)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __-`  
`result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __-`  
`result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __-`  
`result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __-`  
`result)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter partition (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter partition (_FIter, _FIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator partition (_FIterator __begin, _FIterator __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAIter >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`  
`const _Tp &__new_value)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, \_\_gnu\_parallel::\_-`  
`Parallelism)`
- `template<typename _FIterator, typename _Tp >`  
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`  
`const _Tp &__new_value, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`

- `template<typename _FIterator, typename _Tp >`  
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value,`  
`const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Tp >`  
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`  
`_Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, \_\_gnu\_parallel::\_`  
`Parallelism)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`  
`_Tp &__new_value, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`  
`_Tp &__new_value)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2`  
`__begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2, _BinaryPredicate __pred)`

- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`

- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_symmetric_difference (_IIter1 __begin1, _IIter1 __`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_`  
`Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_union (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_union (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_IIter1 __begin1, _IIter1 __end1, _IIter2 __`  
`begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_`  
`mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_`  
`tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_`  
`mergesort\_exact\_tag __parallelism)`

- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`

- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag  
\_\_parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,  
_OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result,  
_UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,  
_OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism  
__parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result,  
_UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,  
_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result,  
_UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`

- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator`  
`__out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

#### 4.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

### 4.15 `std::__profile` Namespace Reference

GNU profile code, replaces standard behavior with profile behavior.

#### Classes

- class [bitset](#)  
*Class `std::bitset` wrapper with performance instrumentation.*
- class [deque](#)  
*Class `std::deque` wrapper with performance instrumentation.*
- class [list](#)  
*List wrapper with performance instrumentation.*

- class `map`  
*Class `std::map` wrapper with performance instrumentation.*
- class `multimap`  
*Class `std::multimap` wrapper with performance instrumentation.*
- class `multiset`  
*Class `std::multiset` wrapper with performance instrumentation.*
- class `set`  
*Class `std::set` wrapper with performance instrumentation.*
- class `unordered_map`  
*Class `std::unordered_map` wrapper with performance instrumentation.*
- class `unordered_multimap`  
*Class `std::unordered_multimap` wrapper with performance instrumentation.*
- class `unordered_multiset`  
*Unordered\_multiset wrapper with performance instrumentation.*
- class `unordered_set`  
*Unordered\_set wrapper with performance instrumentation.*

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`\_Key, \_Compare, \_Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const \_\_iterator\_tracker< _IteratorL, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_\_iterator\_tracker< _Iterator, _Sequence > &__lhs,`  
`const \_\_iterator\_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator!= (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__`  
`__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<typename _Iterator, typename _Sequence >`  
`\_\_iterator\_tracker< _Iterator, _Sequence > operator+ (typename \_\_iterator\_`  
`tracker< _Iterator, _Sequence >::difference_type __n, const \_\_iterator\_`  
`tracker< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`\_\_iterator\_tracker< _IteratorL, _Sequence >::difference_type operator- (const`  
`\_\_iterator\_tracker< _IteratorL, _Sequence > &__lhs, const \_\_iterator\_tracker<`  
`\_IteratorR, \_Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`\_\_iterator\_tracker< _Iterator, _Sequence >::difference_type operator- (const`  
`\_\_iterator\_tracker< _Iterator, _Sequence > &__lhs, const \_\_iterator\_tracker<`  
`\_Iterator, \_Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _`  
`Alloc > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`  
`Alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`  
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs,`  
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const`  
`set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _`  
`Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`  
`Alloc > &__rhs)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`  
`Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`  
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs,`  
`const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const`  
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key,`  
`_Compare, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, mul-`  
`timap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- template<typename \_Value, typename \_Hash, typename \_Pred, typename \_Alloc >  
void **swap** ([unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x,  
[unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Allocator >  
void **swap** ([map](#)< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, [map](#)< \_Key,  
\_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Tp, typename \_Alloc >  
void **swap** ([vector](#)< \_Tp, \_Alloc > &\_\_lhs, [vector](#)< \_Tp, \_Alloc > &\_\_rhs)
- template<typename \_Key, typename \_Compare, typename \_Allocator >  
void **swap** ([set](#)< \_Key, \_Compare, \_Allocator > &\_\_x, [set](#)< \_Key, \_Compare,  
\_Allocator > &\_\_y)
- template<typename \_Value, typename \_Hash, typename \_Pred, typename \_Alloc >  
void **swap** ([unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, [unordered\\_-  
set](#)< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y)
- template<typename \_Tp, typename \_Alloc >  
void **swap** ([vector](#)< \_Tp, \_Alloc > &&\_\_lhs, [vector](#)< \_Tp, \_Alloc > &\_\_rhs)

### 4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

## 4.16 std::chrono Namespace Reference

ISO C++ 0x entities sub namespace for time and date.

### Classes

- struct [duration](#)  
*duration*
- struct [duration\\_values](#)  
*duration\_values*
- struct [system\\_clock](#)  
*system\_clock*
- struct [time\\_point](#)  
*time\_point*
- struct [treat\\_as\\_floating\\_point](#)  
*treat\_as\_floating\_point*

## Typedefs

- typedef `system_clock` `high_resolution_clock`
- typedef `duration`< int, `ratio`< 3600 > > `hours`
- typedef `duration`< int64\_t, `micro` > `microseconds`
- typedef `duration`< int64\_t, `milli` > `milliseconds`
- typedef `duration`< int, `ratio`< 60 > > `minutes`
- typedef `system_clock` `monotonic_clock`
- typedef `duration`< int64\_t, `nano` > `nanoseconds`
- typedef `duration`< int64\_t > `seconds`

## Functions

- template<typename `_ToDuration` , typename `_Rep` , typename `_Period` >  
`enable_if`< `__is_duration`< `_ToDuration` >::value, `_ToDuration` >::type  
`duration_cast` (const `duration`< `_Rep`, `_Period` > &`_d`)
- template<typename `_Clock` , typename `_Duration1` , typename `_Duration2` >  
bool `operator!=` (const `time_point`< `_Clock`, `_Duration1` > &`_lhs`, const  
`time_point`< `_Clock`, `_Duration2` > &`_rhs`)
- template<typename `_Rep1` , typename `_Period1` , typename `_Rep2` , typename `_Period2` >  
bool `operator!=` (const `duration`< `_Rep1`, `_Period1` > &`_lhs`, const `duration`<  
`_Rep2`, `_Period2` > &`_rhs`)
- template<typename `_Rep1` , typename `_Period1` , typename `_Rep2` , typename `_Period2` >  
common\_type< `duration`< `_Rep1`, `_Period1` >, `duration`< `_Rep2`, `_Period2` >  
>::type `operator%` (const `duration`< `_Rep1`, `_Period1` > &`_lhs`, const `dura-`  
`tion`< `_Rep2`, `_Period2` > &`_rhs`)
- template<typename `_Rep1` , typename `_Period` , typename `_Rep2` >  
`duration`< typename `__common_rep_type`< `_Rep1`, typename `enable_if`<!\_  
`is_duration`< `_Rep2` >::value, `_Rep2` >::type >::type, `_Period` > `operator%`  
(const `duration`< `_Rep1`, `_Period` > &`_d`, const `_Rep2` &`_s`)
- template<typename `_Rep1` , typename `_Period` , typename `_Rep2` >  
`duration`< typename `__common_rep_type`< `_Rep1`, `_Rep2` >::type, `_Period` >  
`operator*` (const `duration`< `_Rep1`, `_Period` > &`_d`, const `_Rep2` &`_s`)
- template<typename `_Rep1` , typename `_Period` , typename `_Rep2` >  
`duration`< typename `__common_rep_type`< `_Rep2`, `_Rep1` >::type, `_Period` >  
`operator*` (const `_Rep1` &`_s`, const `duration`< `_Rep2`, `_Period` > &`_d`)
- template<typename `_Rep1` , typename `_Period1` , typename `_Rep2` , typename `_Period2` >  
common\_type< `duration`< `_Rep1`, `_Period1` >, `duration`< `_Rep2`, `_Period2` >  
>::type `operator+` (const `duration`< `_Rep1`, `_Period1` > &`_lhs`, const `dura-`  
`tion`< `_Rep2`, `_Period2` > &`_rhs`)
- template<typename `_Clock` , typename `_Duration1` , typename `_Rep2` , typename `_Period2` >  
`time_point`< `_Clock`, typename common\_type< `_Duration1`, `duration`< `_Rep2`,  
`_Period2` > >::type > `operator+` (const `time_point`< `_Clock`, `_Duration1` >  
&`_lhs`, const `duration`< `_Rep2`, `_Period2` > &`_rhs`)

- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >  
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >,  
_Duration2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs,  
const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,  
_Period2 > >::type > operator- (const time_point< _Clock, _Duration1 > &__  
lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
common_type< _Duration1, _Duration2 >::type operator- (const time_point<  
_Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__  
rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >  
>::type operator- (const duration< _Rep1, _Period1 > &__lhs, const dura-  
tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >  
duration< typename __common_rep_type< _Rep1, typename enable_if<!__  
is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > operator/  
(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _-  
Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool operator< (const time_point< _Clock, _Duration1 > &__lhs, const time_-  
point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration<  
_Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration<  
_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool operator<= (const time_point< _Clock, _Duration1 > &__lhs, const  
time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool operator== (const time_point< _Clock, _Duration1 > &__lhs, const  
time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration<  
_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool operator> (const time_point< _Clock, _Duration1 > &__lhs, const time_-  
point< _Clock, _Duration2 > &__rhs)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration<`  
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool operator>= (const time_point< _Clock, _Duration1 > &__lhs, const`  
`time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration<`  
`_Rep2, _Period2 > &__rhs)`
- `template<typename _ToDuration, typename _Clock, typename _Duration >`  
`enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _-`  
`ToDuration > >::type time_point_cast (const time_point< _Clock, _Duration`  
`> &__t)`

### 4.16.1 Detailed Description

ISO C++ 0x entities sub namespace for time and date.

### 4.16.2 Typedef Documentation

#### 4.16.2.1 `typedef duration<int, ratio<3600> > std::chrono::hours`

hours

Definition at line 488 of file chrono.

#### 4.16.2.2 `typedef duration<int64_t, micro> std::chrono::microseconds`

microseconds

Definition at line 476 of file chrono.

#### 4.16.2.3 `typedef duration<int64_t, milli> std::chrono::milliseconds`

milliseconds

Definition at line 479 of file chrono.

#### 4.16.2.4 `typedef duration<int, ratio< 60> > std::chrono::minutes`

minutes

Definition at line 485 of file chrono.

#### 4.16.2.5 `typedef duration<int64_t, nano> std::chrono::nanoseconds`

nanoseconds

Definition at line 473 of file chrono.

#### 4.16.2.6 `typedef duration<int64_t > std::chrono::seconds`

seconds

Definition at line 482 of file chrono.

### 4.16.3 Function Documentation

#### 4.16.3.1 `template<typename _ToDuration , typename _Rep , typename _Period > enable_if<__is_duration<_ToDuration>::value, _ToDuration>::type std::chrono::duration_cast ( const duration<_Rep, _Period > & __d ) [inline]`

duration\_cast

Definition at line 155 of file chrono.

Referenced by `std::this_thread::sleep_for()`, and `time_point_cast()`.

#### 4.16.3.2 `template<typename _ToDuration , typename _Clock , typename _Duration > enable_if<__is_duration<_ToDuration>::value, time_point<_Clock, _ToDuration >>::type std::chrono::time_point_cast ( const time_point<_Clock, _Duration > & __t ) [inline]`

time\_point\_cast

Definition at line 550 of file chrono.

References `duration_cast()`.

## 4.17 `std::decimal` Namespace Reference

ISO/IEC TR 24733 Decimal floating-point arithmetic.

### Classes

- class [decimal128](#)

3.2.4 Class *decimal128*.

- class [decimal32](#)  
3.2.2 Class [decimal32](#).
- class [decimal64](#)  
3.2.3 Class [decimal64](#).

## Functions

- double **decimal128\_to\_double** ([decimal128](#) \_\_d)
- float **decimal128\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal128\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal128\_to\_long\_long** ([decimal128](#) \_\_d)
- double **decimal32\_to\_double** ([decimal32](#) \_\_d)
- float **decimal32\_to\_float** ([decimal32](#) \_\_d)
- long double **decimal32\_to\_long\_double** ([decimal32](#) \_\_d)
- long long **decimal32\_to\_long\_long** ([decimal32](#) \_\_d)
- double **decimal64\_to\_double** ([decimal64](#) \_\_d)
- float **decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal32](#) \_\_d)
- double **decimal\_to\_double** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal128](#) \_\_d)
- float **decimal\_to\_float** ([decimal32](#) \_\_d)
- float **decimal\_to\_float** ([decimal64](#) \_\_d)
- float **decimal\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)

- bool **operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), unsigned int \_\_rhs)
- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), int \_\_rhs)
- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), unsigned long \_\_rhs)
- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), long \_\_rhs)
- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), long long \_\_rhs)
- **decimal32 operator\*** ([decimal32 \\_\\_lhs](#), unsigned long long \_\_rhs)
- **decimal32 operator\*** (int \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal32 operator\*** (unsigned int \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal32 operator\*** (long \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal32 operator\*** (unsigned long \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal32 operator\*** (long long \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal32 operator\*** (unsigned long long \_\_lhs, [decimal32 \\_\\_rhs](#))
- **decimal64 operator\*** ([decimal32 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), int \_\_rhs)
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), unsigned int \_\_rhs)
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), long \_\_rhs)
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), unsigned long \_\_rhs)
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), long long \_\_rhs)
- **decimal64 operator\*** ([decimal64 \\_\\_lhs](#), unsigned long long \_\_rhs)
- **decimal64 operator\*** (int \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** (unsigned int \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** (long \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** (unsigned long \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** (long long \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal64 operator\*** (unsigned long long \_\_lhs, [decimal64 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal32 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal64 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), int \_\_rhs)
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), unsigned int \_\_rhs)
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), long \_\_rhs)
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), unsigned long \_\_rhs)
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), long long \_\_rhs)
- **decimal128 operator\*** ([decimal128 \\_\\_lhs](#), unsigned long long \_\_rhs)
- **decimal128 operator\*** (int \_\_lhs, [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** (unsigned int \_\_lhs, [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** (long \_\_lhs, [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** (unsigned long \_\_lhs, [decimal128 \\_\\_rhs](#))
- **decimal128 operator\*** (long long \_\_lhs, [decimal128 \\_\\_rhs](#))

- [decimal128 operator\\*](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator\\*](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator+](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator+](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator+](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator+](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator+](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)

- **decimal64 operator+** ([decimal64 \\_\\_lhs](#), unsigned long long [\\_\\_rhs](#))
- **decimal64 operator+** (int [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator+** (unsigned int [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator+** (long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator+** (unsigned long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal32 operator+** ([decimal32 \\_\\_rhs](#))
- **decimal64 operator+** (long long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal128 \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), int [\\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), unsigned int [\\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), long [\\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), unsigned long [\\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), long long [\\_\\_rhs](#))
- **decimal32 operator-** ([decimal32 \\_\\_lhs](#), unsigned long long [\\_\\_rhs](#))
- **decimal32 operator-** (int [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** (unsigned int [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** (long [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** (unsigned long [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** (long long [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal32 operator-** (unsigned long long [\\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal32 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), int [\\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), unsigned int [\\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), long [\\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), unsigned long [\\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), long long [\\_\\_rhs](#))
- **decimal64 operator-** ([decimal64 \\_\\_lhs](#), unsigned long long [\\_\\_rhs](#))
- **decimal64 operator-** (int [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** (unsigned int [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** (long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** (unsigned long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** (long long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal64 operator-** (unsigned long long [\\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal32 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal64 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal128 \\_\\_lhs](#), [decimal32 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal128 \\_\\_lhs](#), [decimal64 \\_\\_rhs](#))
- **decimal128 operator-** ([decimal128 \\_\\_lhs](#), [decimal128 \\_\\_rhs](#))

- **decimal128 operator-** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator-** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator/** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator/** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)

- **decimal128 operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, int \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, long \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, int \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, int \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (int \_\_lhs, decimal128 \_\_rhs)

- bool **operator**< (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal64 \_\_rhs)

- bool **operator==** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator==** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator==** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator==** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator>** (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator>** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator>** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)

- bool **operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)

- `bool operator >= (int __lhs, decimal128 __rhs)`
- `bool operator >= (decimal32 __lhs, decimal32 __rhs)`
- `bool operator >= (unsigned long long __lhs, decimal32 __rhs)`
- `bool operator >= (decimal32 __lhs, unsigned int __rhs)`
- `bool operator >= (decimal128 __lhs, decimal32 __rhs)`
- `bool operator >= (unsigned long __lhs, decimal64 __rhs)`
- `bool operator >= (unsigned int __lhs, decimal64 __rhs)`
- `bool operator >= (decimal32 __lhs, unsigned long __rhs)`
- `bool operator >= (long __lhs, decimal128 __rhs)`
- `bool operator >= (int __lhs, decimal64 __rhs)`
- `bool operator >= (decimal32 __lhs, decimal128 __rhs)`
- `bool operator >= (decimal128 __lhs, long long __rhs)`
- `bool operator >= (int __lhs, decimal32 __rhs)`
- `bool operator >= (decimal64 __lhs, decimal128 __rhs)`
- `bool operator >= (long __lhs, decimal64 __rhs)`
- `bool operator >= (unsigned long long __lhs, decimal64 __rhs)`
- `bool operator >= (long long __lhs, decimal64 __rhs)`
- `bool operator >= (unsigned int __lhs, decimal128 __rhs)`
- `bool operator >= (long __lhs, decimal32 __rhs)`
- `bool operator >= (decimal64 __lhs, long __rhs)`
- `bool operator >= (unsigned long __lhs, decimal128 __rhs)`
- `bool operator >= (decimal128 __lhs, decimal64 __rhs)`

### 4.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

### 4.17.2 Function Documentation

#### 4.17.2.1 `long long std::decimal::decimal32_to_long_long ( decimal32 __d )`

Non-conforming extension: Conversion to integral type.

## 4.18 `std::placeholders` Namespace Reference

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

### 4.18.1 Detailed Description

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

## 4.19 std::rel\_ops Namespace Reference

The generated relational operators are sequestered here.

### Functions

- `template<class _Tp >`  
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator>= (const _Tp &__x, const _Tp &__y)`

### 4.19.1 Detailed Description

The generated relational operators are sequestered here.

### 4.19.2 Function Documentation

#### 4.19.2.1 `template<class _Tp > bool std::rel_ops::operator!= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines != for arbitrary types, in terms of ==.

#### Parameters

- `x` A thing.
- `y` Another thing.

#### Returns

- `x != y`

This function uses `==` to determine its result.

Definition at line 85 of file `stl_relops.h`.

**4.19.2.2** `template<class _Tp > bool std::rel_ops::operator<= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `<=` for arbitrary types, in terms of `<`.

#### Parameters

`x` A thing.

`y` Another thing.

#### Returns

`x <= y`

This function uses `<` to determine its result.

Definition at line 111 of file `stl_relops.h`.

**4.19.2.3** `template<class _Tp > bool std::rel_ops::operator> ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>` for arbitrary types, in terms of `<`.

#### Parameters

`x` A thing.

`y` Another thing.

#### Returns

`x > y`

This function uses `<` to determine its result.

Definition at line 98 of file `stl_relops.h`.

**4.19.2.4** `template<class _Tp > bool std::rel_ops::operator>= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

**Parameters**

- x* A thing.
- y* Another thing.

**Returns**

*x*  $\geq$  *y*

This function uses `<` to determine its result.

Definition at line 124 of file `stl_relops.h`.

## 4.20 `std::this_thread` Namespace Reference

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this\\_thread](#).

**Functions**

- [thread::id](#) `get_id()`
- `template<typename _Rep, typename _Period >`  
void [sleep\\_for](#) (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- `template<typename _Clock, typename _Duration >`  
void [sleep\\_until](#) (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void [yield](#) ()

### 4.20.1 Detailed Description

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this\\_thread](#).

### 4.20.2 Function Documentation

#### 4.20.2.1 `thread::id` `std::this_thread::get_id()` [`inline`]

`get_id`

Definition at line 252 of file `thread`.

#### 4.20.2.2 `template<typename _Rep, typename _Period > void` `std::this_thread::sleep_for` ( const `chrono::duration`< \_Rep, \_Period > & \_\_rtime ) [`inline`]

`sleep_for`

Definition at line 271 of file thread.

References `std::chrono::duration_cast()`.

Referenced by `sleep_until()`.

**4.20.2.3** `template<typename _Clock , typename _Duration > void  
std::this_thread::sleep_until ( const chrono::time_point< _Clock,  
_Duration > & _atime ) [inline]`

`sleep_until`

Definition at line 265 of file thread.

References `sleep_for()`.

**4.20.2.4** `void std::this_thread::yield ( ) [inline]`

`yield`

Definition at line 257 of file thread.

## 4.21 `std::tr1` Namespace Reference

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

### Namespaces

- namespace [\\_\\_detail](#)

### Classes

- struct [\\_\\_is\\_member\\_pointer\\_helper](#)  
*is\_member\_pointer*
- struct [add\\_const](#)  
*add\_const*
- struct [add\\_cv](#)  
*add\_cv*
- struct [add\\_pointer](#)  
*add\_pointer*

- struct [add\\_volatile](#)  
*add\_volatile*
- struct [alignment\\_of](#)  
*alignment\_of*
- struct [array](#)  
*A standard container for storing a fixed size sequence of elements.*
- class [bad\\_weak\\_ptr](#)  
*Exception possibly thrown by [shared\\_ptr](#).*
- struct [extent](#)  
*extent*
- struct [has\\_virtual\\_destructor](#)  
*has\_virtual\_destructor*
- struct [integral\\_constant](#)  
*integral\_constant*
- struct [is\\_abstract](#)  
*is\_abstract*
- struct [is\\_arithmetic](#)  
*is\_arithmetic*
- struct [is\\_array](#)  
*is\_array*
- struct [is\\_class](#)  
*is\_class*
- struct [is\\_compound](#)  
*is\_compound*
- struct [is\\_const](#)  
*is\_const*
- struct [is\\_empty](#)  
*is\_empty*

- struct [is\\_enum](#)  
*is\_enum*
- struct [is\\_floating\\_point](#)  
*is\_floating\_point*
- struct [is\\_function](#)  
*is\_function*
- struct [is\\_fundamental](#)  
*is\_fundamental*
- struct [is\\_integral](#)  
*is\_integral*
- struct [is\\_member\\_function\\_pointer](#)  
*is\_member\_function\_pointer*
- struct [is\\_member\\_object\\_pointer](#)  
*is\_member\_object\_pointer*
- struct [is\\_object](#)  
*is\_object*
- struct [is\\_pointer](#)  
*is\_pointer*
- struct [is\\_polymorphic](#)  
*is\_polymorphic*
- struct [is\\_same](#)  
*is\_same*
- struct [is\\_scalar](#)  
*is\_scalar*
- struct [is\\_union](#)  
*is\_union*
- struct [is\\_void](#)  
*is\_void*

- struct [is\\_volatile](#)  
*is\_volatile*
- struct [rank](#)  
*rank*
- struct [remove\\_all\\_extents](#)  
*remove\_all\_extents*
- struct [remove\\_const](#)  
*remove\_const*
- struct [remove\\_cv](#)  
*remove\_cv*
- struct [remove\\_extent](#)  
*remove\_extent*
- struct [remove\\_pointer](#)  
*remove\_pointer*
- struct [remove\\_volatile](#)  
*remove\_volatile*

## Typedefs

- typedef [integral\\_constant](#)< bool, false > [false\\_type](#)
- typedef [integral\\_constant](#)< bool, true > [true\\_type](#)

## Functions

- template<typename [\\_Tp](#) >  
[std::complex](#)< [\\_Tp](#) > [\\_\\_complex\\_acos](#) (const [std::complex](#)< [\\_Tp](#) > &[\\_z](#))
- template<typename [\\_Tp](#) >  
[std::complex](#)< [\\_Tp](#) > [\\_\\_complex\\_acosh](#) (const [std::complex](#)< [\\_Tp](#) > &[\\_z](#))
- template<typename [\\_Tp](#) >  
[std::complex](#)< [\\_Tp](#) > [\\_\\_complex\\_asin](#) (const [std::complex](#)< [\\_Tp](#) > &[\\_z](#))
- template<typename [\\_Tp](#) >  
[std::complex](#)< [\\_Tp](#) > [\\_\\_complex\\_asinh](#) (const [std::complex](#)< [\\_Tp](#) > &[\\_z](#))

- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atan` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atanh` (const `std::complex< _Tp >` &\_\_z)
- `void __throw_bad_weak_ptr ()`
- `template<typename _Tp >`  
`std::complex< _Tp > acos` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`std::complex< _Tp > acosh` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type arg` (\_Tp \_\_x)
- `template<typename _Tp >`  
`std::complex< _Tp > asin` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`std::complex< _Tp > asinh` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`std::complex< _Tp > atan` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`std::complex< _Tp > atanh` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float betaf` (float \_\_x, float \_\_y)
- `long double betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1` (\_Tp \_\_k)
- `float comp_ellint_1f` (float \_\_k)
- `long double comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2` (\_Tp \_\_k)
- `float comp_ellint_2f` (float \_\_k)
- `long double comp_ellint_2l` (long double \_\_k)

- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp\_ellint\_3 (_Tp __k, _-`  
`Tpn __nu)`
- `float comp\_ellint\_3f (float __k, float __nu)`
- `long double comp\_ellint\_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf\_hyperg (_Tpa __a,`  
`_Tpc __c, _Tp __x)`
- `float conf\_hypergf (float __a, float __c, float __x)`
- `long double conf\_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj (_Tp`  
`__x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_i (_Tpnu __nu,`  
`_Tp __x)`
- `float cyl\_bessel\_if (float __nu, float __x)`
- `long double cyl\_bessel\_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_j (_Tpnu __nu,`  
`_Tp __x)`
- `float cyl\_bessel\_jf (float __nu, float __x)`
- `long double cyl\_bessel\_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_k (_Tpnu __nu,`  
`_Tp __x)`
- `float cyl\_bessel\_kf (float __nu, float __x)`
- `long double cyl\_bessel\_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_neumann (_Tpnu __nu,`  
`_Tp __x)`
- `float cyl\_neumannf (float __nu, float __x)`
- `long double cyl\_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint\_1 (_Tp __k, _Tpp __-`  
`phi)`
- `float ellint\_1f (float __k, float __phi)`
- `long double ellint\_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint\_2 (_Tp __k, _Tpp __-`  
`phi)`
- `float ellint\_2f (float __k, float __phi)`

- long double **ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **expint** (\_Tp \_\_x)
- float **expintf** (float \_\_x)
- long double **expintl** (long double \_\_x)
- template<typename \_Tp >  
\_Tp **fabs** (const std::complex< \_Tp > &\_\_z)
- template<std::size\_t \_Int, typename \_Tp, std::size\_t \_Nm >  
\_Tp & **get** (array< \_Tp, \_Nm > &\_\_arr)
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
const tuple\_element< \_Int, std::pair< \_Tp1, \_Tp2 > >::type & **get** (const std::pair< \_Tp1, \_Tp2 > &\_\_in)
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
tuple\_element< \_Int, std::pair< \_Tp1, \_Tp2 > >::type & **get** (std::pair< \_Tp1, \_Tp2 > &\_\_in)
- template<std::size\_t \_Int, typename \_Tp, std::size\_t \_Nm >  
const \_Tp & **get** (const array< \_Tp, \_Nm > &\_\_arr)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **hermitef** (unsigned int \_\_n, float \_\_x)
- long double **hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **imag** (\_Tp)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **laguerref** (unsigned int \_\_n, float \_\_x)
- long double **laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **legendref** (unsigned int \_\_n, float \_\_x)
- long double **legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **norm** (\_Tp \_\_x)

- `template<typename _Tp, std::size_t _Nm>`  
`bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `long double pow (long double __x, long double __y)`
- `double pow (double __x, double __y)`
- `template<typename _Tp >`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `float pow (float __x, float __y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`
- `template<typename _Tp, std::size_t _Nm >`  
`void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

### 4.21.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

## 4.22 std::tr1::\_\_detail Namespace Reference

Implementation details not part of the namespace [std::tr1](#) interface.

### 4.22.1 Detailed Description

Implementation details not part of the namespace [std::tr1](#) interface.

# Chapter 5

## Class Documentation

### 5.1 `__atomic0::atomic_address` Struct Reference

29.4.2, address types

#### Public Member Functions

- `atomic_address` (void \*\_\_v)
- `atomic_address` (const [atomic\\_address](#) &)
- bool `compare_exchange_strong` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `compare_exchange_strong` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool `compare_exchange_weak` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `compare_exchange_weak` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m=memory\_order\_seq\_cst)
- void \* `exchange` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)
- void \* `fetch_add` (ptrdiff\_t \_\_d, memory\_order \_\_m=memory\_order\_seq\_cst)
- void \* `fetch_sub` (ptrdiff\_t \_\_d, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool `is_lock_free` () const
- void \* `load` (memory\_order \_\_m=memory\_order\_seq\_cst) const
- `operator void *` () const
- void \* `operator+=` (ptrdiff\_t \_\_d)
- void \* `operator-=` (ptrdiff\_t \_\_d)
- [atomic\\_address](#) & `operator=` (const [atomic\\_address](#) &) volatile
- void \* `operator=` (void \*\_\_v)
- void `store` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.1.1 Detailed Description

29.4.2, address types

Definition at line 103 of file atomic\_0.h.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.2 `__atomic0::atomic_bool` Struct Reference

[atomic\\_bool](#)

### Public Member Functions

- **atomic\_bool** (bool \_\_i)
- **atomic\_bool** (const [atomic\\_bool](#) &)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **exchange** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const
- bool **load** (memory\_order \_\_m=memory\_order\_seq\_cst) const
- **operator bool** () const
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &) volatile
- bool **operator=** (bool \_\_i)
- void **store** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.2.1 Detailed Description

[atomic\\_bool](#)

Definition at line 391 of file atomic\_0.h.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.3 `__atomic0::atomic_flag` Struct Reference

[atomic\\_flag](#)

Inherits `__atomic_flag_base`.

### Public Member Functions

- `atomic_flag` (bool \_\_i)
- `atomic_flag` (const [atomic\\_flag](#) &)
- void `clear` (memory\_order \_\_m=memory\_order\_seq\_cst)
- `atomic_flag` & `operator=` (const [atomic\\_flag](#) &) volatile
- bool `test_and_set` (memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.3.1 Detailed Description

[atomic\\_flag](#)

Definition at line 85 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.4 `__atomic2::atomic_address` Struct Reference

29.4.2, address types

### Public Member Functions

- `atomic_address` (void \*\_\_v)
- `atomic_address` (const [atomic\\_address](#) &)
- bool `compare_exchange_strong` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `compare_exchange_strong` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool `compare_exchange_weak` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `compare_exchange_weak` (void \*&\_\_v1, void \*\_\_v2, memory\_order \_\_m=memory\_order\_seq\_cst)
- void \* `exchange` (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)
- void \* `fetch_add` (ptrdiff\_t \_\_d, memory\_order \_\_m=memory\_order\_seq\_cst)

- void \* **fetch\_sub** (ptrdiff\_t \_\_d, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const
- void \* **load** (memory\_order \_\_m=memory\_order\_seq\_cst) const
- **operator void** \* () const
- void \* **operator+=** (ptrdiff\_t \_\_d)
- void \* **operator-=** (ptrdiff\_t \_\_d)
- [atomic\\_address](#) & **operator=** (const [atomic\\_address](#) &) volatile
- void \* **operator=** (void \*\_\_v)
- void **store** (void \*\_\_v, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.4.1 Detailed Description

29.4.2, address types

Definition at line 81 of file atomic\_2.h.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.5 \_\_atomic2::atomic\_bool Struct Reference

[atomic\\_bool](#)

### Public Member Functions

- **atomic\_bool** (bool \_\_i)
- **atomic\_bool** (const [atomic\\_bool](#) &)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **exchange** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const
- bool **load** (memory\_order \_\_m=memory\_order\_seq\_cst) const
- **operator bool** () const
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &) volatile
- bool **operator=** (bool \_\_i)
- void **store** (bool \_\_i, memory\_order \_\_m=memory\_order\_seq\_cst)

### 5.5.1 Detailed Description

[atomic\\_bool](#)

Definition at line 391 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.6 `__atomic2::atomic_flag` Struct Reference

[atomic\\_flag](#)

Inherits `__atomic_flag_base`.

### Public Member Functions

- `atomic_flag` (`bool __i`)
- `atomic_flag` (`const atomic_flag &`)
- `atomic_flag & operator=` (`const atomic_flag &`) volatile

### 5.6.1 Detailed Description

[atomic\\_flag](#)

Definition at line 47 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.7 `__cxxabiv1::__forced_unwind` Class Reference

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

### 5.7.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

Definition at line 47 of file `cxxabi-forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi-forced.h](#)

## 5.8 `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >` Struct Template Reference

Policy for shared `__pool` objects.

Inherits `__common_pool_base< _PoolTp, _Thread >`.

### 5.8.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread> struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 456 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.9 `__gnu_cxx::__detail::__mini_vector< _Tp >` Class Template Reference

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

### Public Types

- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## 5.10 `__gnu_cxx::__detail::Bitmap_counter<_Tp>` Class Template Reference

### Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator \_\_pos) throw ()
- void **insert** (iterator \_\_pos, const\_reference \_\_x)
- reference **operator[]** (const size\_type \_\_pos) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference \_\_x)
- size\_type **size** () const throw ()

### 5.9.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::__mini_vector<_Tp>`

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`. It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses [operator new\(\)](#) to get memory, and [operator delete\(\)](#) to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 71 of file `bitmap_allocator.h`.

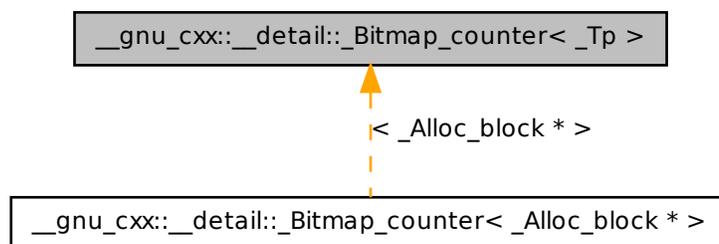
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.10 `__gnu_cxx::__detail::Bitmap_counter<_Tp>` Class Template Reference

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Inheritance diagram for `__gnu_cxx::__detail::_Bitmap_counter<_Tp>`:



## Public Member Functions

- [\\_Bitmap\\_counter](#) ([\\_BPVector](#) &Rvbp, long \_\_index=-1)
- `pointer _M_base () const throw ()`
- `bool _M_finished () const throw ()`
- `size_t * _M_get () const throw ()`
- `_Index_type _M_offset () const throw ()`
- `void _M_reset (long __index=-1) throw ()`
- `void _M_set_internal_bitmap (size_t * __new_internal_marker) throw ()`
- `_Index_type _M_where () const throw ()`
- [\\_Bitmap\\_counter](#) & `operator++ () throw ()`

### 5.10.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Bitmap_counter<_Tp>`

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Definition at line 400 of file `bitmap_allocator.h`.

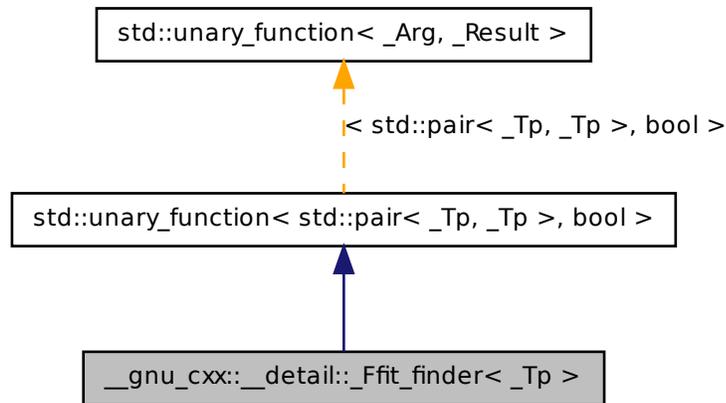
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.11 `__gnu_cxx::__detail::Ffit_finder<_Tp>` Class Template Reference

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Inheritance diagram for `__gnu_cxx::__detail::Ffit_finder<_Tp>`:



### Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

### Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

### 5.11.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Ffit_finder< _Tp >`

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 335 of file `bitmap_allocator.h`.

### 5.11.2 Member Typedef Documentation

**5.11.2.1** `typedef std::pair< _Tp, _Tp > std::unary_function< std::pair< _Tp, _Tp >, bool >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.11.2.2** `typedef bool std::unary_function< std::pair< _Tp, _Tp >, bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

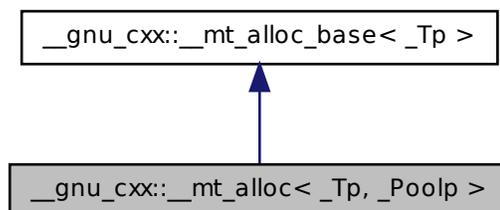
- [bitmap\\_allocator.h](#)

## 5.12 `__gnu_cxx::__mt_alloc< _Tp, _Poolp >` Class Template Reference

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp >`:



## Public Types

- typedef `_Poolp` **\_\_policy\_type**
- typedef `_Poolp::pool_type` **\_\_pool\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `__mt_alloc` (`const __mt_alloc &`) `throw ()`
- `template<typename _Tp1, typename _Poolp1 >`  
`__mt_alloc` (`const __mt_alloc<_Tp1, _Poolp1 > &`) `throw ()`
- `const __pool_base::_Tune` **\_M\_get\_options** ()
- `void` **\_M\_set\_options** (`__pool_base::_Tune __t`)
- `pointer` **address** (`reference __x`) `const`
- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **allocate** (`size_type __n, const void * = 0`)
- `template<typename... _Args >`  
`void` **construct** (`pointer __p, _Args &&... __args`)
- `void` **construct** (`pointer __p, const _Tp & __val`)
- `void` **deallocate** (`pointer __p, size_type __n`)

- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.12.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool,
true >> class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Definition at line 625 of file mt\_allocator.h.

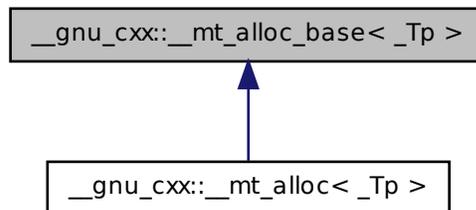
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.13 \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp > Class Template Reference

Base class for \_Tp dependent member functions.

Inheritance diagram for \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp >:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.13.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__mt_alloc_base< _Tp >`

Base class for \_Tp dependent member functions.

Definition at line 566 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.14 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct` Template Reference

Policy for individual \_\_pool objects.

Inherits `__per_type_pool_base< _Tp, _PoolTp, _Thread >`.

### 5.14.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread> struct
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
```

Policy for individual \_\_pool objects.

Definition at line 551 of file mt\_allocator.h.

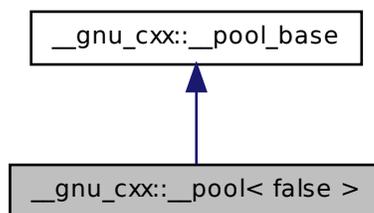
The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.15 \_\_gnu\_cxx::\_\_pool< false > Class Template Reference

Specialization for single thread.

Inheritance diagram for \_\_gnu\_cxx::\_\_pool< false >:



### Public Types

- typedef unsigned short int **\_Binmap\_type**

### Public Member Functions

- **\_\_pool** (const \_\_pool\_base::\_Tune &\_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &, \_Block\_record \*, size\_t)

- `bool _M_check_threshold (size_t __bytes)`
- `void _M_destroy () throw ()`
- `size_t _M_get_align ()`
- `const _Bin_record & _M_get_bin (size_t __which)`
- `size_t _M_get_binmap (size_t __bytes)`
- `const _Tune & _M_get_options () const`
- `size_t _M_get_thread_id ()`
- `void _M_initialize_once ()`
- `void _M_reclaim_block (char * __p, size_t __bytes) throw ()`
- `char * _M_reserve_block (size_t __bytes, const size_t __thread_id)`
- `void _M_set_options (_Tune __t)`

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

#### 5.15.1 Detailed Description

`template<> class __gnu_cxx::__pool< false >`

Specialization for single thread.

Definition at line 192 of file `mt_allocator.h`.

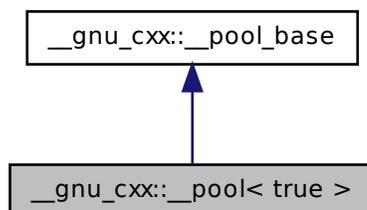
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

### 5.16 `__gnu_cxx::__pool< true >` Class Template Reference

Specialization for thread enabled, via `gthreads.h`.

Inheritance diagram for `__gnu_cxx::__pool< true >`:



## Public Types

- typedef unsigned short int **\_Binmap\_type**

## Public Member Functions

- **\_\_pool** (const `__pool_base::_Tune &__tune`)
- **\_\_attribute\_\_** ((`__const__`)) void **\_M\_destroy\_thread\_key**(void \*) throw ()
- void **\_M\_adjust\_freelist** (const `_Bin_record &__bin`, `_Block_record *__block`, `size_t __thread_id`)
- bool **\_M\_check\_threshold** (`size_t __bytes`)
- void **\_M\_destroy** () throw ()
- `size_t` **\_M\_get\_align** ()
- const `_Bin_record &` **\_M\_get\_bin** (`size_t __which`)
- `size_t` **\_M\_get\_binmap** (`size_t __bytes`)
- const `_Tune &` **\_M\_get\_options** () const
- `size_t` **\_M\_get\_thread\_id** ()
- void **\_M\_initialize** (`__destroy_handler`)
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (`char *__p`, `size_t __bytes`) throw ()
- `char *` **\_M\_reserve\_block** (`size_t __bytes`, const `size_t __thread_id`)
- void **\_M\_set\_options** (`_Tune __t`)

## Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

### 5.16.1 Detailed Description

`template<> class __gnu_cxx::__pool< true >`

Specialization for thread enabled, via `gthreads.h`.

Definition at line 259 of file `mt_allocator.h`.

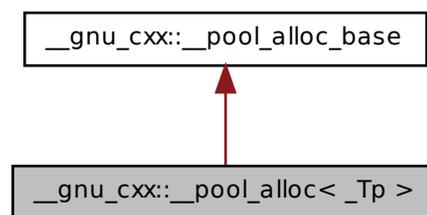
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.17 `__gnu_cxx::__pool_alloc<_Tp>` Class Template Reference

Allocator using a memory pool with a single lock.

Inheritance diagram for `__gnu_cxx::__pool_alloc<_Tp>`:



## Public Types

- `typedef const _Tp * const_pointer`

- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#) &) throw ()
- template<typename \_Tp1 >  
**\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#)< \_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- pointer **allocate** (size\_type \_\_n, const void \*|=0)
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

## Private Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

## Private Member Functions

- **\_\_attribute**\_\_ ((\_\_const\_\_)) \_Obj \*volatile \*\_M\_get\_free\_list(size\_t \_\_bytes) throw ()
- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

## Static Private Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [**\_S\_free\_list\_size**]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

### 5.17.1 Detailed Description

```
template<typename _Tp> class __gnu_cxx::__pool_alloc< _Tp >
```

Allocator using a memory pool with a single lock.

Definition at line 122 of file `pool_allocator.h`.

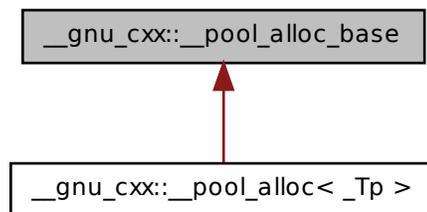
The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 5.18 `__gnu_cxx::__pool_alloc_base` Class Reference

Base class for `__pool_alloc`.

Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



### Protected Types

- enum { `_S_align` }
- enum { `_S_max_bytes` }
- enum { `_S_free_list_size` }

### Protected Member Functions

- `__attribute__((__const__))` `_Obj *volatile *_M_get_free_list(size_t __bytes) throw ()`

- `char * _M_allocate_chunk (size_t __n, int &__nobjs)`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

### Static Protected Attributes

- `static char * _S_end_free`
- `static _Obj *volatile _S_free_list [_S_free_list_size]`
- `static size_t _S_heap_size`
- `static char * _S_start_free`

#### 5.18.1 Detailed Description

Base class for [\\_\\_pool\\_alloc](#). Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size > `_S_max_bytes`, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

Definition at line 74 of file `pool_allocator.h`.

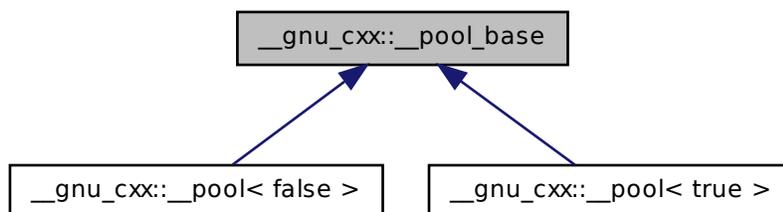
The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

### 5.19 `__gnu_cxx::__pool_base` Struct Reference

Base class for pool object.

Inheritance diagram for `__gnu_cxx::__pool_base`:



## Public Types

- typedef unsigned short int **`_Binmap_type`**

## Public Member Functions

- `__pool_base` (const `_Tune` & `__options`)
- `bool _M_check_threshold` (size\_t `__bytes`)
- `size_t _M_get_align` ()
- `size_t _M_get_binmap` (size\_t `__bytes`)
- `const _Tune` & `_M_get_options` () const
- `void _M_set_options` (`_Tune` `__t`)

## Protected Attributes

- `_Binmap_type` \* `_M_binmap`
- `bool _M_init`
- `_Tune` `_M_options`

### 5.19.1 Detailed Description

Base class for pool object.

Definition at line 47 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.20 `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility< _CharT, _Traits, _Alloc >`.

### Public Types

- typedef `_Util_Base::_CharT_alloc_type` **`_CharT_alloc_type`**
- typedef `__vstring_utility< _CharT, _Traits, _Alloc >` **`_Util_Base`**
- typedef `_Alloc` **`allocator_type`**
- typedef `_CharT_alloc_type::size_type` **`size_type`**
- typedef `_Traits` **`traits_type`**
- typedef `_Traits::char_type` **`value_type`**

### Public Member Functions

- `__rc_string_base` (const `_Alloc` &`_a`)
- `__rc_string_base` (const `__rc_string_base` &`_rcs`)
- `__rc_string_base` (`__rc_string_base` &&`_rcs`)
- `__rc_string_base` (size\_type `_n`, `_CharT` `_c`, const `_Alloc` &`_a`)
- template<typename `_InputIterator` >  
`__rc_string_base` (`_InputIterator` `__beg`, `_InputIterator` `__end`, const `_Alloc` &`_a`)
- void `_M_assign` (const `__rc_string_base` &`_rcs`)
- size\_type `_M_capacity` () const
- void `_M_clear` ()
- bool `_M_compare` (const `__rc_string_base` &) const
- template<>  
bool `_M_compare` (const `__rc_string_base` &`_rcs`) const
- template<>  
bool `_M_compare` (const `__rc_string_base` &`_rcs`) const
- `_CharT` \* `_M_data` () const
- void `_M_erase` (size\_type `__pos`, size\_type `_n`)
- `allocator_type` & `_M_get_allocator` ()
- const `allocator_type` & `_M_get_allocator` () const
- bool `_M_is_shared` () const
- void `_M_leak` ()
- size\_type `_M_length` () const

- `size_type _M_max_size () const`
- `void _M_mutate (size_type __pos, size_type __len1, const _CharT *__s, size_type __len2)`
- `void _M_reserve (size_type __res)`
- `void _M_set_leaked ()`
- `void _M_set_length (size_type __n)`
- `void _M_swap (__rc_string_base &__rcs)`
- `template<typename _InIterator > _CharT * _S_construct (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)`

### Protected Types

- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base > > __const_rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base > > __const_sso_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base > > __rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base > > __sso_iterator`
- `typedef _CharT_alloc_type::const_pointer const_pointer`
- `typedef _CharT_alloc_type::difference_type difference_type`
- `typedef _CharT_alloc_type::pointer pointer`

### Static Protected Member Functions

- `static void _S_assign (_CharT *__d, size_type __n, _CharT __c)`
- `static int _S_compare (size_type __n1, size_type __n2)`
- `static void _S_copy (_CharT *__d, const _CharT *__s, size_type __n)`
- `static void _S_copy_chars (_CharT *__p, _CharT *__k1, _CharT *__k2)`
- `static void _S_copy_chars (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)`
- `template<typename _Iterator > static void _S_copy_chars (_CharT *__p, _Iterator __k1, _Iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)`

- static void `_S_copy_chars` (`_CharT *__p`, `const _CharT *__k1`, `const _CharT *__k2`)
- static void `_S_move` (`_CharT *__d`, `const _CharT *__s`, `size_type __n`)

### 5.20.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`

Documentation? What's that? Nathan Myers <[ncm@cantrip.org](mailto:ncm@cantrip.org)>.

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[__rc_string_base<char_type>]         _M_capacity
_M_dataplus                          _M_refcount
_M_p ----->                        unnamed array of char_type

```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 5.21 `__gnu_cxx::__scoped_lock` Class Reference

Scoped lock idiom.

### Public Types

- typedef `__mutex` `__mutex_type`

### Public Member Functions

- `__scoped_lock` (`__mutex_type` &`__name`)

#### 5.21.1 Detailed Description

Scoped lock idiom.

Definition at line 241 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _-Alloc, _Base>` Class Template Reference

Template class [\\_\\_versa\\_string](#).

Data structure managing sequences of characters and character-like objects.

Inherits `_Base<_CharT, _Traits, _Alloc>`.

### Public Types

- typedef `_Alloc` `allocator_type`
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` `const_iterator`
- typedef `_CharT_alloc_type::const_pointer` `const_pointer`
- typedef `_CharT_alloc_type::const_reference` `const_reference`
- typedef `std::reverse_iterator< const_iterator >` `const_reverse_iterator`
- typedef `_CharT_alloc_type::difference_type` `difference_type`
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` `iterator`
- typedef `_CharT_alloc_type::pointer` `pointer`

- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- `__versa_string ()`
- `__versa_string (const _Alloc &__a)`
- `__versa_string (__versa_string &&__str)`
- `__versa_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `__versa_string (const _CharT *__s, const _Alloc &__a=_Alloc())`
- `__versa_string (std::initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- `__versa_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `template<class _InputIterator >`  
`__versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `__versa_string (const __versa_string &__str)`
- `__versa_string (const __versa_string &__str, size_type __pos, size_type __n=npos)`
- `__versa_string (const __versa_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `~__versa_string ()`
- `__versa_string & append (const __versa_string &__str)`
- `__versa_string & append (const __versa_string &__str, size_type __pos, size_type __n)`
- `__versa_string & append (const _CharT *__s, size_type __n)`
- `__versa_string & append (const _CharT *__s)`
- `__versa_string & append (size_type __n, _CharT __c)`
- `__versa_string & append (std::initializer_list< _CharT > __l)`
- `template<class _InputIterator >`  
`__versa_string & append (_InputIterator __first, _InputIterator __last)`
- `__versa_string & assign (const _CharT *__s)`
- `__versa_string & assign (size_type __n, _CharT __c)`
- `template<class _InputIterator >`  
`__versa_string & assign (_InputIterator __first, _InputIterator __last)`
- `__versa_string & assign (std::initializer_list< _CharT > __l)`
- `__versa_string & assign (const __versa_string &__str)`
- `__versa_string & assign (__versa_string &&__str)`
- `__versa_string & assign (const __versa_string &__str, size_type __pos, size_type __n)`

- `__versa_string & assign` (const `_CharT *__s`, `size_type __n`)
- `const_reference at` (`size_type __n`) const
- `reference at` (`size_type __n`)
- `reference back` ()
- `const_reference back` () const
- `iterator begin` ()
- `const_iterator begin` () const
- `const _CharT * c_str` () const
- `size_type capacity` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- `void clear` ()
- `int compare` (const `_CharT *__s`) const
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`) const
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`, `size_type __pos2`, `size_type __n2`) const
- `int compare` (`size_type __pos1`, `size_type __n1`, const `__versa_string &__str`, `size_type __pos2`, `size_type __n2`) const
- `int compare` (`size_type __pos`, `size_type __n`, const `__versa_string &__str`) const
- `int compare` (const `__versa_string &__str`) const
- `size_type copy` (`_CharT *__s`, `size_type __n`, `size_type __pos=0`) const
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- `const _CharT * data` () const
- `bool empty` () const
- `iterator end` ()
- `const_iterator end` () const
- `__versa_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `iterator erase` (`iterator __position`)
- `iterator erase` (`iterator __first`, `iterator __last`)
- `size_type find` (`_CharT __c`, `size_type __pos=0`) const
- `size_type find` (const `_CharT *__s`, `size_type __pos`, `size_type __n`) const
- `size_type find` (const `__versa_string &__str`, `size_type __pos=0`) const
- `size_type find` (const `_CharT *__s`, `size_type __pos=0`) const
- `size_type find_first_not_of` (const `_CharT *__s`, `size_type __pos`, `size_type __n`) const
- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) const
- `size_type find_first_not_of` (const `__versa_string &__str`, `size_type __pos=0`) const
- `size_type find_first_not_of` (const `_CharT *__s`, `size_type __pos=0`) const
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) const
- `size_type find_first_of` (const `_CharT *__s`, `size_type __pos`, `size_type __n`) const

- size\_type [find\\_first\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_first\\_of](#) (const \_\_versa\_string &\_\_str, size\_type \_\_pos=0) const
- size\_type [find\\_last\\_not\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_not\\_of](#) (const \_\_versa\_string &\_\_str, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_not\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_of](#) (const \_\_versa\_string &\_\_str, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=npos) const
- size\_type [find\\_last\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator >  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, [std::initializer\\_list](#)< \_CharT > \_\_l)
- \_\_versa\_string & [insert](#) (size\_type \_\_pos1, const \_\_versa\_string &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- \_\_versa\_string & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- \_\_versa\_string & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s)
- iterator [insert](#) (iterator \_\_p, \_CharT \_\_c)
- \_\_versa\_string & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- \_\_versa\_string & [insert](#) (size\_type \_\_pos1, const \_\_versa\_string &\_\_str)
- size\_type [length](#) () const
- size\_type [max\\_size](#) () const
- \_\_versa\_string & [operator+=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- \_\_versa\_string & [operator+=](#) (const \_\_versa\_string &\_\_str)
- \_\_versa\_string & [operator+=](#) (\_CharT \_\_c)
- \_\_versa\_string & [operator+=](#) (const \_CharT \*\_\_s)
- \_\_versa\_string & [operator=](#) (\_\_versa\_string &&\_\_str)
- \_\_versa\_string & [operator=](#) (const \_CharT \*\_\_s)
- \_\_versa\_string & [operator=](#) (\_CharT \_\_c)
- \_\_versa\_string & [operator=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- \_\_versa\_string & [operator=](#) (const \_\_versa\_string &\_\_str)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const
- reference [operator\[\]](#) (size\_type \_\_pos)
- void [push\\_back](#) (\_CharT \_\_c)
- [reverse\\_iterator](#) [rbegin](#) ()

- `const_reverse_iterator rbegin ()` const
- `reverse_iterator rend ()`
- `const_reverse_iterator rend ()` const
- `__versa_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `__versa_string & replace (iterator __i1, iterator __i2, const __versa_string & __str)`
- `__versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT * __s)`
- `__versa_string & replace (size_type __pos, size_type __n, const __versa_string & __str)`
- `__versa_string & replace (iterator __i1, iterator __i2, std::initializer_list< _CharT > __l)`
- `template<class _InputIterator > __versa_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, iterator __k1, iterator __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT * __s)`
- `__versa_string & replace (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)`
- `__versa_string & replace (size_type __pos1, size_type __n1, const __versa_string & __str, size_type __pos2, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT * __k1, const _CharT * __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, _CharT * __k1, _CharT * __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n)`
- `void resize (size_type __n, _CharT __c)`
- `size_type rfind (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type rfind (const __versa_string & __str, size_type __pos=npos) const`
- `size_type rfind (const _CharT * __s, size_type __pos=npos) const`
- `size_type rfind (_CharT __c, size_type __pos=npos) const`
- `void shrink_to_fit ()`
- `size_type size () const`
- `__versa_string substr (size_type __pos=0, size_type __n=npos) const`
- `void swap (__versa_string & __s)`

## Static Public Attributes

- static const size\_type `npos`

### 5.22.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template<
typename, typename, typename > class _Base> class __gnu_cxx::__versa_ -
string< _CharT, _Traits, _Alloc, _Base >
```

Template class [\\_\\_versa\\_string](#).

Data structure managing sequences of characters and character-like objects.

Definition at line 52 of file `vstring.h`.

### 5.22.2 Constructor & Destructor Documentation

```
5.22.2.1 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( ) [inline]
```

Default constructor creates an empty string.

Definition at line 130 of file `vstring.h`.

```
5.22.2.2 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const _Alloc & __a ) [inline, explicit]
```

Construct an empty string using allocator *a*.

Definition at line 137 of file `vstring.h`.

```
5.22.2.3 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str ) [inline]
```

Construct string with copy of value of *str*.

**Parameters**

`__str` Source string.

Definition at line 145 of file `vstring.h`.

```
5.22.2.4 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::__versa_string ( __versa_string<_CharT, _Traits, _Alloc, _Base >
&& __str ) [inline]
```

String move constructor.

**Parameters**

`__str` Source string.

The newly-constructed string contains the exact contents of `str`. The contents of `str` are a valid, but unspecified string.

Definition at line 157 of file `vstring.h`.

```
5.22.2.5 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::__versa_string ( std::initializer_list<_CharT > __l, const _Alloc
& __a = _Alloc() ) [inline]
```

Construct string from an initializer list.

**Parameters**

`__l` `std::initializer_list` of characters.

`__a` Allocator to use (default is default allocator).

Definition at line 165 of file `vstring.h`.

```
5.22.2.6 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::__versa_string ( const __versa_string<_CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n = npos )
[inline]
```

Construct string as copy of a substring.

**Parameters**

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy (default remainder).

Definition at line 176 of file `vstring.h`.

```
5.22.2.7 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n, const _Alloc & __a
) [inline]
```

Construct string as copy of a substring.

**Parameters**

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy.
- `__a` Allocator to use.

Definition at line 191 of file `vstring.h`.

```
5.22.2.8 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const _CharT * __s, size_type __n, const _Alloc
& __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

**Parameters**

- `__s` Source character array.
- `__n` Number of characters to copy.
- `__a` Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 208 of file `vstring.h`.

**5.22.2.9** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::__versa_string ( const _CharT * __s, const _Alloc & __a = _Alloc() ) [inline]`

Construct string as copy of a C string.

**Parameters**

- `__s` Source C string.
- `__a` Allocator to use (default is default allocator).

Definition at line 217 of file `vstring.h`.

**5.22.2.10** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::__versa_string ( size_type __n, _CharT __c, const _Alloc & __a = _Alloc() ) [inline]`

Construct string as multiple characters.

**Parameters**

- `__n` Number of characters.
- `__c` Character to use.
- `__a` Allocator to use (default is default allocator).

Definition at line 227 of file `vstring.h`.

**5.22.2.11** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::__versa_string ( _InputIterator __beg, _InputIterator __end, const _Alloc & __a = _Alloc() ) [inline]`

Construct string as copy of a range.

**Parameters**

- `__beg` Start of range.
- `__end` End of range.

`__a` Allocator to use (default is default allocator).

Definition at line 237 of file `vstring.h`.

```
5.22.2.12 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::~~__versa_string( ) [inline]
```

Destroy the string instance.

Definition at line 244 of file `vstring.h`.

### 5.22.3 Member Function Documentation

```
5.22.3.1 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str ) [inline]
```

Append a string to this string.

#### Parameters

`__str` The string to append.

#### Returns

Reference to this string.

Definition at line 676 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `std::getline()`, and `__gnu_cxx::operator+()`.

```
5.22.3.2 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n ) [inline]
```

Append a substring.

### Parameters

- `__str` The string to append.
- `__pos` Index of the first character of `str` to append.
- `__n` The number of characters to append.

### Returns

Reference to this string.

### Exceptions

[\*`std::out\_of\_range`\*](#) if `pos` is not a valid index.

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 693 of file `vstring.h`.

```
5.22.3.3  template<typename _CharT, typename _Traits, typename _Alloc,
          template< typename, typename, typename > class _Base>
          __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,
          _Base >::append ( const _CharT * __s, size_type __n ) [inline]
```

Append a C substring.

### Parameters

- `__s` The C string to append.
- `__n` The number of characters to append.

### Returns

Reference to this string.

Definition at line 705 of file `vstring.h`.

```
5.22.3.4  template<typename _CharT, typename _Traits, typename _Alloc,
          template< typename, typename, typename > class _Base>
          __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,
          _Base >::append ( const _CharT * __s ) [inline]
```

Append a C string.

**Parameters**

`__s` The C string to append.

**Returns**

Reference to this string.

Definition at line 718 of file `vstring.h`.

**5.22.3.5** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( size_type __n, _CharT __c ) [inline]`

Append multiple characters.

**Parameters**

`__n` The number of characters to append.

`__c` The character to use.

**Returns**

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 735 of file `vstring.h`.

**5.22.3.6** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( std::initializer_list< _CharT > __l ) [inline]`

Append an `initializer_list` of characters.

**Parameters**

`__l` The `initializer_list` of characters to append.

**Returns**

Reference to this string.

Definition at line 745 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

**5.22.3.7** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class _Base>  
template<class _InputIterator > __versa_string&  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base  
>::append ( _InputIterator __first, _InputIterator __last )  
[inline]`

Append a range of characters.

#### Parameters

`__first` Iterator referencing the first character to append.

`__last` Iterator marking the end of the range.

#### Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 759 of file `vstring.h`.

**5.22.3.8** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base >::assign ( const __versa_string<_CharT, _Traits, _Alloc,  
_Base > & __str ) [inline]`

Set value to contents of another string.

#### Parameters

`__str` Source string to use.

#### Returns

Reference to this string.

Definition at line 782 of file `vstring.h`.

**5.22.3.9** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base >::assign ( __versa_string<_CharT, _Traits, _Alloc, _Base >  
&& __str ) [inline]`

Set value to contents of another string.

**Parameters**

`__str` Source string to use.

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 798 of file `vstring.h`.

References `std::swap()`.

```
5.22.3.10 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::assign ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __str, size_type __pos, size_type __n )
[inline]
```

Set value to a substring of a string.

**Parameters**

`__str` The string to use.

`__pos` Index of the first character of `str`.

`__n` Number of characters to use.

**Returns**

Reference to this string.

**Exceptions**

[`std::out\_of\_range`](#) if `__pos` is not a valid index.

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 819 of file `vstring.h`.

**5.22.3.11** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::assign ( const _CharT * __s, size_type __n )  
[inline]`

Set value to a C substring.

**Parameters**

- `__s` The C string to use.
- `__n` Number of characters to use.

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 836 of file `vstring.h`.

**5.22.3.12** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::assign ( const _CharT * __s ) [inline]`

Set value to contents of a C string.

**Parameters**

- `__s` The C string to use.

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 852 of file `vstring.h`.

**5.22.3.13** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::assign ( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

**Parameters**

- `__n` Length of the resulting string.
- `__c` The character to use.

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 869 of file `vstring.h`.

```
5.22.3.14 template<typename _CharT, typename _Traits, typename
        _Alloc, template< typename, typename, typename > class
        _Base> template<class _InputIterator > __versa_string&
        __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
        >::assign ( _InputIterator __first, _InputIterator __last )
        [inline]
```

Set value to a range of characters.

**Parameters**

- `__first` Iterator referencing the first character to append.
- `__last` Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 883 of file `vstring.h`.

```
5.22.3.15 template<typename _CharT, typename _Traits, typename _Alloc,
        template< typename, typename, typename > class _Base>
        __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
        _Alloc, _Base >::assign ( std::initializer_list< _CharT > __l )
        [inline]
```

Set value to an `initializer_list` of characters.

**Parameters**

- `__l` The `initializer_list` of characters to assign.

### Returns

Reference to this string.

Definition at line 893 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

**5.22.3.16** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at ( size_type __n ) const [inline]`

Provides access to the data contained in the string.

### Parameters

`__n` The index of the character to access.

### Returns

Read-only (const) reference to the character.

### Exceptions

*[std::out\\_of\\_range](#)* If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 567 of file `vstring.h`.

**5.22.3.17** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at ( size_type __n ) [inline]`

Provides access to the data contained in the string.

### Parameters

`__n` The index of the character to access.

### Returns

Read/write reference to the character.

**Exceptions**

*std::out\_of\_range* If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 586 of file `vstring.h`.

**5.22.3.18** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 624 of file `vstring.h`.

**5.22.3.19** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 616 of file `vstring.h`.

**5.22.3.20** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin ( ) [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 310 of file `vstring.h`.

**5.22.3.21** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 321 of file `vstring.h`.

**5.22.3.22** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::c_str ( ) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1485 of file `vstring.h`.

**5.22.3.23** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::capacity ( ) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 478 of file `vstring.h`.

**5.22.3.24** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 385 of file `vstring.h`.

**5.22.3.25** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 393 of file `vstring.h`.

**5.22.3.26** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear ( ) [inline]`

Erases the string, making it empty.

Definition at line 506 of file `vstring.h`.

**5.22.3.27** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) const [inline]`

Compare to a string.

#### Parameters

`__str` String to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1904 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

**5.22.3.28** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare ( size_type __pos, size_type __n, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) const`

Compare substring to a string.

### Parameters

`__pos` Index of first character of substring.

`__n` Number of characters in substring.

`__str` String to compare against.

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 458 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.29** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare ( size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2 ) const`

Compare substring to a substring.

### Parameters

`__pos1` Index of first character of substring.

`__n1` Number of characters in substring.

`__str` String to compare against.

`__pos2` Index of first character of substring of `str`.

`__n2` Number of characters in substring of `str`.

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer `< 0` if this substring is ordered before the substring of `__str`, `0` if their values are equivalent, or `> 0` if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 475 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

**5.22.3.30** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare ( const _CharT * __s ) const`

Compare to a C string.

#### Parameters

`__s` C string to compare against.

#### Returns

Integer `< 0`, `0`, or `> 0`.

Returns an integer `< 0` if this string is ordered before `__s`, `0` if their values are equivalent, or `> 0` if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 494 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.31 template<typename _CharT, typename _Traits, typename _Alloc ,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare ( size_type __pos, size_type __n1, const _CharT * __s
) const
```

Compare substring to a C string.

#### Parameters

- `__pos` Index of first character of substring.
- `__n1` Number of characters in substring.
- `__s` C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `r1len` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,r1len)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 510 of file `vstring.tcc`.

References `std::min()`.

```
5.22.3.32 template<typename _CharT, typename _Traits, typename _Alloc ,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare ( size_type __pos, size_type __n1, const _CharT * __s,
size_type __n2 ) const
```

Compare substring against a character array.

#### Parameters

- `__pos1` Index of first character of substring.
- `__n1` Number of characters in substring.
- `__s` character array to compare against.
- `__n2` Number of characters of `s`.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `r1en` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 527 of file `vstring.tcc`.

References `std::min()`.

```
5.22.3.33  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy
            ( _CharT * __s, size_type __n, size_type __pos = 0 ) const
```

Copy substring into C string.

**Parameters**

`__s` C string to copy value into.

`__n` Number of characters to copy.

`__pos` Index of first character to copy.

**Returns**

Number of characters actually copied

**Exceptions**

[`std::out\_of\_range`](#) If `pos > size()`.

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 253 of file `vstring.tcc`.

**5.22.3.34** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 402 of file `vstring.h`.

**5.22.3.35** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 411 of file `vstring.h`.

**5.22.3.36** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data ( ) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1495 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

**5.22.3.37** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::empty ( ) const [inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 514 of file `vstring.h`.

**5.22.3.38** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 340 of file `vstring.h`.

**5.22.3.39** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 329 of file `vstring.h`.

**5.22.3.40** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase ( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

#### Parameters

`__pos` Index of first character to remove (default 0).

`__n` Number of characters to remove (default remainder).

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If `__pos` is beyond the end of this string.

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1088 of file `vstring.h`.

Referenced by `std::getline()`.

**5.22.3.41** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase ( iterator __position ) [inline]`

Remove one character.

#### Parameters

`__position` Iterator referencing the character to remove.

#### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1104 of file `vstring.h`.

**5.22.3.42** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase ( iterator __first, iterator __last ) [inline]`

Remove a range of characters.

#### Parameters

`__first` Iterator referencing the first character to remove.

`__last` Iterator referencing the end of the range.

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1125 of file `vstring.h`.

**5.22.3.43** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a C substring.

**Parameters**

- `__s` C string to locate.
- `__pos` Index of character to search from.
- `__n` Number of characters from `__s` to search for.

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 268 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

**5.22.3.44** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline]`

Find position of a string.

**Parameters**

- `__str` String to locate.
- `__pos` Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1531 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

**5.22.3.45** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find ( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

#### Parameters

`__s` C string to locate.

`__pos` Index of character to search from (default 0).

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1545 of file `vstring.h`.

**5.22.3.46** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find ( _CharT __c, size_type __pos = 0 ) const`

Find position of a character.

#### Parameters

`__c` Character to locate.

`__pos` Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 292 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```

5.22.3.47 template<typename _CharT, typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
  __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
  >::find_first_not_of( const _CharT * __s, size_type __pos,
  size_type __n ) const

```

Find position of a character not in C substring.

#### Parameters

- \_\_s* C string containing characters to avoid.
- \_\_pos* Index of character to search from.
- \_\_n* Number of characters from *s* to consider.

#### Returns

Index of first occurrence.

Starting from *\_\_pos*, searches forward for a character not contained in the first *\_\_n* characters of *\_\_s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 390 of file *vstring.tcc*.

References *\_\_gnu\_cxx::\_\_versa\_string*< *\_CharT*, *\_Traits*, *\_Alloc*, *\_Base* >::*npos*, and *\_\_gnu\_cxx::\_\_versa\_string*< *\_CharT*, *\_Traits*, *\_Alloc*, *\_Base* >::*size()*.

```

5.22.3.48 template<typename _CharT, typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
  __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
  >::find_first_not_of( _CharT __c, size_type __pos = 0 ) const

```

Find position of a different character.

#### Parameters

- \_\_c* Character to avoid.
- \_\_pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *\_\_pos*, searches forward for a character other than *\_\_c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

## 5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

761

Definition at line 403 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.22.3.49** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = 0 ) const [inline]`

Find position of a character not in string.

### Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1759 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

**5.22.3.50** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a character not in C string.

### Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1789 of file `vstring.h`.

```
5.22.3.51 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_of( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos = 0 ) const [inline]
```

Find position of a character of string.

**Parameters**

`__str` String containing characters to locate.

`__pos` Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1634 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`.

```
5.22.3.52 template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_of( const _CharT * __s, size_type __pos, size_type
__n ) const
```

Find position of a character of C substring.

**Parameters**

- `__s` String containing characters to locate.
- `__pos` Index of character to search from.
- `__n` Number of characters from `s` to search for.

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 351 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.53** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of ( const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a character of C string.

**Parameters**

- `__s` String containing characters to locate.
- `__pos` Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1663 of file `vstring.h`.

**5.22.3.54** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of ( _CharT __c, size_type __pos = 0 ) const [inline]`

Find position of a character.

**Parameters**

- `__c` Character to locate.
- `__pos` Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1682 of file `vstring.h`.

```
5.22.3.55  template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_not_of ( _CharT __c, size_type __pos = npos ) const
```

Find last position of a different character.

**Parameters**

- `__c` Character to avoid.
- `__pos` Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 437 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.56  template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_not_of ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos = npos ) const [inline]
```

Find last position of a character not in string.

### Parameters

- `__str` String containing characters to avoid.
- `__pos` Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1820 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`.

**5.22.3.57** `template<typename _CharT, typename _Traits, typename _Alloc, typename _Base> class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a character not in C substring.

### Parameters

- `__s` C string containing characters to avoid.
- `__pos` Index of character to search back from.
- `__n` Number of characters from `s` to consider.

### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 415 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```

5.22.3.58 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_not_of ( const _CharT * __s, size_type __pos = npos )
const [inline]

```

Find last position of a character not in C string.

#### Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1851 of file `vstring.h`.

```

5.22.3.59 template<typename _CharT, typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_of ( const _CharT * __s, size_type __pos, size_type
__n ) const

```

Find last position of a character of C substring.

#### Parameters

`__s` C string containing characters to locate.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to search for.

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 368 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.60** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of ( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos ) const [inline]`

Find last position of a character of string.

#### Parameters

`__str` String containing characters to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1697 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`.

**5.22.3.61** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of ( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a character of C string.

#### Parameters

`__s` C string containing characters to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1726 of file `vstring.h`.

**5.22.3.62** `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_of ( _CharT __c, size_type __pos = npos ) const
[inline]`

Find last position of a character.

#### Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1745 of file `vstring.h`.

**5.22.3.63** `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front
( ) [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 600 of file `vstring.h`.

**5.22.3.64** `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 608 of file `vstring.h`.

5.22.3.65 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::get_allocator ( ) const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1502 of file `vstring.h`.

5.22.3.66 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert ( size_type __pos, const _CharT * __s, size_type __n ) [inline]`

Insert a C substring.

#### Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

`__n` The number of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

*[std::out\\_of\\_range](#)* If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1002 of file `vstring.h`.

5.22.3.67 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert ( iterator __p, _CharT __c ) [inline]`

Insert one character.

**Parameters**

- `__p` Iterator referencing position in string to insert at.
- `__c` The character to insert.

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1063 of file `vstring.h`.

**5.22.3.68** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert ( size_type __pos1, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Insert value of a string.

**Parameters**

- `__pos1` Iterator referencing location in string to insert at.
- `__str` The string to insert.

**Returns**

Reference to this string.

**Exceptions**

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 956 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**5.22.3.69** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( iterator __p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

`__p` Iterator referencing location in string to insert at.

`__n` Number of characters to insert

`__c` The character to insert.

#### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 911 of file `vstring.h`.

**5.22.3.70** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert ( iterator __p, std::initializer_list<_CharT> __l ) [inline]`

Insert an `initializer_list` of characters.

#### Parameters

`__p` Iterator referencing location in string to insert at.

`__l` The `initializer_list` of characters to insert.

#### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Definition at line 939 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

**5.22.3.71** `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename > class _Base>
template<class _InputIterator > void __gnu_cxx::__versa_string<
_CharT, _Traits, _Alloc, _Base >::insert ( iterator __p,
_InputIterator __beg, _InputIterator __end ) [inline]`

Insert a range of characters.

#### Parameters

`__p` Iterator referencing location in string to insert at.  
`__beg` Start of range.  
`__end` End of range.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 928 of file `vstring.h`.

**5.22.3.72** `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::insert ( size_type __pos, size_type __n, _CharT
__c ) [inline]`

Insert multiple characters.

#### Parameters

`__pos` Index in string to insert at.  
`__n` Number of characters to insert  
`__c` The character to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

*[std::out\\_of\\_range](#)* If `__pos` is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1045 of file `vstring.h`.

```
5.22.3.73 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base >::insert ( size_type __pos1, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2,
size_type __n ) [inline]
```

Insert a substring.

#### Parameters

- `__pos1` Iterator referencing location in string to insert at.
- `__str` The string to insert.
- `__pos2` Start of characters in `str` to insert.
- `__n` Number of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

- `std::length_error` If new length exceeds `max_size()`.
- `std::out_of_range` If `__pos1 > size()` or `__pos2 > __str.size()`.

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 979 of file `vstring.h`.

```
5.22.3.74 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base >::insert ( size_type __pos, const _CharT * __s )
[inline]
```

Insert a C string.

**Parameters**

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

**Returns**

Reference to this string.

**Exceptions**

`std::length_error` If new length exceeds `max_size()`.

`std::out_of_range` If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1021 of file `vstring.h`.

**5.22.3.75** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length ( ) const [inline]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 426 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**5.22.3.76** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size ( ) const [inline]`

Returns the `size()` of the largest possible string.

Definition at line 431 of file `vstring.h`.

Referenced by `std::getline()`.

```
5.22.3.77 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base >::operator+=( const __versa_string<_CharT,
_Traits, _Alloc, _Base > & __str ) [inline]
```

Append a string to this string.

#### Parameters

`__str` The string to append.

#### Returns

Reference to this string.

Definition at line 635 of file `vstring.h`.

```
5.22.3.78 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base >::operator+=( const _CharT * __s ) [inline]
```

Append a C string.

#### Parameters

`__s` The C string to append.

#### Returns

Reference to this string.

Definition at line 644 of file `vstring.h`.

```
5.22.3.79 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base >::operator+=( _CharT __c ) [inline]
```

Append a character.

#### Parameters

`__c` The character to append.

**Returns**

Reference to this string.

Definition at line 653 of file `vstring.h`.

**5.22.3.80** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=( std::initializer_list< _CharT > __l ) [inline]`

Append an `initializer_list` of characters.

**Parameters**

`__l` The `initializer_list` of characters to be appended.

**Returns**

Reference to this string.

Definition at line 666 of file `vstring.h`.

**5.22.3.81** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=( const _CharT * __s ) [inline]`

Copy contents of `__s` into this string.

**Parameters**

`__s` Source null-terminated string.

Definition at line 287 of file `vstring.h`.

**5.22.3.82** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=( std::initializer_list< _CharT > __l ) [inline]`

Set value to string constructed from initializer list.

**Parameters**

`__l` [std::initializer\\_list](#).

Definition at line 275 of file `vstring.h`.

**5.22.3.83** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=( __versa_string<_CharT, _Traits, _Alloc, _Base > && __str ) [inline]`

String move assignment operator.

**Parameters**

`__str` Source string.

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 263 of file `vstring.h`.

References `std::swap()`.

**5.22.3.84** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=( _CharT __c ) [inline]`

Set value to string of length 1.

**Parameters**

`__c` Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 298 of file `vstring.h`.

**5.22.3.85** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=( const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str ) [inline]`

Assign the value of `str` to this string.

**Parameters**

*\_\_str* Source string.

Definition at line 251 of file `vstring.h`.

**5.22.3.86** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

**Parameters**

*\_\_pos* The index of the character to access.

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 529 of file `vstring.h`.

**5.22.3.87** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] ( size_type __pos ) [inline]`

Subscript access to the data contained in the string.

**Parameters**

*\_\_pos* The index of the character to access.

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 546 of file `vstring.h`.

**5.22.3.88** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base  
>::push_back( _CharT __c ) [inline]`

Append a single character.

#### Parameters

`__c` Character to append.

Definition at line 767 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

**5.22.3.89** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base >::rbegin( ) [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 349 of file `vstring.h`.

**5.22.3.90** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base >::rbegin( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 358 of file `vstring.h`.

**5.22.3.91** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base >::rend( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 376 of file `vstring.h`.

**5.22.3.92** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 367 of file `vstring.h`.

**5.22.3.93** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, std::initializer_list< _CharT > __l ) [inline]`

Replace range of characters with `initializer_list`.

#### Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__l` The `initializer_list` of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1421 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

**5.22.3.94** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str )`  
[inline]

Replace range of characters with string.

#### Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__str` String value to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1270 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**5.22.3.95** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str )`  
[inline]

Replace characters with value from another string.

#### Parameters

- `__pos` Index of first character to replace.

`__n` Number of characters to be replaced.

`__str` String to insert.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If `__pos` is beyond the end of this string.

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1153 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.22.3.96** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c ) [inline]`

Replace characters with multiple characters.

### Parameters

`__pos` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__n2` Number of characters to insert.

`__c` Character to insert.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If `__pos > size()`.

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1252 of file `vstring.h`.

**5.22.3.97** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline]`

Replace range of characters with multiple characters.

#### Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__n` Number of characters to insert.

`__c` Character to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1330 of file `vstring.h`.

**5.22.3.98** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, const _CharT * __s ) [inline]`

Replace range of characters with C string.

**Parameters**

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__s` C string value to insert.

**Returns**

Reference to this string.

**Exceptions**

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1309 of file `vstring.h`.

**5.22.3.99** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

**Parameters**

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__k1` Iterator referencing start of range to insert.
- `__k2` Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1353 of file `vstring.h`.

**5.22.3.100** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) [inline]`

Replace characters with value of a C substring.

#### Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__s` C string to insert.
- `__n2` Number of characters from `__s` to use.

#### Returns

Reference to this string.

#### Exceptions

- `std::out_of_range` If `__pos1 > size()`.
- `std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1204 of file `vstring.h`.

**5.22.3.101** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( size_type __pos, size_type __n1, const _CharT * __s ) [inline]`

Replace characters with value of a C string.

#### Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__s` C string to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If `__pos > size()`.

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1228 of file `vstring.h`.

```
5.22.3.102 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
    _versa_string& __gnu_cxx::_versa_string< _CharT, _Traits,
    _Alloc, _Base >::replace ( size_type __pos1, size_type __n1,
    const _versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2, size_type __n2 ) [inline]
```

Replace characters with value from another string.

**Parameters**

`__pos1` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__str` String to insert.

`__pos2` Index of first character of `str` to use.

`__n2` Number of characters from `str` to use.

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If `__pos1 > size()` or `__pos2 > str.size()`.

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1176 of file `vstring.h`.

**5.22.3.103** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2, const _CharT * __s, size_type __n ) [inline]`

Replace range of characters with C substring.

#### Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__s` C string value to insert.

`__n` Number of characters from `s` to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1288 of file `vstring.h`.

**5.22.3.104** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::reserve ( size_type __res_arg = 0 ) [inline]`

Attempt to preallocate enough memory for specified number of characters.

#### Parameters

`__res_arg` Number of characters required.

#### Exceptions

*[std::length\\_error](#)* If `__res_arg` exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 499 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

```
5.22.3.105 template<typename _CharT, typename _Traits , typename _Alloc ,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize ( size_type __n, _CharT __c )
```

Resizes the string to the specified number of characters.

#### Parameters

`__n` Number of characters the string should contain.

`__c` Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 49 of file `vstring.tcc`.

```
5.22.3.106 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize ( size_type __n ) [inline]
```

Resizes the string to the specified number of characters.

#### Parameters

`__n` Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 458 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

**5.22.3.107** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

#### Parameters

`__s` C string to locate.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to search for.

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 311 of file `vstring.tcc`.

References `std::min()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.22.3.108** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind ( _CharT __c, size_type __pos = npos ) const`

Find last position of a character.

#### Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 333 of file vstring.tcc.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
5.22.3.109 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::rfind ( const _CharT * __s, size_type __pos = npos ) const
[inline]
```

Find last position of a C string.

#### Parameters

`__s` C string to locate.

`__pos` Index of character to start search at (default end).

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1604 of file vstring.h.

```
5.22.3.110 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::rfind ( const __versa_string<_CharT, _Traits, _Alloc, _Base >
& __str, size_type __pos = npos ) const [inline]
```

Find last position of a string.

#### Parameters

`__str` String to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

## 5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

791

Definition at line 1575 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.22.3.111** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 464 of file `vstring.h`.

**5.22.3.112** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size( ) const [inline]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 420 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.22.3.113** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr( size_type __pos = 0, size_type __n = npos ) const [inline]`

Get a substring.

**Parameters**

- `__pos` Index of first character (default 0).
- `__n` Number of characters in substring (default remainder).

**Returns**

The new string.

**Exceptions**

*[std::out\\_of\\_range](#)* If `pos > size()`.

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1883 of file `vstring.h`.

**5.22.3.114** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __s ) [inline]`

Swap contents with another string.

**Parameters**

- `__s` String to swap with.

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1474 of file `vstring.h`.

Referenced by `__gnu_cxx::swap()`.

**5.22.4 Member Data Documentation**

**5.22.4.1** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 77 of file `vstring.h`.

Referenced by `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::_versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

## 5.23 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

### Public Types

- `typedef _ToType::element_type * type`

#### 5.23.1 Detailed Description

```
template<typename _ToType> struct __gnu_cxx::_Caster<_ToType>
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 45 of file `cast.h`.

The documentation for this struct was generated from the following file:

- `cast.h`

## 5.24 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

Mapping from character type to associated types.

## Public Types

- typedef unsigned long **int\_type**
- typedef [std::streamoff](#) **off\_type**
- typedef [std::streampos](#) **pos\_type**
- typedef [std::mbstate\\_t](#) **state\_type**

### 5.24.1 Detailed Description

`template<typename _CharT> struct \_\_gnu\_cxx::\_Char\_types< _CharT >`

Mapping from character type to associated types.

#### Note

This is an implementation class for the generic version of [char\\_traits](#). It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in [char\\_traits](#), can specialize [\\_\\_gnu\\_cxx::\\_Char\\_types](#) while leaving [\\_\\_gnu\\_cxx::char\\_traits](#) alone.

Definition at line 65 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.25 [\\_\\_gnu\\_cxx::\\_ExtPtr\\_allocator](#)< \_Tp > Class Template Reference

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).

## Public Types

- typedef [\\_Pointer\\_adapter](#)< [\\_Relative\\_pointer\\_impl](#)< const \_Tp > > **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef [std::ptrdiff\\_t](#) **difference\_type**
- typedef [\\_Pointer\\_adapter](#)< [\\_Relative\\_pointer\\_impl](#)< \_Tp > > **pointer**
- typedef \_Tp & **reference**

- typedef `std::size_t` `size_type`
- typedef `_Tp` `value_type`

## Public Member Functions

- `_ExtPtr_allocator` (const `_ExtPtr_allocator` &\_\_rarg) throw ()
- template<typename `_Up` >  
`_ExtPtr_allocator` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg) throw ()
- const `std::allocator`< `_Tp` > & `_M_getUnderlyingImp` () const
- `pointer address` (reference `__x`) const
- `const_pointer address` (const\_reference `__x`) const
- `pointer allocate` (size\_type `__n`, void \*`__hint`=0)
- void `construct` (`pointer` `__p`, const `_Tp` &`__val`)
- template<typename... `_Args` >  
void `construct` (`pointer` `__p`, `_Args` &&...`__args`)
- void `deallocate` (`pointer` `__p`, size\_type `__n`)
- void `destroy` (`pointer` `__p`)
- size\_type `max_size` () const throw ()
- template<typename `_Up` >  
bool `operator!=` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg)
- bool `operator!=` (const `_ExtPtr_allocator` &\_\_rarg)
- template<typename `_Up` >  
bool `operator==` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg)
- bool `operator==` (const `_ExtPtr_allocator` &\_\_rarg)

## Friends

- template<typename `_Up` >  
void `swap` (`_ExtPtr_allocator`< `_Up` > &, `_ExtPtr_allocator`< `_Up` > &)

### 5.25.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_ExtPtr_allocator< _Tp >`

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 52 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

## 5.26 `__gnu_cxx::_Invalid_type` Struct Reference

### 5.26.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of void\*, const void\*, and so on.

Definition at line 205 of file pointer.h.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.27 `__gnu_cxx::_Pointer_adapter< _Storage_policy >` Class Template Reference

Inherits `_Storage_policy`.

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Storage_policy::element_type` **element\_type**
- typedef `std::random_access_iterator_tag` **iterator\_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value\_type**

### Public Member Functions

- `_Pointer_adapter` (`element_type * __arg=0`)
- `_Pointer_adapter` (`const _Pointer_adapter & __arg`)
- `template<typename _Up > _Pointer_adapter` (`const _Pointer_adapter< _Up > & __arg`)
- `template<typename _Up > _Pointer_adapter` (`_Up * __arg`)
- `operator __unspecified_bool_type` () const
- `bool operator!` () const
- `reference operator*` () const
- `_Pointer_adapter & operator++` ()
- `_Pointer_adapter operator++` (`int __unused`)

- `_Pointer_adapter & operator+=` (unsigned short `__offset`)
- `_Pointer_adapter & operator+=` (int `__offset`)
- `_Pointer_adapter & operator+=` (unsigned int `__offset`)
- `_Pointer_adapter & operator+=` (long `__offset`)
- `_Pointer_adapter & operator+=` (unsigned long `__offset`)
- `_Pointer_adapter & operator+=` (short `__offset`)
- `template<typename _Up > std::ptrdiff_t operator-` (const `_Pointer_adapter<_Up > &__rhs`) const
- `_Pointer_adapter operator--` (int)
- `_Pointer_adapter & operator--` ()
- `_Pointer_adapter & operator-=` (unsigned short `__offset`)
- `_Pointer_adapter & operator-=` (short `__offset`)
- `_Pointer_adapter & operator-=` (int `__offset`)
- `_Pointer_adapter & operator-=` (long `__offset`)
- `_Pointer_adapter & operator-=` (unsigned long `__offset`)
- `_Pointer_adapter & operator-=` (unsigned int `__offset`)
- `element_type * operator->` () const
- `_Pointer_adapter & operator=` (const `_Pointer_adapter &__arg`)
- `template<typename _Up > _Pointer_adapter & operator=` (`_Up *__arg`)
- `template<typename _Up > _Pointer_adapter & operator=` (const `_Pointer_adapter<_Up > &__arg`)
- `reference operator[]` (`std::ptrdiff_t __index`) const

## Friends

- `_Pointer_adapter operator+` (const `_Pointer_adapter &__lhs`, short `__offset`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter &__lhs`, unsigned short `__offset`)
- `_Pointer_adapter operator+` (unsigned long `__offset`, const `_Pointer_adapter &__rhs`)
- `_Pointer_adapter operator+` (unsigned short `__offset`, const `_Pointer_adapter &__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter &__lhs`, long `__offset`)
- `_Pointer_adapter operator+` (long `__offset`, const `_Pointer_adapter &__rhs`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter &__lhs`, int `__offset`)
- `_Pointer_adapter operator+` (const `_Pointer_adapter &__lhs`, unsigned long `__offset`)
- `_Pointer_adapter operator+` (unsigned int `__offset`, const `_Pointer_adapter &__rhs`)
- `_Pointer_adapter operator+` (short `__offset`, const `_Pointer_adapter &__rhs`)
- `_Pointer_adapter operator+` (int `__offset`, const `_Pointer_adapter &__rhs`)

- [\\_Pointer\\_adapter operator+](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned int \_\_offset)
- `std::ptrdiff_t operator-` (element\_type \*\_\_lhs, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- `std::ptrdiff_t operator-` (const [\\_Pointer\\_adapter](#) &\_\_lhs, element\_type \*\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned short \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, int \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, short \_\_offset)
- `template<typename _Up > std::ptrdiff_t operator-` (\_Up \*\_\_lhs, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, long \_\_offset)
- `template<typename _Up > std::ptrdiff_t operator-` (const [\\_Pointer\\_adapter](#) &\_\_lhs, \_Up \*\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned int \_\_offset)

### 5.27.1 Detailed Description

`template<typename _Storage_policy> class __gnu_cxx::_Pointer_adapter< _Storage_policy >`

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from [ext/cast.h](#). This allows pointer cast operations to be overloaded is necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == \_Pointer\_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == \_Pointer\_adapter<_Std_pointer_impl<const _Tp> >;
_Tp* const == const \_Pointer\_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const \_Pointer\_adapter<_Std_pointer_impl<const _Tp> >;
```

## 5.28 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference 799

Definition at line 281 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.28 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

### Public Types

- typedef `_Tp` `element_type`

### Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (_Tp *__arg)`

### 5.28.1 Detailed Description

`template<typename _Tp> class \_\_gnu\_cxx::\_Relative\_pointer\_impl<_Tp>`

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address. This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 101 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.29 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

### Public Types

- typedef `const _Tp` `element_type`

### Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (const _Tp *__arg)`

#### 5.29.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Relative_pointer_impl< const _Tp >`

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 153 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.30 `__gnu_cxx::_Std_pointer_impl< _Tp >` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

### Public Types

- typedef `_Tp` `element_type`

### Public Member Functions

- `_Tp * get () const`

- bool `operator<` (const `_Std_pointer_impl` &\_\_rarg) const
- bool `operator==` (const `_Std_pointer_impl` &\_\_rarg) const
- void `set` (element\_type \* \_\_arg)

### 5.30.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::Std_pointer_impl< _Tp >`

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer. A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==(())` to support pointer comparison.

Definition at line 58 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.31 `__gnu_cxx::Unqualified_type< _Tp >` Struct Template Reference

### Public Types

- typedef `_Tp` `type`

### 5.31.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::Unqualified_type< _Tp >`

This structure accomodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

Definition at line 233 of file `pointer.h`.

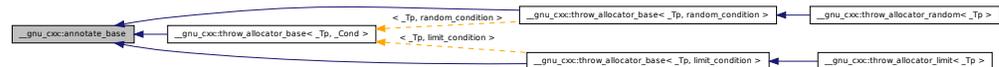
The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.32 `__gnu_cxx::annotate_base` Struct Reference

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Inheritance diagram for `__gnu_cxx::annotate_base`:



### Public Member Functions

- void **check\_allocated** (size\_t label)
- void **check\_allocated** (void \*p, size\_t size)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)

### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### Friends

- [std::ostream](#) & **operator**<< ([std::ostream](#) &, const [annotate\\_base](#) &)

#### 5.32.1 Detailed Description

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Definition at line 94 of file `throw_allocator.h`.

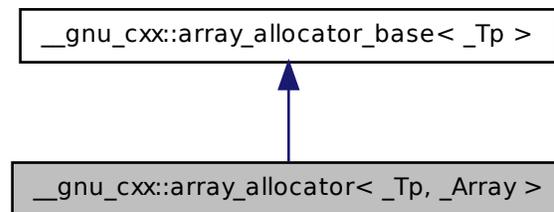
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.33 `__gnu_cxx::array_allocator<_Tp, _Array >` Class Template Reference

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Inheritance diagram for `__gnu_cxx::array_allocator<_Tp, _Array >`:



#### Public Types

- typedef `_Array` **array\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- **array\_allocator** (`array_type *__array=NULL`) throw ()
- **array\_allocator** (`const array_allocator &__o`) throw ()
- `template<typename _Tp1, typename _Array1 >`  
**array\_allocator** (`const array_allocator<_Tp1, _Array1 > &`) throw ()
- **pointer address** (`reference __x`) const
- **const\_pointer address** (`const_reference __x`) const
- **pointer allocate** (`size_type __n, const void * =0`)

- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer, size\_type)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.33.1 Detailed Description

**template<typename \_Tp, typename \_Array = std::tr1::array<\_Tp, 1>> class \_gnu\_cxx::array\_allocator<\_Tp, \_Array >**

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Definition at line 96 of file array\_allocator.h.

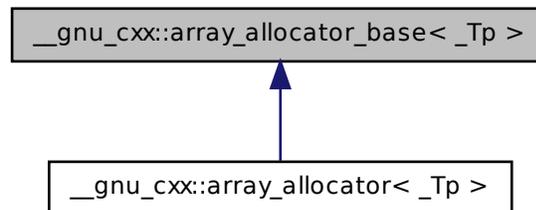
The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

## 5.34 \_\_gnu\_cxx::array\_allocator\_base<\_Tp > Class Template Reference

Base class.

Inheritance diagram for \_\_gnu\_cxx::array\_allocator\_base<\_Tp >:



## Public Types

- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- pointer **address** (reference `__x`) const
- `const_pointer` **address** (const\_reference `__x`) const
- void **construct** (pointer `__p`, const `_Tp &__val`)
- `template<typename... _Args>`  
void **construct** (pointer `__p`, `_Args &&...__args`)
- void **deallocate** (pointer, `size_type`)
- void **destroy** (pointer `__p`)
- `size_type` **max\_size** () const throw ()

### 5.34.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::array_allocator_base< _Tp >`

Base class.

Definition at line 46 of file `array_allocator.h`.

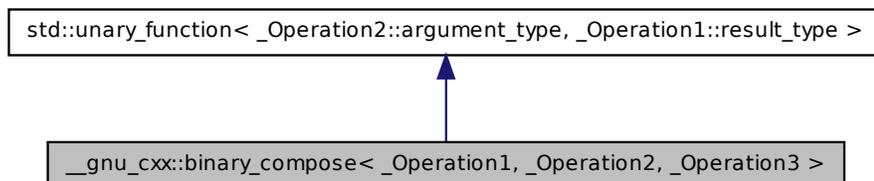
The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

## 5.35 `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >` Class Template Reference

An [SGI extension](#) .

Inheritance diagram for `__gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3 >`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- **binary\_compose** (const `_Operation1` &`_x`, const `_Operation2` &`_y`, const `_Operation3` &`_z`)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &`_x`) const

## Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**
- `_Operation3` **\_M\_fn3**

### 5.35.1 Detailed Description

`template<class _Operation1, class _Operation2, class _Operation3> class __gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3 >`

An [SGI extension](#) .

Definition at line 150 of file `ext/functional`.

### 5.35.2 Member Typedef Documentation

**5.35.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function<_Arg, _Result>::argument_type  
[inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.35.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function<_Arg, _Result>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

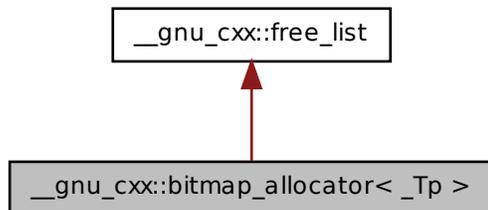
The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.36 `__gnu_cxx::bitmap_allocator<_Tp>` Class Template Reference

Bitmap Allocator, primary template.

Inheritance diagram for `__gnu_cxx::bitmap_allocator<_Tp>`:



## Public Types

- typedef free\_list::\_\_mutex\_type **\_\_mutex\_type**
- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **bitmap\_allocator** (const [bitmap\\_allocator](#) &)
- template<typename \_Tp1 >  
**bitmap\_allocator** (const [bitmap\\_allocator](#)< \_Tp1 > &) throw ()
- pointer [\\_M\\_allocate\\_single\\_object](#) () throw (std::bad\_alloc)
- void [\\_M\\_deallocate\\_single\\_object](#) (pointer \_\_p) throw ()
- const\_pointer **address** (const\_reference \_\_r) const
- pointer **address** (reference \_\_r) const
- pointer **allocate** (size\_type \_\_n, typename [bitmap\\_allocator](#)< void >::const\_pointer)
- pointer **allocate** (size\_type \_\_n)
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&... \_\_args)
- void **construct** (pointer \_\_p, const\_reference \_\_data)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) throw ()
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

## Private Types

- typedef vector\_type::iterator **iterator**
- typedef \_\_detail::\_\_mini\_vector< value\_type > **vector\_type**

## Private Member Functions

- void [\\_M\\_clear](#) ()
- size\_t \* [\\_M\\_get](#) (size\_t \_\_sz) throw (std::bad\_alloc)
- void [\\_M\\_insert](#) (size\_t \*\_\_addr) throw ()

### 5.36.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::bitmap_allocator<_Tp>`

Bitmap Allocator, primary template.

Definition at line 687 of file `bitmap_allocator.h`.

### 5.36.2 Member Function Documentation

**5.36.2.1** `template<typename _Tp> pointer __gnu_cxx::bitmap_allocator<_Tp>::M_allocate_single_object ( ) throw (std::bad_alloc) [inline]`

Allocates memory for a single object of size `sizeof(_Tp)`.

#### Exceptions

*std::bad\_alloc*. If memory can not be allocated.

Complexity: Worst case complexity is  $O(N)$ , but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of  $O(1)$ ! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 821 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan_forward()`.

**5.36.2.2** `template<typename _Tp> void __gnu_cxx::bitmap_allocator<_Tp>::M_deallocate_single_object ( pointer __p ) throw () [inline]`

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity:  $O(\lg(N))$ , but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in  $O(1)$  time by the `deallocate` function.

Definition at line 911 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `std::__rotate()`.

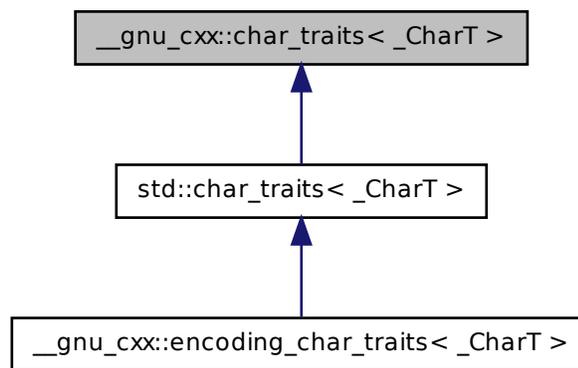
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.37 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Base class used to implement `std::char_traits`.

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



### Public Types

- typedef `_CharT char_type`
- typedef `_Char_types<_CharT>::int_type int_type`
- typedef `_Char_types<_CharT>::off_type off_type`
- typedef `_Char_types<_CharT>::pos_type pos_type`
- typedef `_Char_types<_CharT>::state_type state_type`

### Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type * assign` (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type * copy` (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)

- static `int_type eof()`
- static `bool eq(const char_type &__c1, const char_type &__c2)`
- static `bool eq_int_type(const int_type &__c1, const int_type &__c2)`
- static `const char_type * find(const char_type * __s, std::size_t __n, const char_type &__a)`
- static `std::size_t length(const char_type * __s)`
- static `bool lt(const char_type &__c1, const char_type &__c2)`
- static `char_type * move(char_type * __s1, const char_type * __s2, std::size_t __n)`
- static `int_type not_eof(const int_type &__c)`
- static `char_type to_char_type(const int_type &__c)`
- static `int_type to_int_type(const char_type &__c)`

### 5.37.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::char_traits< _CharT >`

Base class used to implement `std::char_traits`.

#### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod\\_char\\_traits.h](#).

Definition at line 90 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.38 `__gnu_cxx::character< V, I, S >` Struct Template Reference

A POD class that serves as a character abstraction class.

## Public Types

- typedef [character](#)< V, I, S > **char\_type**
- typedef I **int\_type**
- typedef S **state\_type**
- typedef V **value\_type**

## Static Public Member Functions

- template<typename V2 >  
static [char\\_type](#) **from** (const V2 &v)
- template<typename V2 >  
static V2 **to** (const [char\\_type](#) &c)

## Public Attributes

- value\_type **value**

### 5.38.1 Detailed Description

**template<typename V, typename I, typename S = std::mbstate\_t> struct [\\_\\_gnu\\_cxx::character](#)< V, I, S >**

A POD class that serves as a character abstraction class.

Definition at line 45 of file [pod\\_char\\_traits.h](#).

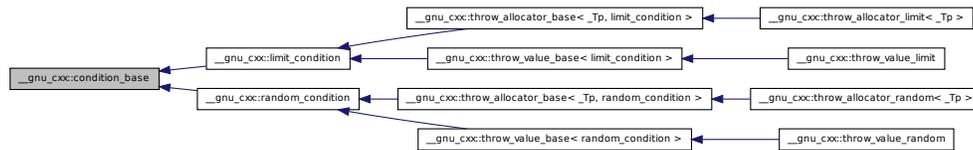
The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

### 5.39 [\\_\\_gnu\\_cxx::condition\\_base](#) Struct Reference

Base struct for condition policy.

Inheritance diagram for `__gnu_cxx::condition_base`:



### 5.39.1 Detailed Description

Base struct for condition policy. Requires a public member function with the signature `void throw_conditionally()`

Definition at line 253 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.40 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::_Constant_binary_fun<_Result, _Arg1, _Arg2 >`.

### Public Types

- typedef `_Arg1` **first\_argument\_type**
- typedef `_Result` **result\_type**
- typedef `_Arg2` **second\_argument\_type**

### Public Member Functions

- **constant\_binary\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** (const `_Arg1` &, const `_Arg2` &) const

## Public Attributes

- `_Result _M_val`

### 5.40.1 Detailed Description

`template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1> struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 316 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.41 `__gnu_cxx::constant_unary_fun< _Result, _Argument >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Constant_unary_fun< _Result, _Argument >`.

## Public Types

- `typedef _Argument argument_type`
- `typedef _Result result_type`

## Public Member Functions

- `constant_unary_fun` (`const _Result &__v`)
- `const result_type & operator()` (`const _Argument &`) `const`

## Public Attributes

- `result_type _M_val`

### 5.41.1 Detailed Description

```
template<class _Result, class _Argument = _Result> struct __gnu_cxx::constant_unary_fun<_Result, _Argument >
```

An [SGI extension](#).

Definition at line 308 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.42 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::_Constant_void_fun<_Result>`.

### Public Types

- typedef `_Result` `result_type`

### Public Member Functions

- `constant_void_fun` (`const _Result &__v`)
- `const result_type & operator() () const`

### Public Attributes

- `result_type _M_val`

### 5.42.1 Detailed Description

```
template<class _Result> struct __gnu_cxx::constant_void_fun<_Result >
```

An [SGI extension](#).

Definition at line 299 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.43 `__gnu_cxx::debug_allocator< _Alloc >` Class Template Reference

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

### Public Types

- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Alloc::value_type` **value\_type**

### Public Member Functions

- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const void \* \_\_hint)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)

#### 5.43.1 Detailed Description

```
template<typename _Alloc> class __gnu_cxx::debug_allocator< _Alloc >
```

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 61 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 5.44 `__gnu_cxx::enc_filebuf<_CharT>` Class Template Reference

class `enc_filebuf`.

Inheritance diagram for `__gnu_cxx::enc_filebuf<_CharT>`:



### Public Types

- typedef `codecvt<char_type, char, __state_type>` `__codecvt_type`
- typedef `__basic_file<char>` `__file_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::off_type` `off_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `traits_type::state_type` `state_type`
- typedef `encoding_char_traits<_CharT>` `traits_type`

### Public Member Functions

- `enc_filebuf` (`state_type &__state`)
- `__filebuf_type * close` ()
- `streamsize in_avail` ()
- `bool is_open` () const throw ()
- `__filebuf_type * open` (const `std::string &__s`, `ios_base::openmode __mode`)
- `__filebuf_type * open` (const `char * __s`, `ios_base::openmode __mode`)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- `streamsize sgetn` (`char_type * __s`, `streamsize __n`)
- `int_type snextc` ()
- `int_type sputbackc` (`char_type __c`)
- `int_type sputc` (`char_type __c`)
- `streamsize sputn` (const `char_type * __s`, `streamsize __n`)
- void `stoss` ()
- `int_type sungetc` ()

## Protected Member Functions

- void `_M_allocate_internal_buffer` ()
  - bool `_M_convert_to_external` (char\_type \*, streamsize)
  - void `_M_create_pback` ()
  - void `_M_destroy_internal_buffer` () throw ()
  - void `_M_destroy_pback` () throw ()
  - pos\_type `_M_seek` (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
  - void `_M_set_buffer` (streamsize \_\_off)
  - bool `_M_terminate_output` ()
  - void `gbump` (int \_\_n)
  - virtual void `imbue` (const locale &\_\_loc)
  - virtual int\_type `overflow` (int\_type \_\_c=encoding\_char\_traits< \_CharT >::eof())
  - virtual int\_type `overflow` (int\_type=traits\_type::eof())
  - virtual int\_type `pbackfail` (int\_type=traits\_type::eof())
  - virtual int\_type `pbackfail` (int\_type \_\_c=encoding\_char\_traits< \_CharT >::eof())
  - void `pbump` (int \_\_n)
  - virtual pos\_type `seekoff` (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type `seekoff` (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual pos\_type `seekpos` (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type `seekpos` (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual \_\_streambuf\_type \* `setbuf` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual basic\_streambuf< char\_type, encoding\_char\_traits< \_CharT > > \* `setbuf` (char\_type \*, streamsize)
  - void `setg` (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
  - void `setp` (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
  - virtual streamsize `showmanyc` ()
  - virtual int `sync` ()
  - virtual int\_type `uflow` ()
  - virtual int\_type `underflow` ()
  - virtual streamsize `xsgetn` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual streamsize `xsgetn` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual streamsize `xspun` (const char\_type \*\_\_s, streamsize \_\_n)
  - virtual streamsize `xspun` (const char\_type \*\_\_s, streamsize \_\_n)
- 
- char\_type \* `eback` () const

- `char_type * gptr () const`
- `char_type * egptr () const`
  
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * epptr () const`

### Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `bool _M_reading`
- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
  
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

### Friends

- `__gnu_cxx::__enable_if<__is_char<_CharT2>::__value, _CharT2 * >::__-  
type __copy_move_a2 (istreambuf_iterator<_CharT2>, istreambuf_iterator<  
_CharT2>, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,  
bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`

- `__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type` **find** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `const _CharT2 &`)
- `basic_istream< _CharT2, _Traits2 > & getline` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`, `_CharT2`)
- class **ios\_base**
- class **istreambuf\_iterator< char\_type, traits\_type >**
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `_CharT2 *`)
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`)
- class **ostreambuf\_iterator< char\_type, traits\_type >**
  
- locale [pubimbue](#) (`const locale &__loc`)
- locale [getloc](#) () `const`
- `__streambuf_type * pubsetbuf` (`char_type *__s`, `streamsize __n`)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- locale `_M_buf_locale`

### 5.44.1 Detailed Description

`template<typename _CharT> class __gnu_cxx::enc_filebuf< _CharT >`

class [enc\\_filebuf](#).

Definition at line 40 of file `enc_filebuf.h`.

### 5.44.2 Member Typedef Documentation

**5.44.2.1** `typedef basic_streambuf<char_type, traits_type> std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__streambuf_type`  
[**inherited**]

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 77 of file `fstream`.

#### 5.44.2.2 `typedef _CharT std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::char_type` [inherited]

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 71 of file `fstream`.

#### 5.44.2.3 `typedef traits_type::int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type` [inherited]

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 73 of file `fstream`.

#### 5.44.2.4 `typedef traits_type::off_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::off_type` [inherited]

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 75 of file `fstream`.

#### 5.44.2.5 `template<typename _CharT> typedef traits_type::pos_type __gnu_cxx::enc_filebuf<_CharT>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 46 of file `enc_filebuf.h`.

**5.44.2.6** `template<typename _CharT > typedef encoding_char_traits<_CharT> __gnu_cxx::enc_filebuf<_CharT>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 44 of file `enc_filebuf.h`.

### 5.44.3 Member Function Documentation

**5.44.3.1** `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_create_pback( ) [inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

**5.44.3.2** `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_destroy_pback( ) throw () [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

**5.44.3.3** `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer( streamsize __off ) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 387 of file `fstream`.

#### 5.44.3.4 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close( ) [inherited]`

Closes the currently associated file.

#### Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

#### 5.44.3.5 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

#### 5.44.3.6 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file `streambuf`.

**5.44.3.7** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT > >::eptr( ) const [inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `pptr()` returns the end pointer for the output sequence

Definition at line 513 of file `streambuf`.

**5.44.3.8** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT > >::gbump( int __n ) [inline, protected, inherited]`

Moving the read position.

#### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

**5.44.3.9** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT > >::getloc( ) const [inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

**5.44.3.10** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr ( ) const` [`inline`, `protected`, `inherited`]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

**5.44.3.11** `virtual void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue ( const locale & )` [`protected`, `virtual`, `inherited`]

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

#### 5.44.3.12 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::in_avail()` `[inline, inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file `streambuf`.

#### 5.44.3.13 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open()` `const throw()` `[inline, inherited]`

Returns true if the external file is open.

Definition at line 222 of file `fstream`.

#### 5.44.3.14 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open(const char* __s, ios_base::openmode __mode)` `[inherited]`

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

#### Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

ios_base Flag combination	stdio equivalent	binary	in	out	trunc	app
<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>
<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>
<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>
<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>
<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>
<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>
<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>
<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>
<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>
<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>
<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>	<code>+</code>
<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>	<code>r</code>
<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>	<code>w</code>
<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>	<code>a</code>
<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>	<code>b</code>

**5.44.3.15** `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open ( const std::string & __s, ios_base::openmode __mode ) [inline, inherited]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

#### Returns

*this* on success, NULL on failure

Definition at line 275 of file `fstream`.

**5.44.3.16** `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::overflow ( int_type = traits_type::eof() ) [inline, protected, virtual, inherited]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

- c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Definition at line 746 of file `streambuf`.

**5.44.3.17** `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbackfail( int_type = traits_type::eof() ) [inline, protected, virtual, inherited]`

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

eof() on failure, *some other value* on success

#### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

#### Note

Base class version does nothing, returns eof().

Definition at line 702 of file `streambuf`.

**5.44.3.18** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbase( ) const [inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

**5.44.3.19** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbump( int __n ) [inline, protected, inherited]`

Moving the write position.

**Parameters**

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file `streambuf`.

**5.44.3.20** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr ( ) const` [`inline`, `protected`, `inherited`]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pptr()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

**5.44.3.21** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubimbue ( const locale & __loc )` [`inline`, `inherited`]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

**5.44.3.22** `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff ( off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out )` [`inline`, `inherited`]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 239 of file streambuf.

**5.44.3.23** `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT >>::pubseekpos ( pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for imbue().

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 244 of file streambuf.

**5.44.3.24** `__streambuf_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT >>::pubsetbuf ( char_type * __s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file streambuf.

**5.44.3.25** `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT >>::pubsync ( ) [inline, inherited]`

Entry point for imbue().

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

**5.44.3.26** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sbumpc ( )` [inline, inherited]

Getting the next character.

**Returns**

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

**5.44.3.27** `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::seekoff ( off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out )` [inline, protected, virtual, inherited]

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

**5.44.3.28** `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::seekpos ( pos_type, ios_base::openmode = ios_base::in | ios_base::out )` [inline, protected, virtual, inherited]

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

**5.44.3.29** `virtual __streambuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::setbuf( char_type * __s, streamsize __n )` [**protected, virtual, inherited**]

Manipulates the buffer.

#### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

#### Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

**5.44.3.30** `virtual basic_streambuf<char_type,encoding_char_traits<_CharT>>* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setbuf( char_type *, streamsize )` [**inline, protected, virtual, inherited**]

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

#### Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

**5.44.3.31** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg( char_type * __gbeg, char_type * __gnext, char_type * __gend )` [`inline`, `protected`, `inherited`]

Setting the three read area pointers.

#### Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

#### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

**5.44.3.32** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setp( char_type * __pbeg, char_type * __pend )` [`inline`, `protected`, `inherited`]

Setting the three write area pointers.

#### Parameters

*pbeg* A pointer.

*pend* A pointer.

#### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `pptr()`

Definition at line 533 of file `streambuf`.

**5.44.3.33** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetc( )` [`inline`, `inherited`]

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

**5.44.3.34** `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn( char_type * __s, streamsize __n )`  
**[inline, inherited]**

Entry point for `xsggetn`.

#### Parameters

*s* A buffer area.

*n* A count.

Returns `xsggetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

**5.44.3.35** `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::showmanyc( )` **[protected, virtual, inherited]**

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
 [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.44.3.36** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::snextc( )` **[inline, inherited]**

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

**5.44.3.37** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputback( char_type __c ) [inline, inherited]`

Pushing characters back into the input stream.

**Parameters**

*c* The character to push back.

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

**5.44.3.38** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::putc( char_type __c ) [inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*c* A character to output.

**Returns**

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

**5.44.3.39** `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn( const char_type * __s, streamsize __n )`  
[inline, inherited]

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.44.3.40** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::stoss( )` [inline, inherited]

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.44.3.41** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc( )` [inline, inherited]

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

**5.44.3.42** `virtual int std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::sync( void ) [protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.44.3.43** `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::uflow( ) [inline, protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 678 of file `streambuf`.

**5.44.3.44** `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::underflow( ) [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.44.3.45 virtual streamsize** `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsgetn ( char_type * __s, streamsize __n )` [**protected, virtual, inherited**]

Multiple character extraction.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

#### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

**5.44.3.46 virtual streamsize** `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsputn ( const char_type * __s, streamsize __n )` [**protected, virtual, inherited**]

Multiple character insertion.

**Parameters**

- s* A buffer area.
- n* Maximum number of characters to write.

**Returns**

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

**5.44.4 Member Data Documentation****5.44.4.1** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf` [protected, inherited]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

**5.44.4.2** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

**5.44.4.3** `size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size` [protected, inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file `fstream`.

**5.44.4.4** `char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf` [protected, inherited]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 151 of file `fstream`.

#### 5.44.4.5 `streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf_size` [protected, inherited]

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

#### 5.44.4.6 `const char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_next` [protected, inherited]

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 163 of file `fstream`.

#### 5.44.4.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_beg` [protected, inherited]

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

#### 5.44.4.8 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` [protected, inherited]

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.44.4.9** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_end` [protected, inherited]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.44.4.10** `ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode` [protected, inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file `fstream`.

**5.44.4.11** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg` [protected, inherited]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

**5.44.4.12** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur` [protected, inherited]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

**5.44.4.13** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT > >::_M_out_end` [protected, inherited]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

**5.44.4.14** `char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT > >::_M_pback` [protected, inherited]

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 137 of file `fstream`.

**5.44.4.15** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT > >::_M_pback_cur_save` [protected, inherited]

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 138 of file `fstream`.

---

## 5.45 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference 843

### 5.44.4.16 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_end_save` [protected, inherited]

Necessary bits for putback buffer management.

#### Note

putbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

### 5.44.4.17 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_init` [protected, inherited]

Necessary bits for putback buffer management.

#### Note

putbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

### 5.44.4.18 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_reading` [protected, inherited]

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

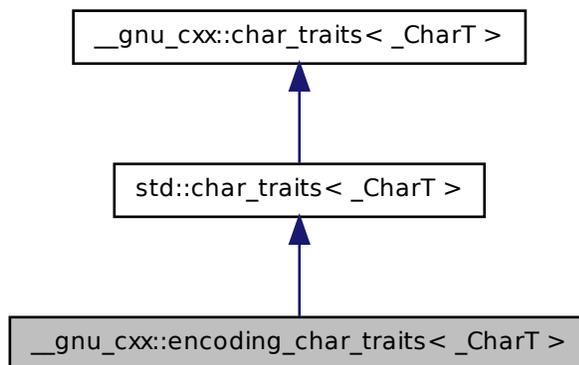
The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## 5.45 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

[encoding\\_char\\_traits](#)

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `std::fpos<state_type>` **pos\_type**
- typedef `encoding_state` **state\_type**

## Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static int\_type **eof** ()
- static bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static bool **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static `std::size_t length` (`const char_type *__s`)
- static `bool It` (`const char_type &__c1, const char_type &__c2`)
- static `char_type * move` (`char_type *__s1, const char_type *__s2, std::size_t __n`)
- static `int_type not_eof` (`const int_type &__c`)
- static `char_type to_char_type` (`const int_type &__c`)
- static `int_type to_int_type` (`const char_type &__c`)

### 5.45.1 Detailed Description

```
template<typename _CharT> struct __gnu_cxx::encoding_char_traits< _-
CharT >
```

[encoding\\_char\\_traits](#)

Definition at line 209 of file `codecvt_specializations.h`.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.46 `__gnu_cxx::encoding_state` Class Reference

Extension to use `iconv` for dealing with character encodings.

### Public Types

- typedef `iconv_t descriptor_type`

### Public Member Functions

- `encoding_state` (`const char *__int, const char *__ext, int __ibom=0, int __-ebom=0, int __bytes=1`)
- `encoding_state` (`const encoding\_state &__obj`)
- `int character_ratio` () const
- `int external_bom` () const
- `const std::string external_encoding` () const
- `bool good` () const throw ()
- `const descriptor_type & in_descriptor` () const
- `int internal_bom` () const
- `const std::string internal_encoding` () const
- `encoding\_state & operator=` (`const encoding\_state &__obj`)
- `const descriptor_type & out_descriptor` () const

## Protected Member Functions

- void **construct** (const [encoding\\_state](#) &\_\_obj)
- void **destroy** () throw ()
- void **init** ()

## Protected Attributes

- int **\_M\_bytes**
- int **\_M\_ext\_bom**
- [std::string](#) **\_M\_ext\_enc**
- descriptor\_type **\_M\_in\_desc**
- int **\_M\_int\_bom**
- [std::string](#) **\_M\_int\_enc**
- descriptor\_type **\_M\_out\_desc**

### 5.46.1 Detailed Description

Extension to use iconv for dealing with character encodings.

Definition at line 49 of file `codecv_t_specializations.h`.

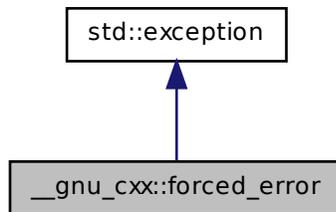
The documentation for this class was generated from the following file:

- [codecv\\_t\\_specializations.h](#)

### 5.47 `__gnu_cxx::forced_error` Struct Reference

Thrown by exception safety machinery.

Inheritance diagram for `__gnu_cxx::forced_error`:



## Public Member Functions

- virtual const char \* [what](#) () const throw ()

### 5.47.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 73 of file `throw_allocator.h`.

### 5.47.2 Member Function Documentation

#### 5.47.2.1 virtual const char\* `std::exception::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::tr1::bad_weak_ptr`, and `std::ios_base::failure`.

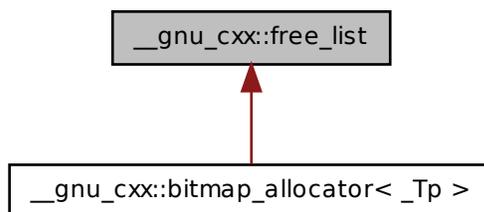
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.48 `__gnu_cxx::free_list` Class Reference

The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).

Inheritance diagram for `__gnu_cxx::free_list`:



### Public Types

- typedef `__mutex` **`__mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

### Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (`size_t __sz`) throw (`std::bad_alloc`)
- void `_M_insert` (`size_t *__addr`) throw ()

#### 5.48.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).

Definition at line 521 of file `bitmap_allocator.h`.

## 5.48.2 Member Function Documentation

### 5.48.2.1 `void __gnu_cxx::free_list::_M_clear ( )`

This function just clears the internal Free List, and gives back all the memory to the OS.

### 5.48.2.2 `size_t* __gnu_cxx::free_list::_M_get ( size_t __sz ) throw (std::bad_alloc)`

This function gets a block of memory of the specified size from the free list.

#### Parameters

`__sz` The size in bytes of the memory required.

#### Returns

A pointer to the new memory block of size at least equal to that requested.

### 5.48.2.3 `void __gnu_cxx::free_list::_M_insert ( size_t * __addr ) throw () [inline]`

This function returns the block of memory to the internal free list.

#### Parameters

`__addr` The pointer to the memory block that was given by a call to the `_M_get` function.

Definition at line 631 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.49 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc > Class` Template Reference

### Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**

- typedef [\\_Ht::const\\_pointer](#) **const\_pointer**
- typedef [\\_Ht::const\\_reference](#) **const\_reference**
- typedef [\\_Tp](#) **data\_type**
- typedef [\\_Ht::difference\\_type](#) **difference\_type**
- typedef [\\_Ht::hasher](#) **hasher**
- typedef [\\_Ht::iterator](#) **iterator**
- typedef [\\_Ht::key\\_equal](#) **key\_equal**
- typedef [\\_Ht::key\\_type](#) **key\_type**
- typedef [\\_Tp](#) **mapped\_type**
- typedef [\\_Ht::pointer](#) **pointer**
- typedef [\\_Ht::reference](#) **reference**
- typedef [\\_Ht::size\\_type](#) **size\_type**
- typedef [\\_Ht::value\\_type](#) **value\_type**

## Public Member Functions

- **hash\_map** (size\_type \_\_n)
- template<class [\\_InputIterator](#) >  
**hash\_map** ([\\_InputIterator](#) \_\_f, [\\_InputIterator](#) \_\_l)
- template<class [\\_InputIterator](#) >  
**hash\_map** ([\\_InputIterator](#) \_\_f, [\\_InputIterator](#) \_\_l, size\_type \_\_n)
- template<class [\\_InputIterator](#) >  
**hash\_map** ([\\_InputIterator](#) \_\_f, [\\_InputIterator](#) \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class [\\_InputIterator](#) >  
**hash\_map** ([\\_InputIterator](#) \_\_f, [\\_InputIterator](#) \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- [pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- [pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)

- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_func** () const
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- [pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- [pair](#)< iterator, bool > **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- \_Tp & **operator[]** (const key\_type &\_\_key)
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** ([hash\\_map](#) &\_\_hs)

## Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_Al >  
bool **operator==** (const [hash\\_map](#)< \_K1, \_T1, \_HF, \_EqK, \_Al > &, const  
[hash\\_map](#)< \_K1, \_T1, \_HF, \_EqK, \_Al > &)

### 5.49.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc >`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 75 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.50 `__gnu_cxx::hash_multimap`< `_Key`, `_Tp`, `_HashFn`, `_EqualKey`, `_Alloc` > Class Template Reference

### Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Ht::const_pointer` **const\_pointer**
- typedef `_Ht::const_reference` **const\_reference**
- typedef `_Tp` **data\_type**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

### Public Member Functions

- **hash\_multimap** (size\_type \_\_n)
- template<class `_InputIterator` >  
**hash\_multimap** (`_InputIterator` \_\_f, `_InputIterator` \_\_l)
- template<class `_InputIterator` >  
**hash\_multimap** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n)
- template<class `_InputIterator` >  
**hash\_multimap** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class `_InputIterator` >  
**hash\_multimap** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- **hash\_multimap** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const

- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- [pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- [pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_func** () const
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** ([hash\\_multimap](#) &\_\_hs)

## Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_Al >  
bool **operator==** (const [hash\\_multimap](#)< \_K1, \_T1, \_HF, \_EqK, \_Al > &, const [hash\\_multimap](#)< \_K1, \_T1, \_HF, \_EqK, \_Al > &)

### 5.50.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc >`

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

Definition at line 288 of file hash\_map.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.51 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

**Public Types**

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::const_iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

**Public Member Functions**

- **hash\_multiset** (size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())

- `hash_multiset` (size\_type \_\_n, const hasher &\_\_hf)
- `hash_multiset` (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator `begin` () const
- size\_type `bucket_count` () const
- void `clear` ()
- size\_type `count` (const key\_type &\_\_key) const
- size\_type `elems_in_bucket` (size\_type \_\_n) const
- bool `empty` () const
- iterator `end` () const
- `pair`< iterator, iterator > `equal_range` (const key\_type &\_\_key) const
- void `erase` (iterator \_\_f, iterator \_\_l)
- void `erase` (iterator \_\_it)
- size\_type `erase` (const key\_type &\_\_key)
- iterator `find` (const key\_type &\_\_key) const
- allocator\_type `get_allocator` () const
- hasher `hash_funct` () const
- template<class \_InputIterator >  
void `insert` (\_InputIterator \_\_f, \_InputIterator \_\_l)
- iterator `insert` (const value\_type &\_\_obj)
- iterator `insert_noresize` (const value\_type &\_\_obj)
- key\_equal `key_eq` () const
- size\_type `max_bucket_count` () const
- size\_type `max_size` () const
- void `resize` (size\_type \_\_hint)
- size\_type `size` () const
- void `swap` (`hash_multiset` &hs)

## Friends

- template<class \_Val, class \_HF, class \_EqK, class \_Al >  
bool `operator==` (const `hash_multiset`< \_Val, \_HF, \_EqK, \_Al > &, const `hash_multiset`< \_Val, \_HF, \_EqK, \_Al > &)

### 5.51.1 Detailed Description

template<class \_Value, class \_HashFcn = hash<\_Value>, class \_EqualKey = equal\_to<\_Value>, class \_Alloc = allocator<\_Value>> class `__gnu_cxx::hash_multiset`< \_Value, \_HashFcn, \_EqualKey, \_Alloc >

This is an SGI extension.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

Definition at line 281 of file hash\_set.

The documentation for this class was generated from the following file:

- [hash\\_set](#)

## 5.52 `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

**Public Types**

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::const_iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

**Public Member Functions**

- **hash\_set** (size\_type \_\_n)
- template<class `_InputIterator` >  
**hash\_set** (`_InputIterator` \_\_f, `_InputIterator` \_\_l)
- template<class `_InputIterator` >  
**hash\_set** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n)
- template<class `_InputIterator` >  
**hash\_set** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n, const hasher & \_\_hf)
- template<class `_InputIterator` >  
**hash\_set** (`_InputIterator` \_\_f, `_InputIterator` \_\_l, size\_type \_\_n, const hasher & \_\_hf, const key\_equal & \_\_eq, const allocator\_type & \_\_a=allocator\_type())

- `hash_set` (size\_type \_\_n, const hasher &\_\_hf)
- `hash_set` (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- iterator `begin` () const
- size\_type `bucket_count` () const
- void `clear` ()
- size\_type `count` (const key\_type &\_\_key) const
- size\_type `elems_in_bucket` (size\_type \_\_n) const
- bool `empty` () const
- iterator `end` () const
- `pair`< iterator, iterator > `equal_range` (const key\_type &\_\_key) const
- void `erase` (iterator \_\_f, iterator \_\_l)
- void `erase` (iterator \_\_it)
- size\_type `erase` (const key\_type &\_\_key)
- iterator `find` (const key\_type &\_\_key) const
- allocator\_type `get_allocator` () const
- hasher `hash_funct` () const
- template<class \_InputIterator >  
void `insert` (\_InputIterator \_\_f, \_InputIterator \_\_l)
- `pair`< iterator, bool > `insert` (const value\_type &\_\_obj)
- `pair`< iterator, bool > `insert_noresize` (const value\_type &\_\_obj)
- key\_equal `key_eq` () const
- size\_type `max_bucket_count` () const
- size\_type `max_size` () const
- void `resize` (size\_type \_\_hint)
- size\_type `size` () const
- void `swap` (`hash_set` &\_\_hs)

## Friends

- template<class \_Val, class \_HF, class \_EqK, class \_Al >  
bool `operator==` (const `hash_set`<\_Val, \_HF, \_EqK, \_Al > &, const `hash_set`<\_Val, \_HF, \_EqK, \_Al > &)

### 5.52.1 Detailed Description

template<class \_Value, class \_HashFcn = hash<\_Value>, class \_EqualKey = equal\_to<\_Value>, class \_Alloc = allocator<\_Value>> class `__gnu_cxx::hash_set`<\_Value, \_HashFcn, \_EqualKey, \_Alloc >

This is an SGI extension.

## Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 80 of file hash\_set.

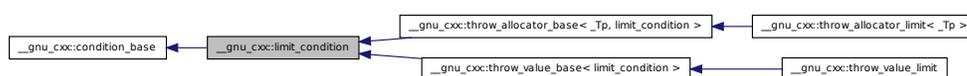
The documentation for this class was generated from the following file:

- [hash\\_set](#)

## 5.53 `__gnu_cxx::limit_condition` Struct Reference

Base class for incremental control and throw.

Inheritance diagram for `__gnu_cxx::limit_condition`:



## Classes

- struct [always\\_adjustor](#)  
*Always enter the condition.*
- struct [limit\\_adjustor](#)  
*Enter the nth condition.*
- struct [never\\_adjustor](#)  
*Never enter the condition.*

## Static Public Member Functions

- static `size_t` & **count** ()
- static `size_t` & **limit** ()
- static void **set\_limit** (const `size_t` \_\_l)
- static void **throw\_conditionally** ()

### 5.53.1 Detailed Description

Base class for incremental control and throw.

Definition at line 262 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.54 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### 5.54.1 Detailed Description

Always enter the condition.

Definition at line 286 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.55 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Enter the nth condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### Public Member Functions

- `limit_adjustor` (const size\_t \_\_l)

### 5.55.1 Detailed Description

Enter the nth condition.

Definition at line 292 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.56 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### 5.56.1 Detailed Description

Never enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.57 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

An allocator that uses `malloc`.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `malloc`
- all deallocation calls `free`.

### Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `malloc_allocator` (const `malloc_allocator` &) throw ()
- `template<typename _Tp1 > malloc_allocator` (const `malloc_allocator`<\_Tp1 > &) throw ()
- pointer `address` (reference `__x`) const
- const\_pointer `address` (const\_reference `__x`) const
- pointer `allocate` (size\_type `__n`, const void \*=0)
- void `construct` (pointer `__p`, const `_Tp` &`__val`)
- `template<typename... _Args> void construct` (pointer `__p`, `_Args` &&...`__args`)
- void `deallocate` (pointer `__p`, size\_type)
- void `destroy` (pointer `__p`)
- size\_type `max_size` () const throw ()

### 5.57.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::malloc_allocator<_Tp >`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free.

Definition at line 52 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc\\_allocator.h](#)

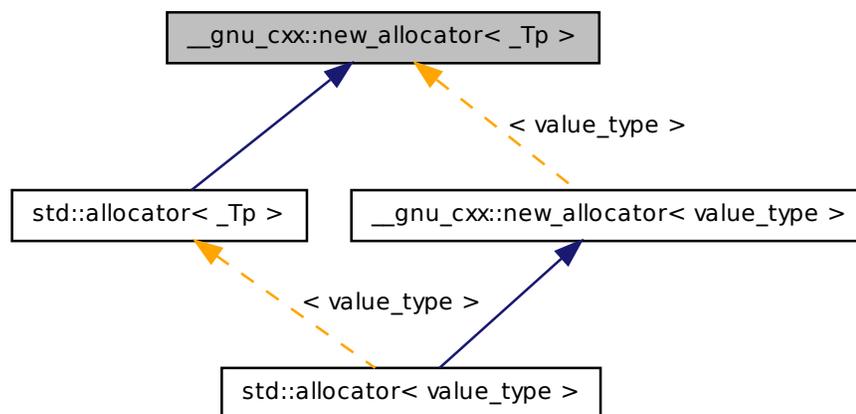
## 5.58 `__gnu_cxx::new_allocator<_Tp >` Class Template Reference

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



## Public Types

- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **new\_allocator** (const [new\\_allocator](#) &) throw ()
- `template<typename _Tp1 >`  
**new\_allocator** (const [new\\_allocator](#)<\_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const
- `const_pointer` **address** (const\_reference \_\_x) const
- pointer **allocate** (size\_type \_\_n, const void \*=0)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)

- `template<typename... _Args>`  
void **construct** (pointer `__p`, `_Args` &&...`__args`)
- void **deallocate** (pointer `__p`, `size_type`)
- void **destroy** (pointer `__p`)
- `size_type` **max\_size** () const throw ()

### 5.58.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::new_allocator< _Tp >`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 51 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new\\_allocator.h](#)

## 5.59 `__gnu_cxx::project1st< _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project1st< _Arg1, _Arg2 >`.

### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Arg1` **operator**() (const `_Arg1` &`__x`, const `_Arg2` &) const

### 5.59.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 233 of file ext/functional.

### 5.59.2 Member Typedef Documentation

**5.59.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`  
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.59.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`  
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.59.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.60 `__gnu_cxx::project2nd< _Arg1, _Arg2 > Struct` Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project2nd< _Arg1, _Arg2 >`.

## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Arg2 operator()` (`const _Arg1 &`, `const _Arg2 &__y`) `const`

### 5.60.1 Detailed Description

`template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd<_Arg1, _Arg2>`

An [SGI extension](#).

Definition at line 237 of file `ext/functional`.

### 5.60.2 Member Typedef Documentation

**5.60.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Result`  
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.60.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function<_Arg1, _Arg2, _Result`  
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.60.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

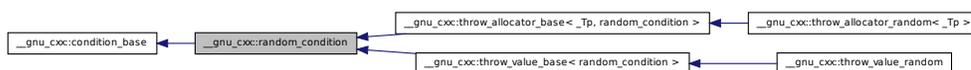
The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.61 `__gnu_cxx::random_condition` Struct Reference

Base class for random probability control and throw.

Inheritance diagram for `__gnu_cxx::random_condition`:



### Classes

- struct [always\\_adjustor](#)  
*Always enter the condition.*
- struct [group\\_adjustor](#)  
*Group condition.*
- struct [never\\_adjustor](#)  
*Never enter the condition.*

### Public Member Functions

- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### 5.61.1 Detailed Description

Base class for random probability control and throw.

Definition at line 334 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.62 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.62.1 Detailed Description

Always enter the condition.

Definition at line 367 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.63 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Group condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### Public Member Functions

- `group_adjustor` (size\_t size)

### 5.63.1 Detailed Description

Group condition.

Definition at line 352 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.64 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.64.1 Detailed Description

Never enter the condition.

Definition at line 361 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.65 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`.

### Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`

- typedef `_Key` **key\_type**
- typedef `value_type *` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Val` **value\_type**

## Public Member Functions

- **rb\_tree** (const `_Compare` &\_\_comp=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
- bool **\_\_rb\_verify** () const
- const `_Node_allocator` & **\_M\_get\_Node\_allocator** () const
- `_Node_allocator` & **\_M\_get\_Node\_allocator** ()
- iterator **\_M\_insert\_equal** (const `value_type` &\_\_x)
- template<typename `_InputIterator` >  
void **\_M\_insert\_equal** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<class `_II` >  
void **\_M\_insert\_equal** (`_II` \_\_first, `_II` \_\_last)
- iterator **\_M\_insert\_equal\_** (const\_iterator \_\_position, const `value_type` &\_\_x)
- `pair`< iterator, bool > **\_M\_insert\_unique** (const `value_type` &\_\_x)
- template<class `_II` >  
void **\_M\_insert\_unique** (`_II` \_\_first, `_II` \_\_last)
- template<typename `_InputIterator` >  
void **\_M\_insert\_unique** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- iterator **\_M\_insert\_unique\_** (const\_iterator \_\_position, const `value_type` &\_\_x)
  
- const\_iterator **begin** () const
- iterator **begin** ()
- void **clear** ()
- `size_type` **count** (const `key_type` &\_\_k) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- `pair`< const\_iterator, const\_iterator > **equal\_range** (const `key_type` &\_\_k) const
  
- `pair`< iterator, iterator > **equal\_range** (const `key_type` &\_\_k)
- void **erase** (const `key_type` \*\_\_first, const `key_type` \*\_\_last)
- const\_iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- const\_iterator **erase** (const\_iterator \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- iterator **erase** (iterator \_\_first, iterator \_\_last)

- iterator **find** (const key\_type &\_\_k)
- const\_iterator **find** (const key\_type &\_\_k) const
- allocator\_type **get\_allocator** () const
- \_Compare **key\_comp** () const
- const\_iterator **lower\_bound** (const key\_type &\_\_k) const
- iterator **lower\_bound** (const key\_type &\_\_k)
- size\_type **max\_size** () const
- [reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [const\\_reverse\\_iterator](#) **rend** () const
- [reverse\\_iterator](#) **rend** ()
- size\_type **size** () const
- void **swap** (\_Rb\_tree &\_\_t)
- iterator **upper\_bound** (const key\_type &\_\_k)
- const\_iterator **upper\_bound** (const key\_type &\_\_k) const

## Protected Types

- typedef \_Rb\_tree\_node\_base \* **\_Base\_ptr**
- typedef const \_Rb\_tree\_node\_base \* **\_Const\_Base\_ptr**

## Protected Member Functions

- \_Link\_type **\_M\_begin** ()
- \_Const\_Link\_type **\_M\_begin** () const
- \_Link\_type **\_M\_clone\_node** (\_Const\_Link\_type \_\_x)
- template<typename... \_Args>  
\_Link\_type **\_M\_create\_node** (\_Args &&...\_\_args)
- void **\_M\_destroy\_node** (\_Link\_type \_\_p)
- \_Const\_Link\_type **\_M\_end** () const
- \_Link\_type **\_M\_end** ()
- \_Link\_type **\_M\_get\_node** ()
- \_Base\_ptr & **\_M\_leftmost** ()
- \_Const\_Base\_ptr **\_M\_leftmost** () const
- void **\_M\_put\_node** (\_Link\_type \_\_p)
- \_Base\_ptr & **\_M\_rightmost** ()
- \_Const\_Base\_ptr **\_M\_rightmost** () const
- \_Base\_ptr & **\_M\_root** ()
- \_Const\_Base\_ptr **\_M\_root** () const

## Static Protected Member Functions

- static const `_Key & _S_key` (`_Const_Link_type __x`)
- static const `_Key & _S_key` (`_Const_Base_ptr __x`)
- static `_Link_type _S_left` (`_Base_ptr __x`)
- static `_Const_Link_type _S_left` (`_Const_Base_ptr __x`)
- static `_Base_ptr _S_maximum` (`_Base_ptr __x`)
- static `_Const_Base_ptr _S_maximum` (`_Const_Base_ptr __x`)
- static `_Base_ptr _S_minimum` (`_Base_ptr __x`)
- static `_Const_Base_ptr _S_minimum` (`_Const_Base_ptr __x`)
- static `_Const_Link_type _S_right` (`_Const_Base_ptr __x`)
- static `_Link_type _S_right` (`_Base_ptr __x`)
- static const\_reference `_S_value` (`_Const_Link_type __x`)
- static const\_reference `_S_value` (`_Const_Base_ptr __x`)

## Protected Attributes

- `_Rb_tree_impl<_Compare > _M_impl`

### 5.65.1 Detailed Description

```
template<class _Key, class _Value, class _KeyOfValue, class _Compare, class
_Alloc = allocator<_Value>> struct __gnu_cxx::rb_tree< _Key, _Value, _-
KeyOfValue, _Compare, _Alloc >
```

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 78 of file `rb_tree`.

The documentation for this struct was generated from the following file:

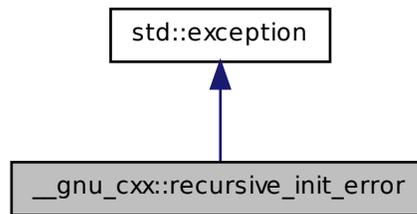
- [rb\\_tree](#)

## 5.66 `__gnu_cxx::recursive_init_error` Class Reference

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



## Public Member Functions

- virtual const char \* [what](#) () const throw ()

### 5.66.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined. Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 620 of file `cxxabi.h`.

### 5.66.2 Member Function Documentation

#### 5.66.2.1 virtual const char\* `std::exception::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::tr1::bad_weak_ptr`, and

[std::ios\\_base::failure](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 5.67 `__gnu_cxx::rope< _CharT, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::Rope_base< _CharT, _Alloc >`.

### Public Types

- typedef `_Rope_RopeConcatenation< _CharT, _Alloc >` **\_\_C**
- typedef `_Rope_RopeFunction< _CharT, _Alloc >` **\_\_F**
- typedef `_Rope_RopeLeaf< _CharT, _Alloc >` **\_\_L**
- typedef `_Rope_RopeSubstring< _CharT, _Alloc >` **\_\_S**
- typedef `_Alloc::template rebind< __C >::other` **\_CAlloc**
- typedef `_Alloc::template rebind< _CharT >::other` **\_DataAlloc**
- typedef `_Alloc::template rebind< __F >::other` **\_FAlloc**
- typedef `_Alloc::template rebind< __L >::other` **\_LAlloc**
- typedef `_Alloc::template rebind< __S >::other` **\_SAlloc**
- typedef `_Rope_const_iterator< _CharT, _Alloc >` **const\_iterator**
- typedef `const _CharT *` **const\_pointer**
- typedef `_CharT` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Rope_iterator< _CharT, _Alloc >` **iterator**
- typedef `_Rope_char_ptr_proxy< _CharT, _Alloc >` **pointer**
- typedef `_Rope_char_ref_proxy< _CharT, _Alloc >` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_CharT` **value\_type**

### Public Member Functions

- **rope** (`const _CharT * __s, const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s, size_t __len, const allocator_type & __a=allocator_type()`)
- **rope** (`const iterator & __s, const iterator & __e, const allocator_type & __a=allocator_type()`)

- **rope** (\_CharT \_\_c, const allocator\_type &\_\_a=allocator\_type())
- **rope** (size\_t \_\_n, \_CharT \_\_c, const allocator\_type &\_\_a=allocator\_type())
- **rope** (const allocator\_type &\_\_a=allocator\_type())
- **rope** (const \_CharT \*\_\_s, const \_CharT \*\_\_e, const allocator\_type &\_\_a=allocator\_type())
- **rope** (char\_producer< \_CharT > \*\_\_fn, size\_t \_\_len, bool \_\_delete\_fn, const allocator\_type &\_\_a=allocator\_type())
- **rope** (const rope &\_\_x, const allocator\_type &\_\_a=allocator\_type())
- **rope** (const const\_iterator &\_\_s, const const\_iterator &\_\_e, const allocator\_type &\_\_a=allocator\_type())
- allocator\_type & **M\_get\_allocator** ()
- const allocator\_type & **M\_get\_allocator** () const
- **rope** & **append** (const \_CharT \*\_\_iter, size\_t \_\_n)
- **rope** & **append** (const \_CharT \*\_\_c\_string)
- **rope** & **append** (const \_CharT \*\_\_s, const \_CharT \*\_\_e)
- **rope** & **append** (const\_iterator \_\_s, const\_iterator \_\_e)
- **rope** & **append** (\_CharT \_\_c)
- **rope** & **append** ()
- **rope** & **append** (const rope &\_\_y)
- **rope** & **append** (size\_t \_\_n, \_CharT \_\_c)
- void **apply\_to\_pieces** (size\_t \_\_begin, size\_t \_\_end, \_Rope\_char\_consumer< \_CharT > &\_\_c) const
- \_CharT **at** (size\_type \_\_pos) const
- \_CharT **back** () const
- void **balance** ()
- const\_iterator **begin** () const
- const\_iterator **begin** ()
- const \_CharT \* **c\_str** () const
- void **clear** ()
- int **compare** (const rope &\_\_y) const
- const\_iterator **const\_begin** () const
- const\_iterator **const\_end** () const
- const\_reverse\_iterator **const\_rbegin** () const
- const\_reverse\_iterator **const\_rend** () const
- void **copy** (\_CharT \*\_\_buffer) const
- size\_type **copy** (size\_type \_\_pos, size\_type \_\_n, \_CharT \*\_\_buffer) const
- void **delete\_c\_str** ()
- void **dump** ()
- bool **empty** () const
- const\_iterator **end** () const
- const\_iterator **end** ()
- void **erase** (size\_t \_\_p, size\_t \_\_n)
- void **erase** (size\_t \_\_p)

- iterator **erase** (const iterator &\_\_p, const iterator &\_\_q)
- iterator **erase** (const iterator &\_\_p)
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- \_CharT **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const rope &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_t \_\_n, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **insert** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **insert** (size\_t \_\_p)
- void **insert** (size\_t \_\_p, const rope &\_\_r)
- void **insert** (size\_t \_\_p, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()
- **reverse\_iterator mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- **reverse\_iterator mutable\_rend** ()
- rope & **operator=** (const rope &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- **const\_reverse\_iterator rbegin** ()
- **const\_reverse\_iterator rbegin** () const
- **const\_reverse\_iterator rend** ()
- **const\_reverse\_iterator rend** () const
- void **replace** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- void **replace** (const iterator &\_\_p, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const rope &\_\_r)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (size\_t \_\_p, const rope &\_\_r)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator \_\_i, const iterator \_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_t \_\_n)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- rope< \_CharT, \_Alloc > **substr** (const\_iterator \_\_start)
- rope **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- rope **substr** (iterator \_\_start) const
- rope **substr** (size\_t \_\_start, size\_t \_\_len=1) const
- rope **substr** (iterator \_\_start, iterator \_\_end) const
- void **swap** (rope &\_\_b)

## Static Public Member Functions

- static `__C * _C_allocate` (size\_t \_\_n)
- static void `_C_deallocate` (\_\_C \* \_\_p, size\_t \_\_n)
- static `_CharT * _Data_allocate` (size\_t \_\_n)
- static void `_Data_deallocate` (\_CharT \* \_\_p, size\_t \_\_n)
- static `__F * _F_allocate` (size\_t \_\_n)
- static void `_F_deallocate` (\_\_F \* \_\_p, size\_t \_\_n)
- static `__L * _L_allocate` (size\_t \_\_n)
- static void `_L_deallocate` (\_\_L \* \_\_p, size\_t \_\_n)
- static `__S * _S_allocate` (size\_t \_\_n)
- static void `_S_deallocate` (\_\_S \* \_\_p, size\_t \_\_n)

## Public Attributes

- `_RopeRep * _M_tree_ptr`

## Static Public Attributes

- static const size\_type `npos`

## Protected Types

- enum { `_S_copy_max` }
- typedef `_Rope_base<_CharT, _Alloc>` **`_Base`**
- typedef `_CharT * _Cstrptr`
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` **`_RopeConcatenation`**
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` **`_RopeFunction`**
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` **`_RopeLeaf`**
- typedef `_Rope_RopeRep<_CharT, _Alloc>` **`_RopeRep`**
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` **`_RopeSubstring`**
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc>` **`_Self_destruct_ptr`**
- typedef `_Base::allocator_type` **`allocator_type`**

## Static Protected Member Functions

- static size\_t `_S_allocated_capacity` (size\_t \_\_n)
- static bool `_S_apply_to_pieces` (\_Rope\_char\_consumer<\_CharT> &\_\_c, const \_RopeRep \* \_\_r, size\_t \_\_begin, size\_t \_\_end)
- static `_RopeRep * _S_concat` (\_RopeRep \* \_\_left, \_RopeRep \* \_\_right)

- static `_RopeRep * _S_concat_char_iter` (`_RopeRep * __r`, `const _CharT * __iter`, `size_t __slen`)
- static `_RopeRep * _S_destr_concat_char_iter` (`_RopeRep * __r`, `const _CharT * __iter`, `size_t __slen`)
- static `_RopeLeaf * _S_destr_leaf_concat_char_iter` (`_RopeLeaf * __r`, `const _CharT * __iter`, `size_t __slen`)
- static `_CharT _S_fetch` (`_RopeRep * __r`, `size_type __pos`)
- static `_CharT * _S_fetch_ptr` (`_RopeRep * __r`, `size_type __pos`)
- static `bool _S_is0` (`_CharT __c`)
- static `_RopeLeaf * _S_leaf_concat_char_iter` (`_RopeLeaf * __r`, `const _CharT * __iter`, `size_t __slen`)
- static `_RopeConcatenation * _S_new_RopeConcatenation` (`_RopeRep * __left`, `_RopeRep * __right`, `allocator_type & __a`)
- static `_RopeFunction * _S_new_RopeFunction` (`char_producer< _CharT > * __f`, `size_t __size`, `bool __d`, `allocator_type & __a`)
- static `_RopeLeaf * _S_new_RopeLeaf` (`_CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `_RopeSubstring * _S_new_RopeSubstring` (`_Rope_RopeRep< _CharT, _Alloc > * __b`, `size_t __s`, `size_t __l`, `allocator_type & __a`)
- static `void _S_ref` (`_RopeRep * __t`)
- static `_RopeLeaf * _S_RopeLeaf_from_unowned_char_ptr` (`const _CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `size_t _S_rounded_up_size` (`size_t __n`)
- static `_RopeRep * _S_substring` (`_RopeRep * __base`, `size_t __start`, `size_t __endp1`)
- static `_RopeRep * _S_tree_concat` (`_RopeRep * __left`, `_RopeRep * __right`)
- static `void _S_unref` (`_RopeRep * __t`)
- static `_RopeRep * replace` (`_RopeRep * __old`, `size_t __pos1`, `size_t __pos2`, `_RopeRep * __r`)

### Static Protected Attributes

- static `_CharT _S_empty_c_str` [1]

### Friends

- class `_Rope_char_ptr_proxy< _CharT, _Alloc >`
- class `_Rope_char_ref_proxy< _CharT, _Alloc >`
- class `_Rope_const_iterator< _CharT, _Alloc >`
- class `_Rope_iterator< _CharT, _Alloc >`
- class `_Rope_iterator_base< _CharT, _Alloc >`
- struct `_Rope_RopeRep< _CharT, _Alloc >`

- struct `_Rope_RopeSubstring< _CharT, _Alloc >`
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 >` **operator+** (const `rope< _CharT2, _Alloc2 >` &\_\_left, const `rope< _CharT2, _Alloc2 >` &\_\_right)
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 >` **operator+** (const `rope< _CharT2, _Alloc2 >` &\_\_left, `_CharT2` \_\_right)
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 >` **operator+** (const `rope< _CharT2, _Alloc2 >` &\_\_left, const `_CharT2 *`\_\_right)

### 5.67.1 Detailed Description

`template<class _CharT, class _Alloc> class __gnu_cxx::rope< _CharT, _Alloc >`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 1508 of file rope.

The documentation for this class was generated from the following files:

- `rope`
- `ropeimpl.h`

## 5.68 `__gnu_cxx::select1st< _Pair >` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select1st< _Pair >`.

### Public Types

- typedef `_Pair` `argument_type`
- typedef `_Pair::first_type` `result_type`

## Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`

### 5.68.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select1st< _Pair >`

An [SGI extension](#) .

Definition at line 199 of file `ext/functional`.

### 5.68.2 Member Typedef Documentation

**5.68.2.1** `typedef _Pair std::unary_function< _Pair , _Pair::first_type >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.68.2.2** `typedef _Pair::first_type std::unary_function< _Pair , _Pair::first_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.69 `__gnu_cxx::select2nd< _Pair >` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select2nd< _Pair >`.

### Public Types

- `typedef _Pair argument_type`

- `typedef _Pair::second_type result_type`

## Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

### 5.69.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select2nd< _Pair >`

An [SGI extension](#).

Definition at line 203 of file `ext/functional`.

### 5.69.2 Member Typedef Documentation

**5.69.2.1** `typedef _Pair std::unary_function< _Pair , _Pair::second_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.69.2.2** `typedef _Pair::second_type std::unary_function< _Pair , _Pair::second_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.70 `__gnu_cxx::slist< _Tp, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::_Slist_base< _Tp, _Alloc >`.

## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Slist\_iterator< \_Tp, const \_Tp &, const \_Tp \* > **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Slist\_iterator< \_Tp, \_Tp &, \_Tp \* > **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **slist** (const allocator\_type &\_\_a=allocator\_type())
- **slist** (size\_type \_\_n)
- template<class \_InputIterator >  
**slist** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- **slist** (size\_type \_\_n, const value\_type &\_\_x, const allocator\_type &\_\_a=allocator\_type())
- **slist** (const [slist](#) &\_\_x)
- template<class \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<class \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_fill\_assign** (size\_type \_\_n, const \_Tp &\_\_val)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_val)
- template<class \_InputIterator >  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **before\_begin** ()
- const\_iterator **before\_begin** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_pos)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase\_after** (iterator \_\_pos)

- iterator **erase\_after** (iterator \_\_before\_first, iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- iterator **insert** (iterator \_\_pos)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- size\_type **max\_size** () const
- void **merge** (slist &\_\_x)
- template<class \_StrictWeakOrdering >  
void **merge** (slist &, \_StrictWeakOrdering)
- **slist & operator=** (const slist &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** ()
- void **push\_front** (const value\_type &\_\_x)
- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **resize** (size\_type new\_size)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_pos, slist &\_\_x)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice\_after** (iterator \_\_pos, slist &\_\_x)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **swap** (slist &\_\_x)
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_pred)
- void **unique** ()

## Private Types

- typedef `_Alloc::template rebind< _Slist_node< _Tp > >::other` `_Node_alloc`

## Private Member Functions

- `_Slist_node_base * _M_erase_after` (`_Slist_node_base * __pos`)
- `_Slist_node_base * _M_erase_after` (`_Slist_node_base *`, `_Slist_node_base *`)
- `_Slist_node< _Tp > * _M_get_node` ()
- void `_M_put_node` (`_Slist_node< _Tp > * __p`)

## Private Attributes

- `_Slist_node_base _M_head`

### 5.70.1 Detailed Description

`template<class _Tp, class _Alloc = allocator<_Tp>> class __gnu_cxx::slist< _Tp, _Alloc >`

This is an SGI extension.

#### Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg000> for more.

Definition at line 291 of file `slist`.

The documentation for this class was generated from the following file:

- [slist](#)

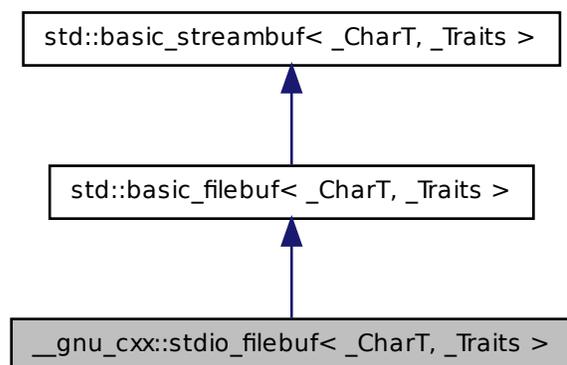
## 5.71 `__gnu_cxx::stdio_filebuf< _CharT, _Traits >` Class Template Reference

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 885

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



### Public Types

- typedef `codecvt<char_type, char, __state_type>` `__codecvt_type`
- typedef `__basic_file<char>` `__file_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::off_type` `off_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `std::size_t` `size_t`
- typedef `_Traits` `traits_type`

### Public Member Functions

- `stdio_filebuf()`
- `stdio_filebuf(int __fd, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`
- `stdio_filebuf(std::_c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast<size_t>(BUFSIZ))`

- virtual `~stdio_filebuf ()`
- `__filebuf_type * close ()`
- `int fd ()`
- `std::__c_file * file ()`
- `streamsize in_avail ()`
- `bool is_open () const throw ()`
- `__filebuf_type * open (const char *__s, ios_base::openmode __mode)`
- `__filebuf_type * open (const std::string &__s, ios_base::openmode __mode)`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type *__s, streamsize __n)`
- `void stoss ()`
- `int_type sungetc ()`

## Protected Member Functions

- `void _M_allocate_internal_buffer ()`
- `bool _M_convert_to_external (char_type *, streamsize)`
- `void _M_create_pback ()`
- `void _M_destroy_internal_buffer () throw ()`
- `void _M_destroy_pback () throw ()`
- `pos_type _M_seek (off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer (streamsize __off)`
- `bool _M_terminate_output ()`
- `void gbump (int __n)`
- `virtual void imbue (const locale &__loc)`
- `virtual int_type overflow (int_type __c=_Traits::eof())`
- `virtual int_type overflow (int_type=traits_type::eof())`
- `virtual int_type pbackfail (int_type=traits_type::eof())`
- `virtual int_type pbackfail (int_type __c=_Traits::eof())`
- `void pbump (int __n)`
- `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
- `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 887

- virtual `pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
- virtual `__streambuf_type * setbuf (char_type *__s, streamsize __n)`
- virtual `basic_streambuf<char_type, _Traits> * setbuf (char_type *, streamsize)`
- void `setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)`
- void `setp (char_type *__pbeg, char_type *__pend)`
- virtual `streamsize showmanyc ()`
- virtual `int sync ()`
- virtual `int_type uflow ()`
- virtual `int_type underflow ()`
- virtual `streamsize xsgetn (char_type *__s, streamsize __n)`
- virtual `streamsize xsgetn (char_type *__s, streamsize __n)`
- virtual `streamsize xspun (const char_type *__s, streamsize __n)`
- virtual `streamsize xspun (const char_type *__s, streamsize __n)`
  
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
  
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * epptr () const`

### Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `bool _M_reading`
- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
  
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`  
`_CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`  
`bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`  
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`  
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _`  
`Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
- `class ios_base`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, _CharT2 *)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
- `class ostreambuf_iterator< char_type, traits_type >`
  
- `locale pubimbue (const locale &__loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_`  
`base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_`  
`base::in|ios_base::out)`
- `int pubsync ()`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `locale _M_buf_locale`

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 889

### 5.71.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 49 of file `stdio_filebuf.h`.

### 5.71.2 Member Typedef Documentation

**5.71.2.1** `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_filebuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 77 of file `fstream`.

**5.71.2.2** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 53 of file `stdio_filebuf.h`.

**5.71.2.3** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 55 of file `stdio_filebuf.h`.

**5.71.2.4** `template<typename _CharT , typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::off_type  
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 57 of file `stdio_filebuf.h`.

**5.71.2.5** `template<typename _CharT , typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::pos_type  
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 56 of file `stdio_filebuf.h`.

**5.71.2.6** `template<typename _CharT , typename _Traits = std::char_traits<_  
CharT>> typedef _Traits __gnu_cxx::stdio_filebuf< _CharT, _Traits  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 54 of file `stdio_filebuf.h`.

### 5.71.3 Constructor & Destructor Documentation

**5.71.3.1** `template<typename _CharT , typename _Traits =  
std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf< _CharT,  
_Traits >::stdio_filebuf ( ) [inline]`

deferred initialization

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 891

Definition at line 64 of file `stdio_filebuf.h`.

```
5.71.3.2 template<typename _CharT, typename _Traits >
__gnu_cxx::stdio_filebuf<_CharT, _Traits >::stdio_filebuf
( int __fd, std::ios_base::openmode __mode, size_t __size =
  static_cast<size_t>(BUFSIZ) )
```

### Parameters

*fd* An open file descriptor.

*mode* Same meaning as in a standard filebuf.

*size* Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 126 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits >::M_buf_size`, `std::basic_filebuf<_CharT, _Traits >::M_mode`, `std::basic_filebuf<_CharT, _Traits >::M_reading`, `std::basic_filebuf<_CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits >::is_open()`.

```
5.71.3.3 template<typename _CharT, typename _Traits >
__gnu_cxx::stdio_filebuf<_CharT, _Traits >::stdio_filebuf (
  std::_c_file * __f, std::ios_base::openmode __mode, size_t __size =
  static_cast<size_t>(BUFSIZ) )
```

### Parameters

*f* An open `FILE*`.

*mode* Same meaning as in a standard filebuf.

*size* Optimal or preferred size of internal buffer, in chars. Defaults to system's `BUFSIZ`.

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 142 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits >::M_buf_size`, `std::basic_filebuf<_CharT, _Traits >::M_mode`, `std::basic_filebuf<_CharT, _Traits >::M_reading`, `std::basic_filebuf<_CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits >::is_open()`.

**5.71.3.4** `template<typename _CharT, typename _Traits >`  
`__gnu_cxx::stdio_filebuf< _CharT, _Traits >::~~stdio_filebuf ( )`  
`[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 121 of file `stdio_filebuf.h`.

## 5.71.4 Member Function Documentation

**5.71.4.1** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf< _CharT, _Traits >::_M_create_pback ( )`  
`[inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

**5.71.4.2** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback ( ) throw`  
`() [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.71.4.3** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf< _CharT, _Traits >::_M_set_buffer ( streamsize`  
`__off ) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr\(\) - eback\(\)` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr\(\) - pbase\(\) == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.`

Definition at line 387 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 893

**5.71.4.4** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::_filebuf_type * std::basic_filebuf<_CharT, _Traits>::close ( ) [inherited]`

Closes the currently associated file.

### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(EOF)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**5.71.4.5** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.4.6** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.71.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.71.4.8** `template<typename _CharT , typename _Traits =  
std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf< _CharT,  
_Traits >::fd ( ) [inline]`

#### Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 107 of file `stdio_filebuf.h`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 895

**5.71.4.9** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf<  
_CharT, _Traits >::file ( ) [inline]`

### Returns

The underlying FILE\*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 117 of file `stdio_filebuf.h`.

**5.71.4.10** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits >::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

**5.71.4.11** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits >::getloc ( ) const  
[inline, inherited]`

Locale access.

### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

**5.71.4.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits >::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.71.4.13** `template<typename _CharT, typename _Traits > void  
std::basic_filebuf< _CharT, _Traits >::imbue ( const locale & )  
[protected, virtual, inherited]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 855 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 897

**5.71.4.14** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline,  
inherited]`

Looking ahead into the stream.

### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file `streambuf`.

**5.71.4.15** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf<_CharT, _Traits>::is_open ( ) const throw ()  
[inline, inherited]`

Returns true if the external file is open.

Definition at line 222 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::close()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, and `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`.

**5.71.4.16** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits>::_filebuf_type * std::basic_filebuf<_CharT,  
_Traits>::open ( const char * __s, ios_base::openmode __mode )  
[inherited]`

Opens an external file.

### Parameters

*s* The name of the file.

*mode* The open mode flags.

### Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +-----

```

-----+ | ios_base Flag combination stdio equivalent | |binary in out trunc app
| +-----+ | + w | | + a | | + a | | + w | | +
r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +-----+
-----+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + +
w+b | | + + + + a+b | | + + + a+b | +-----+

```

Definition at line 94 of file fstream.tcc.

References std::basic\_filebuf< \_CharT, \_Traits >::\_M\_mode, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_reading, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer(), std::ios\_base::ate, std::basic\_filebuf< \_CharT, \_Traits >::close(), std::ios\_base::end, and std::basic\_filebuf< \_CharT, \_Traits >::is\_open().

**5.71.4.17** `template<typename _CharT, typename _Traits> __filebuf_type*  
std::basic_filebuf< _CharT, _Traits >::open ( const std::string &  
__s, ios_base::openmode __mode ) [inline, inherited]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

#### Returns

`this` on success, NULL on failure

Definition at line 275 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::open().

**5.71.4.18** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::overflow ( int_type =  
traits_type::eof() ) [inline, protected, virtual,  
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

- c* An additional character to consume.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 899

### Returns

`eof()` to indicate failure, something else (usually `c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 746 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xspn()`.

```
5.71.4.19 template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail( int_type =
traits_type::eof() ) [inline, protected, virtual,
inherited]
```

Tries to back up the input sequence.

### Parameters

`c` The character to be inserted back into the sequence.

### Returns

`eof()` on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 702 of file streambuf.

**5.71.4.20** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

**5.71.4.21** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

- *n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.71.4.22** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 901

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.71.4.23** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

**5.71.4.24** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

**5.71.4.25** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

**5.71.4.26** `template<typename _CharT, typename _Traits> __streambuf_type*  
std::basic_streambuf< _CharT, _Traits >::pubsetbuf ( char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubf00` dispatch to the protected derived `f00` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

**5.71.4.27** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline,  
inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 903

**5.71.4.28** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sbumpc( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

**5.71.4.29** `template<typename _CharT, typename _Traits> virtual  
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff  
( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [inline, protected,  
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

**5.71.4.30** `template<typename _CharT, typename _Traits> virtual pos_type  
std::basic_streambuf<_CharT, _Traits>::seekpos( pos_type,  
ios_base::openmode = ios_base::in | ios_base::out ) [inline,  
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file streambuf.

**5.71.4.31** `template<typename _CharT, typename _Traits> virtual  
basic_streambuf<char_type,_Traits>* std::basic_streambuf<  
_CharT,_Traits >::setbuf ( char_type *, streamsize ) [inline,  
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

#### Note

Base class version does nothing, returns `this`.

Definition at line 569 of file streambuf.

**5.71.4.32** `template<typename _CharT , typename _Traits > basic_filebuf<  
_CharT, _Traits >::_streambuf_type * std::basic_filebuf<  
_CharT, _Traits >::setbuf ( char_type * __s, streamsize __n )  
[protected, virtual, inherited]`

Manipulates the buffer.

#### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

#### Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 657 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 905

**5.71.4.33** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::setg ( char_type *  
__gbeg, char_type * __gnext, char_type * __gend ) [inline,  
protected, inherited]`

Setting the three read area pointers.

### Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

**5.71.4.34** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::setp ( char_type *  
__pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

### Parameters

*pbeg* A pointer.

*pend* A pointer.

### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.71.4.35** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sgetc ( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.71.4.36** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

#### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

**5.71.4.37** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::showmanyc ( )  
[protected, virtual, inherited]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 907

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 178 of file `fstream.tcc`.

References [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_mode](#), [std::ios\\_base::binary](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::egptr\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::gptr\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::is\\_open\(\)](#).

### **5.71.4.38 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::snextc( ) [inline, inherited]`**

Getting the next character.

#### **Returns**

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Definition at line 276 of file `streambuf`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

### **5.71.4.39 `template<typename _CharT, typename _Traits> int_type std::basic_streambuf<_CharT, _Traits>::sputbackc( char_type _c ) [inline, inherited]`**

Pushing characters back into the input stream.

#### **Parameters**

*c* The character to push back.

#### **Returns**

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#).

**5.71.4.40** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

#### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**5.71.4.41** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.71.4.42** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stoss ( ) [inline,  
inherited]`

Tosses a character.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 909

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.71.4.43** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::ungetc()`.

**5.71.4.44** `template<typename _CharT, typename _Traits> int  
std::basic_filebuf<_CharT, _Traits>::sync ( ) [protected,  
virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 838 of file `fstream.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

**5.71.4.45** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::uflow( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 678 of file `streambuf`.

**5.71.4.46** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits  
>::underflow( ) [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 911

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, and `std::min()`.

**5.71.4.47** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsgetn ( char_type * __s,  
streamsize __n ) [protected, virtual, inherited]`

Multiple character extraction.

### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

### Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

**5.71.4.48** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xspun ( const  
char_type * __s, streamsize __n ) [protected, virtual,  
inherited]`

Multiple character insertion.

### Parameters

- s* A buffer area.

*n* Maximum number of characters to write.

### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

## 5.71.5 Member Data Documentation

### 5.71.5.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf` [protected, inherited]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

### 5.71.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

### 5.71.5.3 `template<typename _CharT, typename _Traits> size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size` [protected, inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 913

Definition at line 116 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.4** `template<typename _CharT, typename _Traits> char*  
std::basic_filebuf<_CharT, _Traits>::_M_ext_buf` [**protected**,  
**inherited**]

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 151 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.5** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`  
[**protected**, **inherited**]

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.6** `template<typename _CharT, typename _Traits> const char*  
std::basic_filebuf<_CharT, _Traits>::_M_ext_next` [**protected**,  
**inherited**]

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 163 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_beg`  
[**protected**, **inherited**]

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

**5.71.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.71.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.71.5.10** `template<typename _CharT, typename _Traits> ios_base::openmode  
std::basic_filebuf< _CharT, _Traits >::_M_mode [protected,  
inherited]`

Place to stash `in || out || in | out` settings for current `filebuf`.

## 5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 915

Definition at line 94 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.71.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

**5.71.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

**5.71.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

**5.71.5.14** `template<typename _CharT, typename _Traits> char_type  
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected,  
inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

**5.71.5.15** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  
[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

**5.71.5.16** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  
[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

5.71.5.17 `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf<_CharT, _Traits>::_M_pback_init  
[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

5.71.5.18 `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf<_CharT, _Traits>::_M_reading [protected,  
inherited]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

The documentation for this class was generated from the following file:

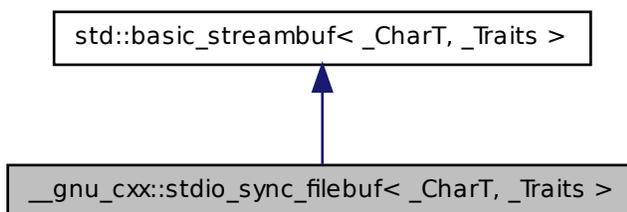
- [stdio\\_filebuf.h](#)

## 5.72 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` > Class Template Reference

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C `FILE*`'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >`:



## Public Types

- typedef `_CharT` `char_type`
  - typedef `traits_type::int_type` `int_type`
  - typedef `traits_type::off_type` `off_type`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef `_Traits` `traits_type`
- 
- typedef `basic_streambuf< char_type, traits_type >` `__streambuf_type`

## Public Member Functions

- `stdio_sync_filebuf` (`std::__c_file *__f`)
- `std::__c_file *const file` ()
- `streamsize in_avail` ()
- `int_type sbumpc` ()
- `int_type sgetc` ()
- `streamsize sgetn` (`char_type *__s`, `streamsize __n`)
- `int_type snextc` ()
- `int_type sputback` (`char_type __c`)
- `int_type sputc` (`char_type __c`)
- `streamsize sputn` (`const char_type *__s`, `streamsize __n`)
- void `stoss` ()
- `int_type sungetc` ()

## Protected Member Functions

- void `gbump` (int \_\_n)
- virtual void `imbue` (const locale &)
- virtual `int_type overflow` (`int_type` \_\_c=traits\_type::eof())
- virtual `int_type pbackfail` (`int_type` \_\_c=traits\_type::eof())
- void `pbump` (int \_\_n)
- virtual `std::streampos seekoff` (`std::streamoff` \_\_off, `std::ios_base::seekdir` \_\_dir, `std::ios_base::openmode`=`std::ios_base::in`|`std::ios_base::out`)
- virtual `pos_type seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode`=`ios_base::in`|`ios_base::out`)
- virtual `std::streampos seekpos` (`std::streampos` \_\_pos, `std::ios_base::openmode` \_\_mode=`std::ios_base::in`|`std::ios_base::out`)
- virtual `pos_type seekpos` (`pos_type`, `ios_base::openmode`=`ios_base::in`|`ios_base::out`)
- virtual `basic_streambuf<char_type, _Traits> * setbuf` (`char_type` \*, `streamsize`)
- void `setg` (`char_type` \*\_\_gbeg, `char_type` \*\_\_gnext, `char_type` \*\_\_gend)
- void `setp` (`char_type` \*\_\_pbeg, `char_type` \*\_\_pend)
- virtual `streamsize showmanyc` ()
- virtual `int sync` ()
- template<>  
`stdio_sync_filebuf<wchar_t>::int_type syncgetc` ()
- template<>  
`stdio_sync_filebuf<char>::int_type syncgetc` ()
- `int_type syncgetc` ()
- template<>  
`stdio_sync_filebuf<wchar_t>::int_type syncputc` (`int_type` \_\_c)
- template<>  
`stdio_sync_filebuf<char>::int_type syncputc` (`int_type` \_\_c)
- `int_type syncputc` (`int_type` \_\_c)
- template<>  
`stdio_sync_filebuf<wchar_t>::int_type syncungetc` (`int_type` \_\_c)
- `int_type syncungetc` (`int_type` \_\_c)
- template<>  
`stdio_sync_filebuf<char>::int_type syncungetc` (`int_type` \_\_c)
- virtual `int_type uflow` ()
- virtual `int_type underflow` ()
- template<>  
`std::streamsize xsgetn` (`char` \*\_\_s, `std::streamsize` \_\_n)
- virtual `std::streamsize xsgetn` (`char_type` \*\_\_s, `std::streamsize` \_\_n)
- template<>  
`std::streamsize xsgetn` (`wchar_t` \*\_\_s, `std::streamsize` \_\_n)
- template<>  
`std::streamsize xspun` (const `char` \*\_\_s, `std::streamsize` \_\_n)

- `template<>`  
`std::streamsize xsputn` (const `wchar_t` \* \_\_s, `std::streamsize` \_\_n)
- virtual `std::streamsize xsputn` (const `char_type` \* \_\_s, `std::streamsize` \_\_n)
  - `char_type` \* `eback` () const
  - `char_type` \* `gptr` () const
  - `char_type` \* `egptr` () const
  - `char_type` \* `pbase` () const
  - `char_type` \* `pptr` () const
  - `char_type` \* `epptr` () const

## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2` (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator<`  
`_CharT2 >`, `_CharT2 *`)
- `streamsize __copy_streambufs_eof` (`__streambuf_type` \*, `__streambuf_type` \*,  
`bool` &)
- class `basic_ios< char_type, traits_type >`
- class `basic_istream< char_type, traits_type >`
- class `basic_ostream< char_type, traits_type >`
- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`  
`iterator< _CharT2 > >::__type find` (`istreambuf_iterator< _CharT2 >`,  
`istreambuf_iterator< _CharT2 >`, const `_CharT2` &)
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline` (`basic_istream< _CharT2, _`  
`Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`, `_CharT2`)
- class `istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2,`  
`_Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`)
- `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2,`  
`_Traits2 > &`, `_CharT2 *`)
- class `ostreambuf_iterator< char_type, traits_type >`
- locale `pubimbue` (const locale & \_\_loc)
- locale `getloc` () const
- `__streambuf_type` \* `pubsetbuf` (`char_type` \* \_\_s, `streamsize` \_\_n)

## 5.72 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

921

- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `locale _M_buf_locale`

### 5.72.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 55 of file `stdio_sync_filebuf.h`.

### 5.72.2 Member Typedef Documentation

**5.72.2.1** `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 133 of file `streambuf`.

**5.72.2.2** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 59 of file `stdio_sync_filebuf.h`.

**5.72.2.3** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 61 of file `stdio_sync_filebuf.h`.

**5.72.2.4** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::off_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 63 of file `stdio_sync_filebuf.h`.

**5.72.2.5** `template<typename _CharT , typename _Traits = std::char_traits<_CharT>> typedef traits_type::pos_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 62 of file `stdio_sync_filebuf.h`.

5.72.2.6 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _Traits __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 60 of file `stdio_sync_filebuf.h`.

### 5.72.3 Member Function Documentation

5.72.3.1 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.72.3.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.72.3.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.72.3.4** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> std::_c_file* const  
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ( )  
[inline]`

### Returns

The underlying FILE\*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 87 of file `stdio_sync_filebuf.h`.

**5.72.3.5** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

**5.72.3.6** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::getloc ( ) const  
[inline, inherited]`

Locale access.

### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

**5.72.3.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr ( ) const [inline,  
protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.72.3.8** `template<typename _CharT, typename _Traits> virtual void  
std::basic_streambuf<_CharT, _Traits>::imbue ( const locale & )  
[inline, protected, virtual, inherited]`

Changes translations.

**Parameters**

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits >](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT >>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type >](#).

Definition at line 554 of file streambuf.

**5.72.3.9** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits >::in_avail ( ) [inline,  
inherited]`

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

**5.72.3.10** `template<typename _CharT , typename _Traits  
= std::char_traits<_CharT>> virtual int_type  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >::overflow (   
int_type = traits_type::eof() ) [inline, protected,  
virtual]`

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

*c* An additional character to consume.

**Returns**

eof() to indicate failure, something else (usually *c*, or not\_eof())

## 5.72 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference

927

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 140 of file `stdio_sync_filebuf.h`.

```
5.72.3.11 template<typename _CharT, typename _Traits
= std::char_traits<_CharT>> virtual int_type
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail (
int_type = traits_type::eof() ) [inline, protected,
virtual]
```

Tries to back up the input sequence.

### Parameters

*c* The character to be inserted back into the sequence.

### Returns

`eof()` on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 115 of file `stdio_sync_filebuf.h`.

**5.72.3.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

**5.72.3.13** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

- *n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.72.3.14** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.72.3.15** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

**5.72.3.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

**5.72.3.17** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits >::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

**5.72.3.18** `template<typename _CharT, typename _Traits> __streambuf_type*  
std::basic_streambuf<_CharT, _Traits >::pubsetbuf ( char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubf00` dispatch to the protected derived `f00` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

**5.72.3.19** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits >::pubsync ( ) [inline,  
inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits >::sync()`.

**5.72.3.20** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sbumpc( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

**5.72.3.21** `template<typename _CharT, typename _Traits> virtual  
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff  
( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [inline, protected,  
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

**5.72.3.22** `template<typename _CharT, typename _Traits> virtual pos_type  
std::basic_streambuf<_CharT, _Traits>::seekpos( pos_type,  
ios_base::openmode = ios_base::in | ios_base::out ) [inline,  
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file streambuf.

```
5.72.3.23 template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits >::setbuf ( char_type *, streamsize ) [inline,
protected, virtual, inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

#### Note

Base class version does nothing, returns `this`.

Definition at line 569 of file streambuf.

```
5.72.3.24 template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT,_Traits >::setg ( char_type *
__gbeg, char_type * __gnext, char_type * __gend ) [inline,
protected, inherited]
```

Setting the three read area pointers.

#### Parameters

*gbeg* A pointer.  
*gnext* A pointer.  
*gend* A pointer.

#### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

```
5.72.3.25 template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT,_Traits >::setp ( char_type *
__pbeg, char_type * __pend ) [inline, protected,
inherited]
```

Setting the three write area pointers.

**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.72.3.26** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sgetc( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.72.3.27** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

**Parameters**

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

**5.72.3.28** `template<typename _CharT, typename _Traits> virtual streamsize  
std::basic_streambuf< _CharT, _Traits >::showmanyc ( )  
[inline, protected, virtual, inherited]`

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
[27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 627 of file `streambuf`.

**5.72.3.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or `eof`.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.72.3.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputbackc ( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

**Parameters**

*c* The character to push back.

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.72.3.31** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*c* A character to output.

**Returns**

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.72.3.32** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type *  
_s, streamsize _n ) [inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspun(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.72.3.33** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stoss ( ) [inline,  
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.72.3.34** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.72.3.35** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync( void ) [inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 159 of file `stdio_sync_filebuf.h`.

**5.72.3.36** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow( ) [inline, protected, virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 107 of file `stdio_sync_filebuf.h`.

**5.72.3.37** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow( ) [inline, protected, virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

Definition at line 100 of file `stdio_sync_filebuf.h`.

```
5.72.3.38 template<typename _CharT , typename _Traits =
std::char_traits<_CharT>> virtual std::streamsize
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >::xsgetn (
char_type * __s, std::streamsize __n ) [protected, virtual]
```

Multiple character extraction.

**Parameters**

*s* A buffer area.

*n* Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by [sbumpc\(\)](#). Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

**5.72.3.39** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual std::streamsize __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xspn( const char_type * __s, std::streamsize __n ) [protected, virtual]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

### 5.72.4 Member Data Documentation

**5.72.4.1** `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected, inherited]`

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

**5.72.4.2** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg [protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

**5.72.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.72.4.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.72.4.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

**5.72.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_cur  
[protected, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

**5.72.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_end  
[protected, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

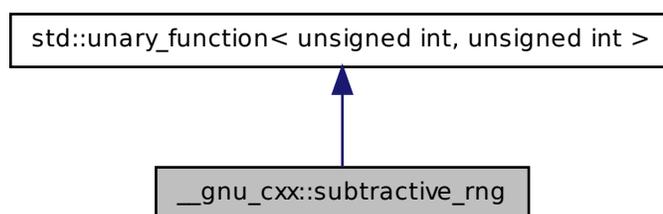
Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

## 5.73 `__gnu_cxx::subtractive_rng` Class Reference

Inheritance diagram for `__gnu_cxx::subtractive_rng`:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- [subtractive\\_rng](#) (unsigned int `__seed`)
- [subtractive\\_rng](#) ()
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

#### 5.73.1 Detailed Description

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 348 of file `ext/functional`.

## 5.73.2 Member Typedef Documentation

**5.73.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.73.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

## 5.73.3 Constructor & Destructor Documentation

**5.73.3.1** `__gnu_cxx::subtractive_rng::subtractive_rng ( unsigned int __seed )  
[inline]`

Ctor allowing you to initialize the seed.

Definition at line 390 of file `ext/functional`.

**5.73.3.2** `__gnu_cxx::subtractive_rng::subtractive_rng ( ) [inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 394 of file `ext/functional`.

## 5.73.4 Member Function Documentation

**5.73.4.1** `unsigned int __gnu_cxx::subtractive_rng::operator() ( unsigned int  
__limit ) [inline]`

Returns a number less than the argument.

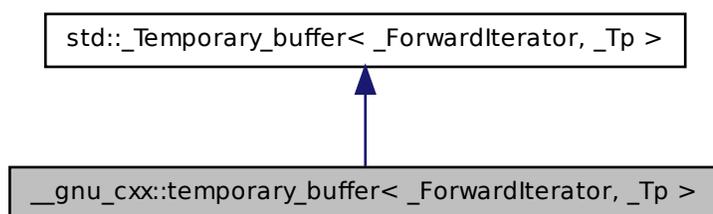
Definition at line 359 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.74 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- [temporary\\_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- [~temporary\\_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

### 5.74.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type> struct __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp >
```

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

Definition at line 182 of file `ext/memory`.

### 5.74.2 Constructor & Destructor Documentation

```
5.74.2.1 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp
>::temporary_buffer ( _ForwardIterator __first, _ForwardIterator
__last ) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 185 of file `ext/memory`.

```
5.74.2.2 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp
>::~~temporary_buffer ( ) [inline]
```

Destroys objects and frees storage.

Definition at line 189 of file `ext/memory`.

### 5.74.3 Member Function Documentation

**5.74.3.1** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( )  
[inline, inherited]`

As per Table mumble.

Definition at line 151 of file stl\_tempbuf.h.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

**5.74.3.2** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( )  
[inline, inherited]`

As per Table mumble.

Definition at line 156 of file stl\_tempbuf.h.

**5.74.3.3** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( )  
const [inline, inherited]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file stl\_tempbuf.h.

Referenced by `std::stable_partition()`.

**5.74.3.4** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const  
[inline, inherited]`

As per Table mumble.

Definition at line 141 of file stl\_tempbuf.h.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

## 5.75 `__gnu_cxx::throw_allocator_base< _Tp, _Cond >` > Class Template Reference

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

Inheritance diagram for `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`:



### Public Types

- typedef `const value_type * const_pointer`
- typedef `const value_type & const_reference`
- typedef `ptrdiff_t difference_type`
- typedef `value_type * pointer`
- typedef `value_type & reference`
- typedef `size_t size_type`
- typedef `_Tp value_type`

### Public Member Functions

- pointer `allocate` (`size_type __n`, `std::allocator< void >::const_pointer hint=0`)
- void `check_allocated` (`size_type __n`)
- void `check_allocated` (`void *p`, `size_t size`)
- void `check_allocated` (`pointer __p`, `size_type __n`)
- void `check_allocated` (`size_t label`)
- template<typename... `_Args`>  
 void `construct` (`pointer __p`, `_Args &&...__args`)
- void `construct` (`pointer __p`, `const value_type &val`)
- void `deallocate` (`pointer __p`, `size_type __n`)
- void `destroy` (`pointer __p`)
- void `erase` (`void *p`, `size_t size`)
- void `insert` (`void *p`, `size_t size`)
- `size_type max_size` () const throw ()

## Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### 5.75.1 Detailed Description

**template<typename \_Tp, typename \_Cond> class \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >**

Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.

Note: Deallocate not allowed to throw.

Definition at line 598 of file throw\_allocator.h.

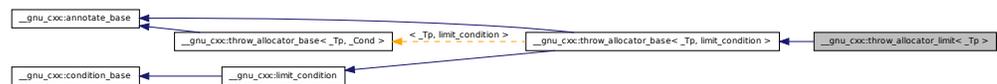
The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

## 5.76 \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp > Struct Template Reference

Allocator throwing via limit condition.

Inheritance diagram for \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >:



## Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) throw ()
- `template<typename _Tp1 >`  
**throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)<\_Tp1 > &) throw ()
- pointer **allocate** (size\_type \_\_n, `std::allocator< void >::const_pointer` hint=0)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_type \_\_n)
- void **check\_allocated** (size\_t label)
- void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const value\_type &val)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (pointer \_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const throw ()

## Static Public Member Functions

- static size\_t & **count** ()
- static size\_t **get\_label** ()
- static size\_t & **limit** ()
- static void **set\_label** (size\_t l)
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

### 5.76.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_limit<_Tp >`

Allocator throwing via limit condition.

Definition at line 682 of file `throw_allocator.h`.

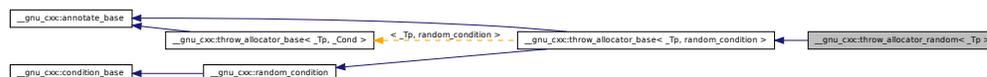
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.77 `__gnu_cxx::throw_allocator_random< _Tp >` Struct Template Reference

Allocator throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_allocator_random< _Tp >`:



### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) throw ()
- template<typename \_Tp1 >  
**throw\_allocator\_random** (const [throw\\_allocator\\_random](#)< \_Tp1 > &) throw ()
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_t label)
- void **construct** (pointer \_\_p, const value\_type &val)
- void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (pointer \_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const throw ()
- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static `size_t` `get_label` ()
- static void `set_label` (`size_t` l)
- static void `set_probability` (`double` \_\_p)
- static void `throw_conditionally` ()

#### 5.77.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_random<_Tp>`

Allocator throwing via random condition.

Definition at line 701 of file `throw_allocator.h`.

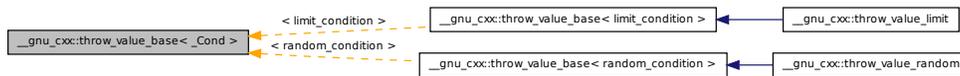
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.78 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Inheritance diagram for `__gnu_cxx::throw_value_base<_Cond>`:



### Public Types

- typedef `_Cond` `condition_type`

### Public Member Functions

- `throw_value_base` (`const` [throw\\_value\\_base](#) &\_\_v)
- `throw_value_base` (`const` `std::size_t` \_\_i)

- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_base](#) & **operator=** (const [throw\\_value\\_base](#) &\_\_v)

## Public Attributes

- `std::size_t_M_i`

### 5.78.1 Detailed Description

```
template<typename _Cond> struct __gnu_cxx::throw_value_base< _Cond >
```

Class with exception generation control. Intended to be used as a `value_type` in templated code. Note: Destructor not allowed to throw.

Definition at line 453 of file `throw_allocator.h`.

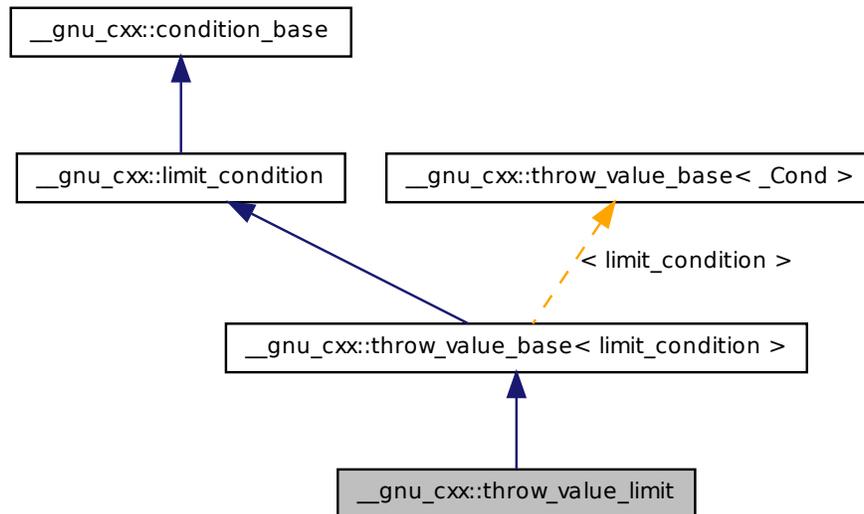
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.79 `__gnu_cxx::throw_value_limit` Struct Reference

Type throwing via limit condition.

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



## Public Types

- typedef `throw_value_base< limit_condition > base_type`
- typedef `limit_condition condition_type`

## Public Member Functions

- `throw_value_limit` (const `throw_value_limit` &\_\_other)
- `throw_value_limit` (const `std::size_t` \_\_i)
- `throw_value_base` & `operator++` ()

## Static Public Member Functions

- static `size_t` & `count` ()
- static `size_t` & `limit` ()
- static void `set_limit` (const `size_t` \_\_l)
- static void `throw_conditionally` ()

## Public Attributes

- `std::size_t _M_i`

### 5.79.1 Detailed Description

Type throwing via limit condition.

Definition at line 559 of file `throw_allocator.h`.

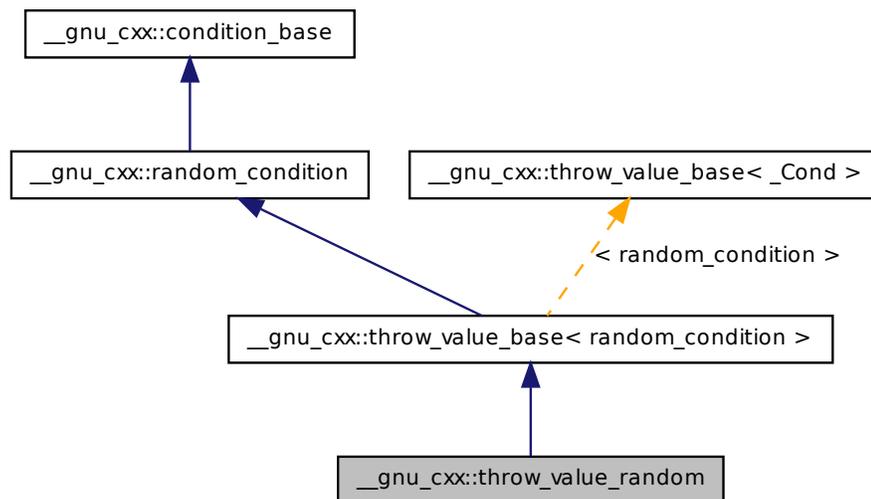
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.80 `__gnu_cxx::throw_value_random` Struct Reference

Type throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_value_random`:



## Public Types

- typedef `throw_value_base< random_condition >` `base_type`
- typedef `random_condition` `condition_type`

## Public Member Functions

- `throw_value_random` (const `throw_value_random` &\_\_other)
- `throw_value_random` (const `std::size_t` \_\_i)
- `throw_value_base` & `operator++` ()
- void `seed` (unsigned long \_\_s)

## Static Public Member Functions

- static void `set_probability` (double \_\_p)
- static void `throw_conditionally` ()

## Public Attributes

- `std::size_t` `M_i`

### 5.80.1 Detailed Description

Type throwing via random condition.

Definition at line 574 of file `throw_allocator.h`.

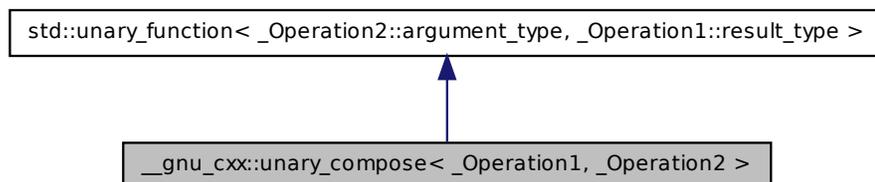
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.81 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

An [SGI extension](#) .

Inheritance diagram for `__gnu_cxx::unary_compose<_Operation1, _Operation2 >`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- **unary\_compose** (const `_Operation1` &\_\_x, const `_Operation2` &\_\_y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &\_\_x) const

## Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**

### 5.81.1 Detailed Description

```
template<class _Operation1, class _Operation2> class __gnu_cxx::unary_compose<_Operation1, _Operation2 >
```

An [SGI extension](#) .

Definition at line 125 of file `ext/functional`.

## 5.82 `__gnu_debug::__is_same<_Type1, _Type2 >` Struct Template Reference957

### 5.81.2 Member Typedef Documentation

**5.81.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function<_Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.81.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function<_Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.82 `__gnu_debug::__is_same<_Type1, _Type2 >` Struct Template Reference

### Static Public Attributes

- static const bool `value`

### 5.82.1 Detailed Description

`template<typename _Type1, typename _Type2> struct __gnu_debug::__is_  
same<_Type1, _Type2 >`

Determine if the two types are the same.

Definition at line 43 of file `formatter.h`.

The documentation for this struct was generated from the following file:

- [formatter.h](#)

## 5.83 `__gnu_debug::_After_nth_from< _Iterator >` Class Template Reference

### Public Member Functions

- `_After_nth_from` (const difference\_type &\_\_n, const \_Iterator &\_\_base)
- `bool operator()` (const \_Iterator &\_\_x) const

#### 5.83.1 Detailed Description

```
template<typename _Iterator> class __gnu_debug::_After_nth_from< _Iterator >
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 63 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.84 `__gnu_debug::_Not_equal_to< _Type >` Class Template Reference

### Public Member Functions

- `_Not_equal_to` (const \_Type &\_\_v)
- `bool operator()` (const \_Type &\_\_x) const

#### 5.84.1 Detailed Description

```
template<typename _Type> class __gnu_debug::_Not_equal_to< _Type >
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 48 of file `safe_sequence.h`.

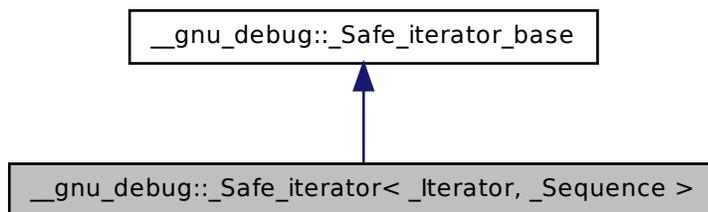
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.85 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >` Class Template Reference

Safe iterator wrapper.

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >`:



### Public Types

- typedef `_Iterator` **Base\_iterator**
- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::iterator_category` **iterator\_category**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value\_type**

### Public Member Functions

- `_Safe_iterator` ()
- `_Safe_iterator` (const `_Iterator` &`_i`, const `_Sequence` \*`__seq`)
- template<typename `_MutableIterator` >  
`_Safe_iterator` (const `_Safe_iterator`< `_MutableIterator`, typename `__gnu_cxx::__enable_if`<(std::`__are_same`< `_MutableIterator`, typename `_Sequence::iterator::Base_iterator` >::`__value`), `_Sequence` >::`__type` > &`_x`)
- `_Safe_iterator` (const `_Safe_iterator` &`_x`)
- void `_M_attach` (const `_Sequence` \*`__seq`)
- void `_M_attach` (`_Safe_sequence_base` \*`__seq`, bool `__constant`)

- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant) throw ()
- void [\\_M\\_attach\\_single](#) (const [\\_Sequence](#) \*\_\_seq)
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_can\\_advance](#) (const [difference\\_type](#) &\_\_n) const
- [\\_GLIBCXX\\_PURE](#) bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- bool [\\_M\\_decrementable](#) () const
- bool [\\_M\\_dereferenceable](#) () const
- void [\\_M\\_detach](#) ()
- void [\\_M\\_detach\\_single](#) () throw ()
- const [\\_Sequence](#) \* [\\_M\\_get\\_sequence](#) () const
- bool [\\_M\\_incrementable](#) () const
- void [\\_M\\_invalidate](#) ()
- void [\\_M\\_invalidate\\_single](#) ()
- bool [\\_M\\_is\\_begin](#) () const
- bool [\\_M\\_is\\_end](#) () const
- [\\_GLIBCXX\\_PURE](#) bool [\\_M\\_singular](#) () const throw ()
- template<typename [\\_Other](#) >  
bool [\\_M\\_valid\\_range](#) (const [\\_Safe\\_iterator](#)< [\\_Other](#), [\\_Sequence](#) > &\_\_rhs) const
- [\\_Iterator base](#) () const
- operator [\\_Iterator](#) () const
- reference operator\* () const
- [\\_Safe\\_iterator](#) **operator+** (const [difference\\_type](#) &\_\_n) const
- [\\_Safe\\_iterator](#) & **operator++** ()
- [\\_Safe\\_iterator](#) **operator++** (int)
- [\\_Safe\\_iterator](#) & **operator+=** (const [difference\\_type](#) &\_\_n)
- [\\_Safe\\_iterator](#) **operator-** (const [difference\\_type](#) &\_\_n) const
- [\\_Safe\\_iterator](#) **operator--** (int)
- [\\_Safe\\_iterator](#) & **operator--** ()
- [\\_Safe\\_iterator](#) & **operator-=** (const [difference\\_type](#) &\_\_n)
- pointer **operator->** () const
- [\\_Safe\\_iterator](#) & **operator=** (const [\\_Safe\\_iterator](#) &\_\_x)
- reference **operator[]** (const [difference\\_type](#) &\_\_n) const

## Static Public Member Functions

- template<typename [\\_Iterator1](#) , typename [\\_Iterator2](#) >  
static [std::pair](#)< [difference\\_type](#), [\\_Distance\\_precision](#) > [\\_M\\_get\\_distance](#)  
(const [\\_Iterator1](#) &\_\_lhs, const [\\_Iterator2](#) &\_\_rhs)
- template<typename [\\_Iterator1](#) , typename [\\_Iterator2](#) >  
static [std::pair](#)< [difference\\_type](#), [\\_Distance\\_precision](#) > [\\_M\\_get\\_distance](#)  
(const [\\_Iterator1](#) &\_\_lhs, const [\\_Iterator2](#) &\_\_rhs, [std::forward\\_iterator\\_tag](#))

- `template<typename _Iterator1 , typename _Iterator2 >`  
  static `std::pair`< `difference_type`, `_Distance_precision` > `_M_get_distance`  
  (`const _Iterator1 &_lhs`, `const _Iterator2 &_rhs`, `std::random_access_iterator_tag`)

## Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

## Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex ()` throw ()

### 5.85.1 Detailed Description

`template<typename _Iterator, typename _Sequence> class __gnu_debug::Safe_iterator`< `_Iterator`, `_Sequence` >

Safe iterator wrapper. The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 63 of file `safe_iterator.h`.

### 5.85.2 Constructor & Destructor Documentation

**5.85.2.1** `template<typename _Iterator, typename _Sequence>`  
`__gnu_debug::Safe_iterator`< `_Iterator`, `_Sequence` >::`Safe_iterator`  
( ) [`inline`]

#### Postcondition

the iterator is singular and unattached

Definition at line 99 of file `safe_iterator.h`.

**5.85.2.2** `template<typename _Iterator, typename _Sequence>`  
`__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_Safe_iterator`  
`( const _Iterator & __i, const _Sequence * __seq ) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

Definition at line 108 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**5.85.2.3** `template<typename _Iterator, typename _Sequence>`  
`__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_Safe_iterator`  
`( const _Safe_iterator<_Iterator, _Sequence > & __x ) [inline]`

Copy construction.

Definition at line 119 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**5.85.2.4** `template<typename _Iterator, typename _Sequence>`  
`template<typename _MutableIterator > __gnu_debug::_-`  
`Safe_iterator<_Iterator, _Sequence >::_Safe_iterator`  
`( const _Safe_iterator<_MutableIterator, typename`  
`__gnu_cxx::enable_if<(std::_are_same<_MutableIterator,`  
`typename _Sequence::iterator::_Base_iterator >::_value), _Sequence`  
`>::_type > & __x ) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 136 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

### 5.85.3 Member Function Documentation

**5.85.3.1** `template<typename _Iterator, typename _Sequence> void  
__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_attach (  
const _Sequence * __seq ) [inline]`

Attach iterator to the given sequence.

Definition at line 321 of file `safe_iterator.h`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=()`.

**5.85.3.2** `void __gnu_debug::Safe_iterator_base::_M_attach (  
_Safe_sequence_base * __seq, bool __constant ) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_iterator_base::_Safe_iterator_base()`.

**5.85.3.3** `template<typename _Iterator, typename _Sequence>  
void __gnu_debug::Safe_iterator<_Iterator, _Sequence  
>::_M_attach_single ( const _Sequence * __seq ) [inline]`

Likewise, but not thread-safe.

Definition at line 329 of file `safe_iterator.h`.

**5.85.3.4** `void __gnu_debug::Safe_iterator_base::_M_attach_single  
( _Safe_sequence_base * __seq, bool __constant ) throw ()  
[inherited]`

Likewise, but not thread-safe.

**5.85.3.5** `bool __gnu_debug::Safe_iterator_base::_M_attached_to ( const  
_Safe_sequence_base * __seq ) const [inline, inherited]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::_M_sequence`.

**5.85.3.6** `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_can_compare ( const _Safe_iterator_base & __x ) const throw ()` **[inherited]**

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

**5.85.3.7** `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable ( ) const` **[inline]**

Is the iterator dereferenceable?

Definition at line 345 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_check_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`.

**5.85.3.8** `void __gnu_debug::_Safe_iterator_base::_M_detach ( )` **[inherited]**

Detach the iterator for whatever sequence it is attached to, if any.

**5.85.3.9** `void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw ()` **[inherited]**

Likewise, but not thread-safe.

**5.85.3.10** `template<typename _Iterator, typename _Sequence> template<typename _Iterator1 , typename _Iterator2 > static std::pair<difference_type, _Distance_precision> __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_get_distance ( const _Iterator1 & __lhs, const _Iterator2 & __rhs )` **[inline, static]**

Determine the distance between two iterators with some known precision.

Definition at line 375 of file `safe_iterator.h`.

## 5.85 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference

965

### 5.85.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex ( ) throw ()` [`protected`, `inherited`]

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_invalidate()`.

### 5.85.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable ( ) const` [`inline`]

Is the iterator incrementable?

Definition at line 350 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++()`.

### 5.85.3.13 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_invalidate ( )`

Invalidate the iterator, making it singular.

Definition at line 106 of file `safe_iterator.tcc`.

References `__gnu_debug::Safe_iterator_base::M_get_mutex()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_invalidate_single()`.

### 5.85.3.14 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_invalidate_single ( )`

Likewise, but not thread-safe.

Definition at line 115 of file `safe_iterator.tcc`.

References `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_next`, `__gnu_debug::Safe_iterator_base::M_sequence`, `__gnu_debug::Safe_iterator_base::M_singular()`, `__gnu_debug::Safe_iterator_base::M_version`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

**5.85.3.15** `template<typename _Iterator, typename _Sequence> bool  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin  
( ) const [inline]`

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 400 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

**5.85.3.16** `template<typename _Iterator, typename _Sequence> bool  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end ( ) const [inline]`

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 404 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**5.85.3.17** `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw () [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular()`, `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

**5.85.3.18** `template<typename _Iterator, typename _Sequence> _Iterator  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base ( ) const [inline]`

Return the underlying iterator.

Definition at line 311 of file `safe_iterator.h`.

## 5.85 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >` Class Template Reference 967

---

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.85.3.19** `template<typename _Iterator, typename _Sequence>`  
`__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::operator`  
`_Iterator ( ) const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 317 of file `safe_iterator.h`.

**5.85.3.20** `template<typename _Iterator, typename _Sequence> reference`  
`__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::operator*` (  
`) const [inline]`

Iterator dereference.

### Precondition

iterator is dereferenceable

Definition at line 175 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_dereferenceable()`.

**5.85.3.21** `template<typename _Iterator, typename _Sequence> _Safe_iterator`  
`__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::operator++ (`  
`int ) [inline]`

Iterator postincrement.

### Precondition

iterator is incrementable

Definition at line 218 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_incrementable()`.

**5.85.3.22** `template<typename _Iterator, typename _Sequence>  
 _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
 >::operator++( ) [inline]`

Iterator preincrement.

**Precondition**

iterator is incrementable

Definition at line 204 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _-  
 Iterator, _Sequence >::_M_incrementable()`.

**5.85.3.23** `template<typename _Iterator, typename _Sequence> _Safe_iterator  
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--(   
 int ) [inline]`

Iterator postdecrement.

**Precondition**

iterator is decrementable

Definition at line 248 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.85.3.24** `template<typename _Iterator, typename _Sequence>  
 _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
 >::operator--( ) [inline]`

Iterator predecrement.

**Precondition**

iterator is decrementable

Definition at line 234 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.85.3.25** `template<typename _Iterator, typename _Sequence> pointer  
 __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->(   
 ) const [inline]`

Iterator dereference.

**Precondition**

iterator is dereferenceable

**Todo**

Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

Definition at line 190 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

```
5.85.3.26 template<typename _Iterator, typename _Sequence>
    _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
    >::operator=( const _Safe_iterator< _Iterator, _Sequence > & __x
    ) [inline]
```

Copy assignment.

Definition at line 156 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach()`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**5.85.4 Member Data Documentation**

```
5.85.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
[inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

```
5.85.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior
[inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file safe\_base.h.

#### 5.85.4.3 `_Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::_M_attached_to()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_is_begin()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_is_end()`, `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`, `__gnu_debug::Safe_iterator_base::_M_base()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator=()`.

#### 5.85.4.4 `unsigned int __gnu_debug::Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`.

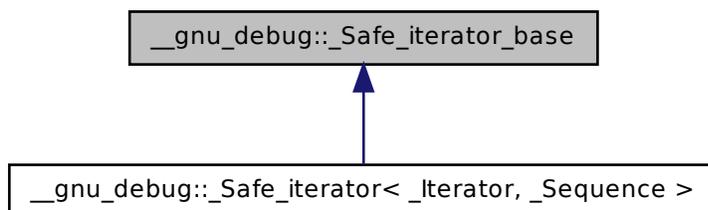
The documentation for this class was generated from the following files:

- [safe\\_iterator.h](#)
- [safe\\_iterator.tcc](#)

## 5.86 `__gnu_debug::Safe_iterator_base` Class Reference

Basic functionality for a *safe* iterator.

Inheritance diagram for `__gnu_debug::_Safe_iterator_base`:



### Public Member Functions

- `void _M_attach (_Safe_sequence_base * __seq, bool __constant)`
- `void _M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`
- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `_GLIBCXX_PURE bool _M_can_compare (const _Safe_iterator_base & __x) const throw ()`
- `void _M_detach ()`
- `void _M_detach_single () throw ()`
- `_GLIBCXX_PURE bool _M_singular () const throw ()`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `_Safe_iterator_base ()`
- `_Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_iterator_base (const _Safe_iterator_base &)`
- `_Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `_Safe_iterator_base & operator= (const _Safe_iterator_base &)`

### 5.86.1 Detailed Description

Basic functionality for a *safe* iterator. The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 51 of file `safe_base.h`.

### 5.86.2 Constructor & Destructor Documentation

#### 5.86.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( )` [`inline`, `protected`]

Initializes the iterator and makes it singular.

Definition at line 77 of file `safe_base.h`.

#### 5.86.2.2 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( const _Safe_sequence_base * __seq, bool __constant )` [`inline`, `protected`]

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 88 of file `safe_base.h`.

References `_M_attach()`.

#### 5.86.2.3 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( const _Safe_iterator_base & __x, bool __constant )` [`inline`, `protected`]

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 95 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

### 5.86.3 Member Function Documentation

**5.86.3.1** `void __gnu_debug::_Safe_iterator_base::_M_attach (   
_Safe_sequence_base * __seq, bool __constant )`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_iterator_base()`.

**5.86.3.2** `void __gnu_debug::_Safe_iterator_base::_M_attach_single (   
_Safe_sequence_base * __seq, bool __constant ) throw ()`

Likewise, but not thread-safe.

**5.86.3.3** `bool __gnu_debug::_Safe_iterator_base::_M_attached_to ( const   
_Safe_sequence_base * __seq ) const [inline]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `_M_sequence`.

**5.86.3.4** `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_   
can_compare ( const _Safe_iterator_base & __x ) const throw   
()`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

**5.86.3.5** `void __gnu_debug::_Safe_iterator_base::_M_detach ( )`

Detach the iterator for whatever sequence it is attached to, if any.

**5.86.3.6** `void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw   
()`

Likewise, but not thread-safe.

### 5.86.3.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected]`

For use in [\\_Safe\\_iterator](#).

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

### 5.86.3.8 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ()`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular()`, `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

## 5.86.4 Member Data Documentation

### 5.86.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

### 5.86.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

### 5.86.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

## 5.87 `__gnu_debug::_Safe_sequence<_Sequence >` Class Template Reference 975

Referenced by `_M_attached_to()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_is_begin()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_is_end()`, `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::operator=()`.

### 5.86.4.4 unsigned int `__gnu_debug::_Safe_iterator_base::_M_version`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.87 `__gnu_debug::_Safe_sequence<_Sequence >` Class Template Reference

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

Inheritance diagram for `__gnu_debug::_Safe_sequence<_Sequence >`:



## Public Member Functions

- void [\\_M\\_invalidate\\_all](#) () const
- template<typename [\\_Predicate](#) >  
void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- template<typename [\\_Iterator](#) >  
void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#) > &\_\_x)

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

### 5.87.1 Detailed Description

```
template<typename \_Sequence> class \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it. The class template [\\_Safe\\_sequence](#) simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `types` that are instantiations of class template [\\_Safe\\_iterator](#) for this sequence. Iterators will then be tracked automatically.

Definition at line 97 of file `safe_sequence.h`.

## 5.87 \_\_gnu\_debug::\_Safe\_sequence< \_Sequence > Class Template Reference 977

### 5.87.2 Member Function Documentation

**5.87.2.1** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.87.2.2** `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

**5.87.2.3** `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ()` [protected, inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.87.2.4** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const`  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.87.2.5** `template<typename _Sequence > template<typename _Predicate > void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if ( _Predicate __pred )`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

Definition at line 121 of file `safe_sequence.h`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, and `__gnu_debug::_Safe_sequence_base::_M_iterators`.

**5.87.2.6** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )`  
**[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.87.2.7** `void __gnu_debug::Safe_sequence_base::M_swap (`  
`_Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.87.2.8** `template<typename _Sequence> template<typename _Iterator >`  
`void __gnu_debug::Safe_sequence<_Sequence >::M_transfer_iter`  
`( const _Safe_iterator<_Iterator, _Sequence > & __x )`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

Definition at line 154 of file `safe_sequence.h`.

References `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_get_mutex()`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_sequence`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence >::base()`.

### 5.87.3 Member Data Documentation

**5.87.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence >::M_transfer_iter()`.

**5.87.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 5.87.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

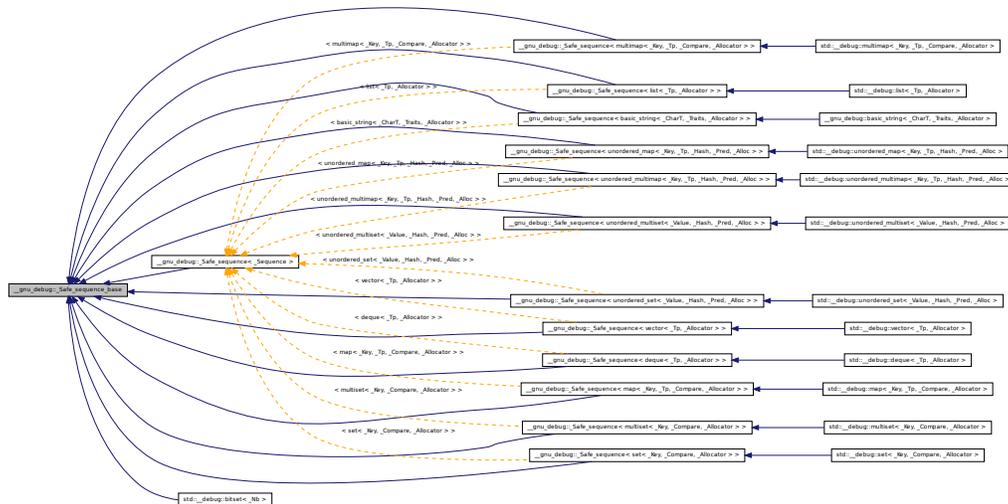
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.88 `__gnu_debug::Safe_sequence_base` Class Reference

Base class that supports tracking of iterators that reference a sequence.

Inheritance diagram for `__gnu_debug::Safe_sequence_base`:



## Public Member Functions

- void [\\_M\\_invalidate\\_all](#) () const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- [~\\_Safe\\_sequence\\_base](#) ()
- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

### 5.88.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence. The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 159 of file `safe_base.h`.

### 5.88.2 Constructor & Destructor Documentation

#### 5.88.2.1 `__gnu_debug::_Safe_sequence_base::~_Safe_sequence_base ( )` [inline, protected]

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 179 of file `safe_base.h`.

### 5.88.3 Member Function Documentation

**5.88.3.1** `void __gnu_debug::Safe_sequence_base::M_detach_all ( )`  
[protected]

Detach all iterators, leaving them singular.

**5.88.3.2** `void __gnu_debug::Safe_sequence_base::M_detach_singular ( )`  
[protected]

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.88.3.3** `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw ()` [protected]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.88.3.4** `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const`  
[inline]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.88.3.5** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )`  
[protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.88.3.6** `void __gnu_debug::Safe_sequence_base::M_swap (`  
`_Safe_sequence_base & __x )` [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.88.4 Member Data Documentation

### 5.88.4.1 `_Safe_iterator_base*` `__gnu_debug::Safe_sequence_base::M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 5.88.4.2 `_Safe_iterator_base*` `__gnu_debug::Safe_sequence_base::M_iterators`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

### 5.88.4.3 `unsigned int` `__gnu_debug::Safe_sequence_base::M_version` `[mutable]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference

Class `std::basic_string` with safety/checking/debug instrumentation.



## Public Member Functions

- **basic\_string** (const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const [\\_Base](#) &\_\_base)
- **basic\_string** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[\\_Base::npos](#), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator >  
**basic\_string** (\_InputIterator \_\_begin, \_InputIterator \_\_end, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** ([basic\\_string](#) &&\_\_str)
- **basic\_string** (const \_CharT \*\_\_s, size\_type \_\_n, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (std::initializer\_list< \_CharT > \_\_l, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const [basic\\_string](#) &\_\_str)
- **basic\_string** (const \_CharT \*\_\_s, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (size\_type \_\_n, \_CharT \_\_c, const \_Allocator &\_\_a=\_Allocator())
  
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< \_Iterator, [basic\\_string](#)< \_CharT, \_Traits, \_Allocator > > &\_\_x)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **append** (size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **append** (initializer\_list< \_CharT > \_\_l)
- [basic\\_string](#) & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **append** (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator >  
[basic\\_string](#) & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & **assign** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **assign** ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & **assign** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 985**

---

- `basic_string & assign` (const `_CharT * __s`, `size_type __n`)
- `basic_string & assign` (const `_CharT * __s`)
- `basic_string & assign` (`size_type __n`, `_CharT __c`)
- `basic_string & assign` (`_InputIterator __first`, `_InputIterator __last`)
- `basic_string & assign` (`initializer_list<_CharT> __l`)
- `basic_string & assign` (const `basic_string & __x`)
- `basic_string & assign` (`basic_string && __x`)
- `basic_string & assign` (const `basic_string & __str`, `size_type __pos`, `size_type __n`)
- `basic_string & assign` (const `_CharT * __s`, `size_type __n`)
- `basic_string & assign` (const `_CharT * __s`)
- `basic_string & assign` (`size_type __n`, `_CharT __c`)
- `template<typename _InputIterator>`  
`basic_string & assign` (`_InputIterator __first`, `_InputIterator __last`)
- `basic_string & assign` (`std::initializer_list<_CharT> __l`)
- `const_reference at` (`size_type __n`) `const`
- `reference at` (`size_type __n`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `iterator begin` ()
- `const_iterator begin` () `const`
- `const _CharT * c_str` () `const`
- `const _CharT * c_str` () `const`
- `size_type capacity` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `void clear` ()
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT * __s`, `size_type __n2`) `const`
- `int compare` (const `_CharT * __s`) `const`
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT * __s`) `const`
- `int compare` (const `basic_string & __str`) `const`
- `int compare` (`size_type __pos1`, `size_type __n1`, const `basic_string & __str`) `const`
- `int compare` (`size_type __pos1`, `size_type __n1`, const `basic_string & __str`, `size_type __pos2`, `size_type __n2`) `const`
- `int compare` (`size_type __pos1`, `size_type __n1`, const `_CharT * __s`) `const`
- `int compare` (const `basic_string & __str`) `const`
- `int compare` (const `_CharT * __s`) `const`
- `int compare` (`size_type __pos`, `size_type __n`, const `basic_string & __str`) `const`
- `int compare` (`size_type __pos1`, `size_type __n1`, const `_CharT * __s`, `size_type __n2`) `const`

- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- size\_type `copy` (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- size\_type `copy` (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- const \_CharT \* `data` () const
- const \_CharT \* `data` () const
- bool `empty` () const
- `iterator` `end` ()
- `const_iterator` `end` () const
- `iterator` `end` ()
- `const_iterator` `end` () const
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`npos`)
- `iterator` `erase` (`iterator` \_\_position)
- `iterator` `erase` (`iterator` \_\_first, `iterator` \_\_last)
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`Base::npos`)
- `iterator` `erase` (`iterator` \_\_position)
- `iterator` `erase` (`iterator` \_\_first, `iterator` \_\_last)
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find` (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find` (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find_first_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_of` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_of` (\_CharT \_\_c, size\_type \_\_pos=0) const

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 987**

---

- `size_type find_first_of` (const `basic_string` &\_\_str, `size_type` \_\_pos=0) const
- `size_type find_first_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_first_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=0) const
- `size_type find_first_of` (`_CharT` \_\_c, `size_type` \_\_pos=0) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_not_of` (const `basic_string` &\_\_str, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_not_of` (`_CharT` \_\_c, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`npos`) const
- `size_type find_last_not_of` (const `basic_string` &\_\_str, `size_type` \_\_pos=`npos`) const
- `size_type find_last_not_of` (`_CharT` \_\_c, `size_type` \_\_pos=`npos`) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_last_of` (const `basic_string` &\_\_str, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (`_CharT` \_\_c, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_of` (const `basic_string` &\_\_str, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`_Base::npos`) const
- `size_type find_last_of` (`_CharT` \_\_c, `size_type` \_\_pos=`npos`) const
- `allocator_type get_allocator` () const
- `basic_string` & `insert` (`size_type` \_\_pos1, const `basic_string` &\_\_str, `size_type` \_\_pos2, `size_type` \_\_n)
- `basic_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s, `size_type` \_\_n)
- `iterator insert` (`iterator` \_\_p, `_CharT` \_\_c)
- `basic_string` & `insert` (`size_type` \_\_pos1, const `basic_string` &\_\_str)
- `void insert` (`iterator` \_\_p, `InputIterator` \_\_beg, `InputIterator` \_\_end)
- `basic_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s, `size_type` \_\_n)
- `basic_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s)
- `basic_string` & `insert` (`size_type` \_\_pos, `size_type` \_\_n, `_CharT` \_\_c)
- `basic_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s)
- `void insert` (`iterator` \_\_p, `size_type` \_\_n, `_CharT` \_\_c)
- `basic_string` & `insert` (`size_type` \_\_pos1, const `basic_string` &\_\_str, `size_type` \_\_pos2, `size_type` \_\_n)

- [iterator insert](#) (iterator \_\_p, \_CharT \_\_c)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string & insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- void [insert](#) (iterator \_\_p, initializer\_list< \_CharT > \_\_l)
- void **insert** (iterator \_\_p, [std::initializer\\_list](#)< \_CharT > \_\_l)
- template<typename \_InputIterator >  
void **insert** (iterator \_\_p, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string & insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- size\_type [length](#) () const
- size\_type [max\\_size](#) () const
- [basic\\_string & operator+=](#) (\_CharT \_\_c)
- [basic\\_string & operator+=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string & operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string & operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string & operator+=](#) (initializer\_list< \_CharT > \_\_l)
- [basic\\_string & operator+=](#) (\_CharT \_\_c)
- [basic\\_string & operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string & operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string & operator=](#) ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string & operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string & operator=](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string & operator=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string & operator=](#) (\_CharT \_\_c)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const
- reference [operator\[\]](#) (size\_type \_\_pos)
- const\_reference **operator[]** (size\_type \_\_pos) const
- reference **operator[]** (size\_type \_\_pos)
- void **push\_back** (\_CharT \_\_c)
- void [push\\_back](#) (\_CharT \_\_c)
- [reverse\\_iterator rbegin](#) ()
- [const\\_reverse\\_iterator rbegin](#) () const
- [reverse\\_iterator rbegin](#) ()
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const
- [reverse\\_iterator rend](#) ()
- [reverse\\_iterator rend](#) ()
- [const\\_reverse\\_iterator rend](#) () const
- [basic\\_string & replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string & replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str)
- [basic\\_string & replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 989**

---

- `template<typename _InputIterator >`  
`basic_string & replace (iterator __i1, iterator __i2, _InputIterator __j1, _InputIterator __j2)`
- `basic_string & replace (iterator __i1, iterator __i2, std::initializer_list<_CharT > __l)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)`
- `basic_string & replace (iterator __i1, iterator __i2, iterator __k1, iterator __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, _CharT * __k1, _CharT * __k2)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __k1, const _CharT * __k2)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (iterator __i1, iterator __i2, initializer_list<_CharT > __l)`
- `basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s)`
- `basic_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT * __s)`
- `basic_string & replace (size_type __pos1, size_type __n1, const basic_string & __str, size_type __pos2, size_type __n2)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n, _CharT __c)`
- `void resize (size_type __n)`
- `void resize (size_type __n)`
- `void resize (size_type __n, _CharT __c)`
- `size_type rfind (const _CharT * __s, size_type __pos=__Base::npos) const`

- `size_type rfind (_CharT __c, size_type __pos=npos) const`
- `size_type rfind (const _CharT *__s, size_type __pos=npos) const`
- `size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type rfind (const basic_string &__str, size_type __pos=_Base::npos) const`
- `size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type rfind (const basic_string &__str, size_type __pos=npos) const`
- `size_type rfind (_CharT __c, size_type __pos=_Base::npos) const`
- `void shrink_to_fit ()`
- `size_type size () const`
- `basic_string substr (size_type __pos=0, size_type __n=_Base::npos) const`
- `basic_string substr (size_type __pos=0, size_type __n=npos) const`
- `void swap (basic_string &__s)`
- `void swap (basic_string< _CharT, _Traits, _Allocator > &__x)`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Static Public Attributes

- `static const size_type npos`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

### 5.89.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>,
typename _Allocator = std::allocator<_CharT>> class __gnu_debug::basic_
string< _CharT, _Traits, _Allocator >
```

Class `std::basic_string` with safety/checking/debug instrumentation.

Definition at line 42 of file `debug/string`.

## 5.89.2 Member Function Documentation

**5.89.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.89.2.2** `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.89.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` [protected, inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

**5.89.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( ) const`  
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.89.2.5** `void __gnu_debug::Safe_sequence<basic_string<_CharT, _Traits, _Allocator>>::_M_invalidate_if ( _Predicate __pred )`  
[inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.89.2.6** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( )`  
**[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.89.2.7** `void __gnu_debug::Safe_sequence_base::_M_swap (`  
`_Safe_sequence_base & __x )` **[protected, inherited]**

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.89.2.8** `void __gnu_debug::Safe_sequence< basic_string< _CharT,`  
`_Traits, _Allocator > >::_M_transfer_iter ( const_Safe_iterator<`  
`_Iterator, basic_string< _CharT, _Traits, _Allocator > > & __x )`  
**[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

**5.89.2.9** `basic_string& std::basic_string< _CharT, _Traits, _Allocator`  
`>::append ( const basic_string< _CharT, _Traits, _Allocator > &`  
`__str )` **[inherited]**

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

**5.89.2.10** `basic_string& std::basic_string< _CharT, _Traits, _Allocator`  
`>::append ( const basic_string< _CharT, _Traits, _Allocator > &`  
`__str, size_type __pos, size_type __n )` **[inherited]**

Append a substring.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 993

---

### Parameters

- str* The string to append.
- pos* Index of the first character of *str* to append.
- n* The number of characters to append.

### Returns

Reference to this string.

### Exceptions

- [\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

#### 5.89.2.11 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( const _CharT* __s, size_type __n )` [inherited]

Append a C substring.

### Parameters

- s* The C string to append.
- n* The number of characters to append.

### Returns

Reference to this string.

#### 5.89.2.12 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( const _CharT* __s )` [inline, inherited]

Append a C string.

### Parameters

- s* The C string to append.

### Returns

Reference to this string.

Definition at line 954 of file `basic_string.h`.

**5.89.2.13** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( size_type __n, _CharT __c ) [inherited]`

Append multiple characters.

**Parameters**

*n* The number of characters to append.

*c* The character to use.

**Returns**

Reference to this string.

Appends *n* copies of *c* to this string.

**5.89.2.14** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( initializer_list<_CharT> __l ) [inline, inherited]`

Append an `initializer_list` of characters.

**Parameters**

*l* The `initializer_list` of characters to append.

**Returns**

Reference to this string.

Definition at line 978 of file `basic_string.h`.

**5.89.2.15** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( _InputIterator __first, _InputIterator __last ) [inline, inherited]`

Append a range of characters.

**Parameters**

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

**Returns**

Reference to this string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 995

---

Appends characters in the range [first,last) to this string.

Definition at line 992 of file `basic_string.h`.

**5.89.2.16** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const basic_string<_CharT, _Traits, _Allocator> & __str ) [inherited]`

Set value to contents of another string.

### Parameters

*str* Source string to use.

### Returns

Reference to this string.

**5.89.2.17** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( basic_string<_CharT, _Traits, _Allocator> && __str ) [inline, inherited]`

Set value to contents of another string.

### Parameters

*str* Source string to use.

### Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1027 of file `basic_string.h`.

**5.89.2.18** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n ) [inline, inherited]`

Set value to a substring of a string.

### Parameters

*str* The string to use.

*pos* Index of the first character of *str*.

*n* Number of characters to use.

### Returns

Reference to this string.

### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1047 of file `basic_string.h`.

#### 5.89.2.19 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const _CharT* __s, size_type __n ) [inherited]`

Set value to a C substring.

### Parameters

*s* The C string to use.

*n* Number of characters to use.

### Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

#### 5.89.2.20 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const _CharT* __s ) [inline, inherited]`

Set value to contents of a C string.

### Parameters

*s* The C string to use.

### Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1075 of file `basic_string.h`.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 997

---

**5.89.2.21** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( size_type __n, _CharT __c ) [inline, inherited]`

Set value to multiple characters.

### Parameters

*n* Length of the resulting string.

*c* The character to use.

### Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1091 of file `basic_string.h`.

**5.89.2.22** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( _InputIterator __first, _InputIterator __last ) [inline, inherited]`

Set value to a range of characters.

### Parameters

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

### Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1104 of file `basic_string.h`.

**5.89.2.23** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( initializer_list<_CharT> __l ) [inline, inherited]`

Set value to an `initializer_list` of characters.

### Parameters

*l* The `initializer_list` of characters to assign.

**Returns**

Reference to this string.

Definition at line 1114 of file basic\_string.h.

**5.89.2.24** `const_reference std::basic_string<_CharT, _Traits, _Allocator >::at ( size_type __n ) const [inline, inherited]`

Provides access to the data contained in the string.

**Parameters**

*n* The index of the character to access.

**Returns**

Read-only (const) reference to the character.

**Exceptions**

[\*std::out\\_of\\_range\*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 847 of file basic\_string.h.

**5.89.2.25** `reference std::basic_string<_CharT, _Traits, _Allocator >::at ( size_type __n ) [inline, inherited]`

Provides access to the data contained in the string.

**Parameters**

*n* The index of the character to access.

**Returns**

Read/write reference to the character.

**Exceptions**

[\*std::out\\_of\\_range\*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 866 of file basic\_string.h.

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference** **999**

---

**5.89.2.26** `iterator std::basic_string<_CharT, _Traits, _Allocator>::begin ( )`  
`[inline, inherited]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 591 of file `basic_string.h`.

**5.89.2.27** `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::begin ( ) const`  
`[inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 602 of file `basic_string.h`.

**5.89.2.28** `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::c_str ( ) const`  
`[inline, inherited]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1723 of file `basic_string.h`.

**5.89.2.29** `size_type std::basic_string<_CharT, _Traits, _Allocator>::capacity ( ) const`  
`[inline, inherited]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 759 of file `basic_string.h`.

**5.89.2.30** `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cbegin ( ) const`  
`[inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 666 of file `basic_string.h`.

**5.89.2.31** `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cend ( ) const`  
`[inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 674 of file `basic_string.h`.

**5.89.2.32** `void std::basic_string<_CharT, _Traits, _Allocator >::clear ( )`  
`[inline, inherited]`

Erases the string, making it empty.

Definition at line 786 of file basic\_string.h.

**5.89.2.33** `int std::basic_string<_CharT, _Traits, _Allocator >::compare (`  
`size_type __pos, size_type __nl, const _CharT * __s ) const`  
`[inherited]`

Compare substring to a C string.

#### Parameters

*pos* Index of first character of substring.

*nl* Number of characters in substring.

*s* C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *nl* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.89.2.34** `int std::basic_string<_CharT, _Traits, _Allocator >::compare (`  
`size_type __pos, size_type __n, const basic_string<_CharT,`  
`_Traits, _Allocator > & __str ) const [inherited]`

Compare substring to a string.

#### Parameters

*pos* Index of first character of substring.

*n* Number of characters in substring.

*str* String to compare against.

#### Returns

Integer < 0, 0, or > 0.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1001

---

Form the substring of this string from the  $n$  characters starting at  $pos$ . Returns an integer  $< 0$  if the substring is ordered before  $str$ ,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after  $str$ . Determines the effective length  $rlen$  of the strings to compare as the smallest of the length of the substring and  $str.size()$ . The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.89.2.35** `int std::basic_string<_CharT, _Traits, _Allocator>::compare ( size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n2 ) const [inherited]`

Compare substring to a substring.

### Parameters

*pos1* Index of first character of substring.

*n1* Number of characters in substring.

*str* String to compare against.

*pos2* Index of first character of substring of *str*.

*n2* Number of characters in substring of *str*.

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the  $n1$  characters starting at  $pos1$ . Form the substring of  $str$  from the  $n2$  characters starting at  $pos2$ . Returns an integer  $< 0$  if this substring is ordered before the substring of  $str$ ,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of  $str$ . Determines the effective length  $rlen$  of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.89.2.36** `int std::basic_string<_CharT, _Traits, _Allocator>::compare ( const _CharT* __s ) const [inherited]`

Compare to a C string.

### Parameters

*s* C string to compare against.

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.89.2.37** `int std::basic_string<_CharT,_Traits,_Allocator>::compare ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) const` [**inherited**]

Compare substring against a character array.

**Parameters**

*pos1* Index of first character of substring.

*n1* Number of characters in substring.

*s* character array to compare against.

*n2* Number of characters of *s*.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

**5.89.2.38** `int std::basic_string<_CharT,_Traits,_Allocator>::compare ( const basic_string<_CharT,_Traits,_Allocator> & __str ) const` [**inline, inherited**]

Compare to a string.

**Parameters**

*str* String to compare against.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1003

---

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before *str*,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2130 of file `basic_string.h`.

**5.89.2.39** `size_type std::basic_string<_CharT, _Traits, _Allocator>::copy`  
( `_CharT* __s`, `size_type __n`, `size_type __pos = 0` ) `const`  
[`inherited`]

Copy substring into C string.

### Parameters

*s* C string to copy value into.

*n* Number of characters to copy.

*pos* Index of first character to copy.

### Returns

Number of characters actually copied

### Exceptions

[\*std::out\\_of\\_range\*](#) If `pos > size()`.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than `size()`, `out_of_range` is thrown.

**5.89.2.40** `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::crbegin` ( ) `const` [`inline`, `inherited`]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 683 of file `basic_string.h`.

**5.89.2.41** `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::crend( ) const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 692 of file basic\_string.h.

**5.89.2.42** `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::data( ) const [inline, inherited]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1733 of file basic\_string.h.

**5.89.2.43** `bool std::basic_string<_CharT, _Traits, _Allocator>::empty( ) const [inline, inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 794 of file basic\_string.h.

**5.89.2.44** `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::end( ) const [inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 621 of file basic\_string.h.

**5.89.2.45** `iterator std::basic_string<_CharT, _Traits, _Allocator>::end( ) [inline, inherited]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 610 of file basic\_string.h.

**5.89.2.46** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::erase( size_type __pos = 0, size_type __n = npos ) [inline, inherited]`

Remove characters.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1005

---

### Parameters

- pos* Index of first character to remove (default 0).
- n* Number of characters to remove (default remainder).

### Returns

Reference to this string.

### Exceptions

*std::out\_of\_range* If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are  $< n$  characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1304 of file `basic_string.h`.

### 5.89.2.47 iterator `std::basic_string<_CharT, _Traits, _Allocator>::erase ( iterator __position ) [inline, inherited]`

Remove one character.

### Parameters

- position* Iterator referencing the character to remove.

### Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1320 of file `basic_string.h`.

### 5.89.2.48 iterator `std::basic_string<_CharT, _Traits, _Allocator>::erase ( iterator __first, iterator __last ) [inherited]`

Remove a range of characters.

### Parameters

- first* Iterator referencing the first character to remove.
- last* Iterator referencing the end of the range.

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**5.89.2.49** `size_type std::basic_string<_CharT, _Traits, _Allocator >::find ( const _CharT * __s, size_type __pos, size_type __n ) const [inherited]`

Find position of a C substring.

**Parameters**

*s* C string to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

**Returns**

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns npos.

**5.89.2.50** `size_type std::basic_string<_CharT, _Traits, _Allocator >::find ( const basic_string<_CharT, _Traits, _Allocator > & __str, size_type __pos = 0 ) const [inline, inherited]`

Find position of a string.

**Parameters**

*str* String to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1768 of file basic\_string.h.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1007

---

**5.89.2.51** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find ( const _CharT * __s, size_type __pos = 0 ) const [inline, inherited]`

Find position of a C string.

### Parameters

- s* C string to locate.
- pos* Index of character to search from (default 0).

### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1782 of file `basic_string.h`.

**5.89.2.52** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find ( _CharT __c, size_type __pos = 0 ) const [inherited]`

Find position of a character.

### Parameters

- c* Character to locate.
- pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

**5.89.2.53** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0 ) const [inline, inherited]`

Find position of a character not in string.

### Parameters

- str* String containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1992 of file `basic_string.h`.

**5.89.2.54** `size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of( const _CharT * __s, size_type __pos, size_type __n ) const` `[inherited]`

Find position of a character not in C substring.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from.

*n* Number of characters from *s* to consider.

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.55** `size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of( _CharT __c, size_type __pos = 0 ) const` `[inherited]`

Find position of a different character.

### Parameters

*c* Character to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1009

---

**5.89.2.56** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of( const _CharT * __s, size_type __pos = 0 ) const` `[inline, inherited]`

Find position of a character not in C string.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2021 of file `basic_string.h`.

**5.89.2.57** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of( const _CharT * __s, size_type __pos = 0 ) const` `[inline, inherited]`

Find position of a character of C string.

### Parameters

*s* String containing characters to locate.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1898 of file `basic_string.h`.

**5.89.2.58** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0 ) const` `[inline, inherited]`

Find position of a character of string.

**Parameters**

*str* String containing characters to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1870 of file `basic_string.h`.

**5.89.2.59** `size_type std::basic_string< _CharT, _Traits, _Allocator  
>::find_first_of( const _CharT * __s, size_type __pos, size_type  
__n ) const [inherited]`

Find position of a character of C substring.

**Parameters**

*s* String containing characters to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.60** `size_type std::basic_string< _CharT, _Traits, _Allocator  
>::find_first_of( _CharT __c, size_type __pos = 0 ) const  
[inline, inherited]`

Find position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1011

---

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1917 of file `basic_string.h`.

**5.89.2.61** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of ( _CharT __c, size_type __pos = npos ) const`  
`[inherited]`

Find last position of a different character.

### Parameters

*c* Character to avoid.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.62** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of ( const _CharT * __s, size_type __pos = npos ) const`  
`[inline, inherited]`

Find last position of a character not in C string.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2080 of file `basic_string.h`.

**5.89.2.63** `size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of ( const basic_string< _CharT, _Traits, _Allocator > & __str, size_type __pos = npos ) const [inline, inherited]`

Find last position of a character not in string.

#### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2051 of file `basic_string.h`.

**5.89.2.64** `size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of ( const _CharT * __s, size_type __pos, size_type __n ) const [inherited]`

Find last position of a character not in C substring.

#### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from.

*n* Number of characters from *s* to consider.

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.65** `size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of ( const _CharT * __s, size_type __pos = npos ) const [inline, inherited]`

Find last position of a character of C string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1013

---

### Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1959 of file `basic_string.h`.

**5.89.2.66** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npow ) const [inline, inherited]`

Find last position of a character of string.

### Parameters

- str* String containing characters to locate.
- pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1931 of file `basic_string.h`.

**5.89.2.67** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of ( _CharT __c, size_type __pos = npow ) const [inline, inherited]`

Find last position of a character.

### Parameters

- c* Character to locate.
- pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1978 of file `basic_string.h`.

**5.89.2.68** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of ( const _CharT * __s, size_type __pos, size_type __n ) const [inherited]`

Find last position of a character of C substring.

#### Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.69** `allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator ( ) const [inline, inherited]`

Return copy of allocator used to construct this string.

Definition at line 1740 of file `basic_string.h`.

**5.89.2.70** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos, const _CharT * __s ) [inline, inherited]`

Insert a C string.

#### Parameters

- pos* Iterator referencing location in string to insert at.
- s* The C string to insert.

#### Returns

Reference to this string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1015

---

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1240 of file `basic_string.h`.

```
5.89.2.71 void std::basic_string<_CharT, _Traits, _Allocator>::insert  
( iterator __p, size_type __n, _CharT __c ) [inline,  
inherited]
```

Insert multiple characters.

### Parameters

*p* Iterator referencing location in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1131 of file `basic_string.h`.

```
5.89.2.72 void std::basic_string<_CharT, _Traits, _Allocator>::insert  
( iterator __p, InputIterator __beg, InputIterator __end )  
[inline, inherited]
```

Insert a range of characters.

### Parameters

*p* Iterator referencing location in string to insert at.

*beg* Start of range.

*end* End of range.

**Exceptions**

*std::length\_error* If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1147 of file `basic_string.h`.

```
5.89.2.73 void std::basic_string<_CharT, _Traits, _Allocator >::insert
( iterator __p, initializer_list<_CharT > __l ) [inline,
inherited]
```

Insert an `initializer_list` of characters.

**Parameters**

*p* Iterator referencing location in string to insert at.

*l* The `initializer_list` of characters to insert.

**Exceptions**

*std::length\_error* If new length exceeds `max_size()`.

Definition at line 1158 of file `basic_string.h`.

```
5.89.2.74 basic_string& std::basic_string<_CharT, _Traits, _Allocator
>::insert ( size_type __pos1, const basic_string<_CharT,
_Traits, _Allocator > & __str, size_type __pos2, size_type __n )
[inline, inherited]
```

Insert a substring.

**Parameters**

*pos1* Iterator referencing location in string to insert at.

*str* The string to insert.

*pos2* Start of characters in `str` to insert.

*n* Number of characters to insert.

**Returns**

Reference to this string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1017

---

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If `pos1 > size()` or `pos2 > str.size()`.

Starting at `pos1`, insert `n` character of `str` beginning with `pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `pos1` is beyond the end of this string or `pos2` is beyond the end of `str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1199 of file `basic_string.h`.

### 5.89.2.75 `iterator std::basic_string<_CharT, _Traits, _Allocator>::insert ( iterator __p, _CharT __c ) [inline, inherited]`

Insert one character.

#### Parameters

*p* Iterator referencing position in string to insert at.

*c* The character to insert.

#### Returns

Iterator referencing newly inserted char.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts character `c` at position referenced by `p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1280 of file `basic_string.h`.

### 5.89.2.76 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos, const _CharT* __s, size_type __n ) [inherited]`

Insert a C substring.

#### Parameters

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

*n* The number of characters to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**5.89.2.77** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos, size_type __n, _CharT __c )`  
**[inline, inherited]**

Insert multiple characters.

### Parameters

*pos* Index in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* > `length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1263 of file `basic_string.h`.

**5.89.2.78** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str )`  
**[inline, inherited]**

Insert value of a string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1019

---

### Parameters

*posI* Iterator referencing location in string to insert at.  
*str* The string to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts value of *str* starting at *posI*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1177 of file `basic_string.h`.

**5.89.2.79** `size_type std::basic_string<_CharT, _Traits, _Allocator>::length ( ) const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 707 of file `basic_string.h`.

**5.89.2.80** `size_type std::basic_string<_CharT, _Traits, _Allocator>::max_size ( ) const [inline, inherited]`

Returns the `size()` of the largest possible string.

Definition at line 712 of file `basic_string.h`.

**5.89.2.81** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( initializer_list<_CharT> __l ) [inline, inherited]`

Append an `initializer_list` of characters.

### Parameters

*l* The `initializer_list` of characters to be appended.

### Returns

Reference to this string.

Definition at line 912 of file `basic_string.h`.

**5.89.2.82** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Append a string to this string.

**Parameters**

*str* The string to append.

**Returns**

Reference to this string.

Definition at line 881 of file basic\_string.h.

**5.89.2.83** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( const _CharT * __s ) [inline, inherited]`

Append a C string.

**Parameters**

*s* The C string to append.

**Returns**

Reference to this string.

Definition at line 890 of file basic\_string.h.

**5.89.2.84** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( _CharT __c ) [inline, inherited]`

Append a character.

**Parameters**

*c* The character to append.

**Returns**

Reference to this string.

Definition at line 899 of file basic\_string.h.

---

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator >` Class Template Reference 1021

---

### 5.89.2.85 `reference std::basic_string<_CharT, _Traits, _Allocator >::operator[] ( size_type __pos ) [inline, inherited]`

Subscript access to the data contained in the string.

#### Parameters

*pos* The index of the character to access.

#### Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 826 of file `basic_string.h`.

### 5.89.2.86 `const_reference std::basic_string<_CharT, _Traits, _Allocator >::operator[] ( size_type __pos ) const [inline, inherited]`

Subscript access to the data contained in the string.

#### Parameters

*pos* The index of the character to access.

#### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 809 of file `basic_string.h`.

### 5.89.2.87 `void std::basic_string<_CharT, _Traits, _Allocator >::push_back ( _CharT __c ) [inline, inherited]`

Append a single character.

#### Parameters

*c* Character to append.

Definition at line 1000 of file `basic_string.h`.

**5.89.2.88** `reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rbegin( )` [`inline`, `inherited`]

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 630 of file `basic_string.h`.

**5.89.2.89** `const_reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rbegin( ) const` [`inline`, `inherited`]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 639 of file `basic_string.h`.

**5.89.2.90** `const_reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rend( ) const` [`inline`, `inherited`]

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 657 of file `basic_string.h`.

**5.89.2.91** `reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rend( )` [`inline`, `inherited`]

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 648 of file `basic_string.h`.

**5.89.2.92** `basic_string&` `std::basic_string<_CharT, _Traits, _Allocator>::replace( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 )` [`inherited`]

Replace characters with value of a C substring.

#### Parameters

*pos* Index of first character to replace.

*n1* Number of characters to be replaced.

*s* C string to insert.

*n2* Number of characters from *s* to use.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1023

---

### Returns

Reference to this string.

### Exceptions

*std::out\_of\_range* If `pos1 > size()`.

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `n2` characters of `s` are inserted, or all of `s` if `n2` is too large. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**5.89.2.93** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, InputIterator __k1, InputIterator __k2 ) [inline, inherited]`

Replace range of characters with range.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*k1* Iterator referencing start of range to insert.

*k2* Iterator referencing end of range to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, characters in the range `[k1, k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1548 of file `basic_string.h`.

**5.89.2.94** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n2 ) [inline, inherited]`

Replace characters with value from another string.

**Parameters**

*pos1* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*str* String to insert.  
*pos2* Index of first character of str to use.  
*n2* Number of characters from str to use.

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If *pos1* > size() or *pos2* > str.size().  
*std::length\_error* If new length exceeds max\_size().

Removes the characters in the range [*pos1*,*pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, out\_of\_range is thrown. If the length of the result exceeds max\_size(), length\_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1381 of file basic\_string.h.

**5.89.2.95** `basic_string& std::basic_string<_CharT, _Traits, _Allocator  
>::replace ( size_type __pos, size_type __n1, const _CharT * __s )  
[inline, inherited]`

Replace characters with value of a C string.

**Parameters**

*pos* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*s* C string to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If *pos* > size().  
*std::length\_error* If new length exceeds max\_size().

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1025

---

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file `basic_string.h`.

**5.89.2.96** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, initializer_list<_CharT> __l ) [inline, inherited]`

Replace range of characters with `initializer_list`.

### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- l* The `initializer_list` of characters to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, characters in the range `[k1, k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1616 of file `basic_string.h`.

**5.89.2.97** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline, inherited]`

Replace range of characters with multiple characters.

### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- n* Number of characters to insert.
- c* Character to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1525 of file `basic_string.h`.

**5.89.2.98** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Replace characters with value from another string.

**Parameters**

*pos* Index of first character to replace.

*n* Number of characters to be replaced.

*str* String to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If *pos* is beyond the end of this string.

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1359 of file `basic_string.h`.

**5.89.2.99** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Replace range of characters with string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1027

---

### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- str* String value to insert.

### Returns

Reference to this string.

### Exceptions

***std::length\_error*** If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1465 of file `basic_string.h`.

### 5.89.2.100 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, const _CharT * __s, size_type __n )` [`inline`, `inherited`]

Replace range of characters with C substring.

### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.
- n* Number of characters from *s* to insert.

### Returns

Reference to this string.

### Exceptions

***std::length\_error*** If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1483 of file `basic_string.h`.

**5.89.2.101** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, const _CharT * __s )`  
[inline, inherited]

Replace range of characters with C string.

#### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*s* C string value to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1504 of file `basic_string.h`.

**5.89.2.102** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c )`  
[inline, inherited]

Replace characters with multiple characters.

#### Parameters

*pos* Index of first character to replace.

*n1* Number of characters to be replaced.

*n2* Number of characters to insert.

*c* Character to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::out\_of\_range* If *pos* > `size()`.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1029

---

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `n2` copies of `c` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1447 of file `basic_string.h`.

**5.89.2.103** `void std::basic_string<_CharT, _Traits, _Allocator>::reserve ( size_type __res_arg = 0 )` [**inherited**]

Attempt to preallocate enough memory for specified number of characters.

### Parameters

*res\_arg* Number of characters required.

### Exceptions

*std::length\_error* If *res\_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**5.89.2.104** `void std::basic_string<_CharT, _Traits, _Allocator>::resize ( size_type __n )` [**inline, inherited**]

Resizes the string to the specified number of characters.

### Parameters

*n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 739 of file `basic_string.h`.

**5.89.2.105** `void std::basic_string<_CharT, _Traits, _Allocator >::resize ( size_type __n, _CharT __c ) [inherited]`

Resizes the string to the specified number of characters.

#### Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

**5.89.2.106** `size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind ( const basic_string<_CharT, _Traits, _Allocator > & __str, size_type __pos = npos ) const [inline, inherited]`

Find last position of a string.

#### Parameters

- str* String to locate.
- pos* Index of character to search back from (default end).

#### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1812 of file `basic_string.h`.

**5.89.2.107** `size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind ( const _CharT * __s, size_type __pos = npos ) const [inline, inherited]`

Find last position of a C string.

#### Parameters

- s* C string to locate.
- pos* Index of character to start search at (default end).

**5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1031**

---

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1840 of file `basic_string.h`.

**5.89.2.108** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const [inherited]`

Find last position of a C substring.

**Parameters**

*s* C string to locate.

*pos* Index of character to search back from.

*n* Number of characters from *s* to search for.

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

**5.89.2.109** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind ( _CharT __c, size_type __pos = npos ) const [inherited]`

Find last position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.89.2.110** `void std::basic_string<_CharT, _Traits, _Allocator >::shrink_to_fit ( ) [inline, inherited]`

A non-binding request to reduce capacity() to size().

Definition at line 745 of file basic\_string.h.

**5.89.2.111** `size_type std::basic_string<_CharT, _Traits, _Allocator >::size ( ) const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 701 of file basic\_string.h.

**5.89.2.112** `basic_string std::basic_string<_CharT, _Traits, _Allocator >::substr ( size_type __pos = 0, size_type __n = npos ) const [inline, inherited]`

Get a substring.

#### Parameters

*pos* Index of first character (default 0).

*n* Number of characters in substring (default remainder).

#### Returns

The new string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* > size().

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, *out\_of\_range* is thrown.

Definition at line 2112 of file basic\_string.h.

**5.89.2.113** `void std::basic_string<_CharT, _Traits, _Allocator >::swap ( basic_string<_CharT, _Traits, _Allocator > & __s ) [inherited]`

Swap contents with another string.

## 5.89 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1033

---

### Parameters

*s* String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

### 5.89.3 Member Data Documentation

#### 5.89.3.1 `__Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

#### 5.89.3.2 `__Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

#### 5.89.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

#### 5.89.3.4 `const size_type std::basic_string<_CharT, _Traits, _Allocator>::npos` [static, inherited]

Value returned by various member functions when they fail.

Definition at line 271 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

## 5.90 `__gnu_parallel::__accumulate_binop_reduct< _BinOp >` Struct Template Reference

General reduction, using a binary operator.

### Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp &__b`)
- `template<typename _Result, typename _Addend > _Result operator()` (`const _Result &__x, const _Addend &__y`)

### Public Attributes

- `_BinOp & __binop`

#### 5.90.1 Detailed Description

`template<typename _BinOp> struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >`

General reduction, using a binary operator.

Definition at line 335 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

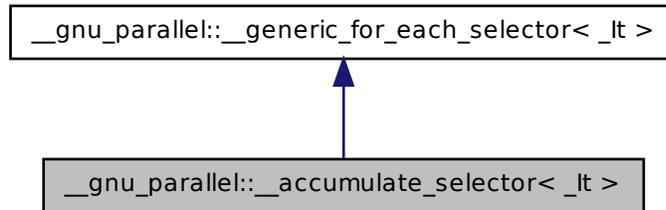
- [for\\_each\\_selectors.h](#)

## 5.91 `__gnu_parallel::__accumulate_selector< _It >` Struct Template Reference

[std::accumulate\(\)](#) selector.

## 5.91 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference 1035

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



### Public Member Functions

- `template<typename _Op > std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.91.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector<_It >`

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

### 5.91.2 Member Function Documentation

**5.91.2.1** `template<typename _It> template<typename _Op > std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It >::operator() ( _Op __o, _It __i ) [inline]`

Functor execution.

**Parameters**

`__o` Operator (unused).

`__i` iterator referencing object.

**Returns**

The current value.

Definition at line 216 of file `for_each_selectors.h`.

### 5.91.3 Member Data Documentation

#### 5.91.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

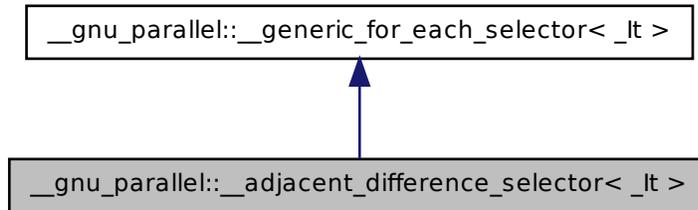
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.92 `__gnu_parallel::__adjacent_difference_selector< _It >` Struct Template Reference

Selector that returns the difference between two adjacent `__elements`.

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



## Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It \_M\_finish\_iterator`

### 5.92.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__adjacent_difference_selector<_It>`

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

### 5.92.2 Member Data Documentation

**5.92.2.1** `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It >::_M_finish_iterator`  
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

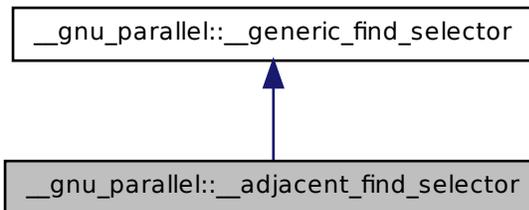
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.93 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Test predicate on two adjacent elements.

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 5.93.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

### 5.93.2 Member Function Documentation

5.93.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred ) [inline]`

Corresponding sequential algorithm on a sequence.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__pred` Find predicate.

Definition at line 105 of file `find_selectors.h`.

5.93.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

#### Parameters

- `__i1` Iterator on first sequence.
- `__i2` Iterator on second sequence (unused).
- `__pred` Find predicate.

Definition at line 90 of file `find_selectors.h`.

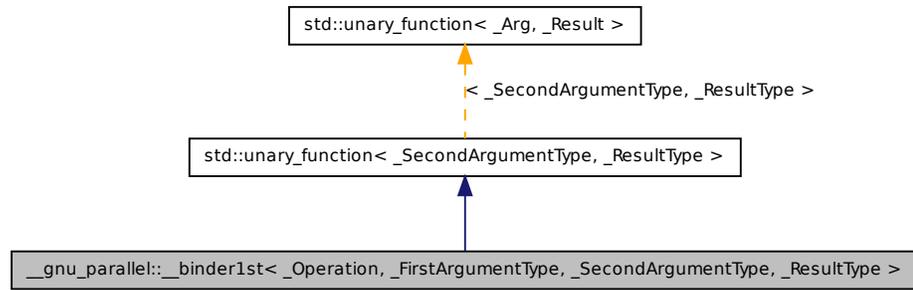
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

### 5.94 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to `std::binder1st`, but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



## Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

## Public Member Functions

- `__binder1st` (`const _Operation &__x, const _FirstArgumentType &__y`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`
- `_ResultType operator()` (`const _SecondArgumentType &__x`)

## Protected Attributes

- `_Operation` `_M_op`
- `_FirstArgumentType` `_M_value`

### 5.94.1 Detailed Description

```

template<typename _Operation, typename _FirstArgumentType, typename
_SecondArgumentType, typename _ResultType> class __gnu_parallel::__-
binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _-
ResultType >
  
```

Similar to `std::binder1st`, but giving the argument types explicitly.

**5.95 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference** **1041**

---

Definition at line 192 of file `parallel/base.h`.

## 5.94.2 Member Typedef Documentation

**5.94.2.1 `typedef _SecondArgumentType std::unary_function< _SecondArgumentType, _ResultType >::argument_type` [inherited]**

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.94.2.2 `typedef _ResultType std::unary_function< _SecondArgumentType, _ResultType >::result_type` [inherited]**

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

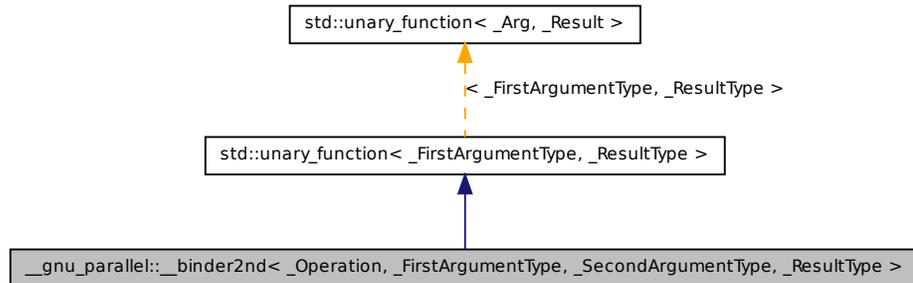
**5.95 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference**

Similar to `std::binder2nd`, but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _`

---

FirstArgumentType, \_SecondArgumentType, \_ResultType >:



## Public Types

- typedef `_FirstArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

## Public Member Functions

- `__binder2nd` (const `_Operation` &`__x`, const `_SecondArgumentType` &`__y`)
- `_ResultType` `operator()` (`_FirstArgumentType` &`__x`)
- `_ResultType` `operator()` (const `_FirstArgumentType` &`__x`) const

## Protected Attributes

- `_Operation` `_M_op`
- `_SecondArgumentType` `_M_value`

### 5.95.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename
_SecondArgumentType, typename _ResultType> class __gnu_parallel::__-
binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _-
ResultType >
```

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

## 5.95.2 Member Typedef Documentation

5.95.2.1 `typedef _FirstArgumentType std::unary_function<_FirstArgumentType, _ResultType>::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.95.2.2 `typedef _ResultType std::unary_function<_FirstArgumentType, _ResultType>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

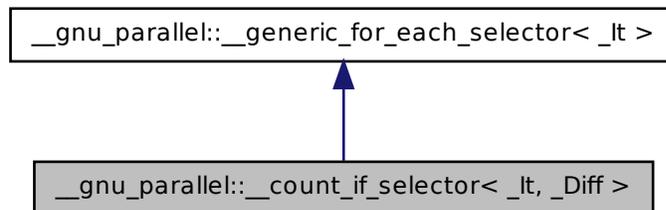
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.96 `__gnu_parallel::__count_if_selector<_It, _Diff>` Struct Template Reference

`std::count_if()` selector.

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff>`:



## Public Member Functions

- `template<typename _Op > _Diff operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.96.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector< _It, _Diff >`

`std::count_if ()` selector.

Definition at line 194 of file `for_each_selectors.h`.

### 5.96.2 Member Function Documentation

**5.96.2.1** `template<typename _It, typename _Diff > template<typename _Op > _Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() ( _Op & __o, _It __i ) [inline]`

Functor execution.

#### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

#### Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

### 5.96.3 Member Data Documentation

**5.96.3.1** `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

## 5.97 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference 1045

Definition at line 47 of file `for_each_selectors.h`.

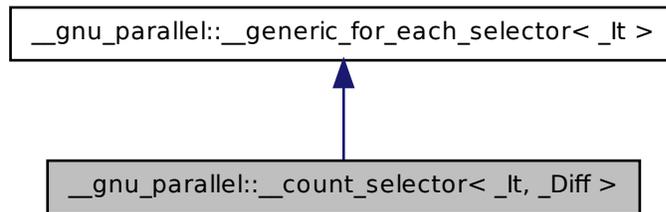
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.97 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

`std::count()` selector.

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



### Public Member Functions

- `template<typename _ValueType > _Diff operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.97.1 Detailed Description

```
template<typename _It, typename _Diff> struct __gnu_parallel::__count_  
selector< _It, _Diff >
```

std::count() selector.

Definition at line 180 of file for\_each\_selectors.h.

### 5.97.2 Member Function Documentation

```
5.97.2.1 template<typename _It, typename _Diff> template<typename  
_ValueType > _Diff __gnu_parallel::__count_selector< _It, _Diff  
>::operator() ( _ValueType & __v, _It __i ) [inline]
```

Functor execution.

#### Parameters

`__v` Current value.

`__i` iterator referencing object.

#### Returns

1 if count, 0 if does not count.

Definition at line 188 of file for\_each\_selectors.h.

### 5.97.3 Member Data Documentation

```
5.97.3.1 template<typename _It > _It __gnu_parallel::__  
generic_for_each_selector< _It >::__M_finish_iterator  
[inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file for\_each\_selectors.h.

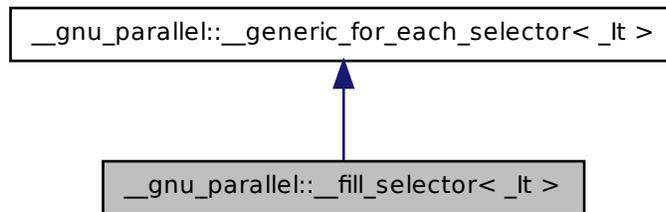
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.98 `__gnu_parallel::__fill_selector<_It >` Struct Template Reference

`std::fill()` selector.

Inheritance diagram for `__gnu_parallel::__fill_selector<_It >`:



### Public Member Functions

- `template<typename _ValueType > bool operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.98.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector<_It >`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

## 5.98.2 Member Function Documentation

**5.98.2.1** `template<typename _It > template<typename _ValueType > bool  
__gnu_parallel::__fill_selector< _It >::operator() ( _ValueType &  
__v, _It __i ) [inline]`

Functor execution.

### Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

## 5.98.3 Member Data Documentation

**5.98.3.1** `template<typename _It > _It __gnu_parallel::__-  
generic_for_each_selector< _It >::M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

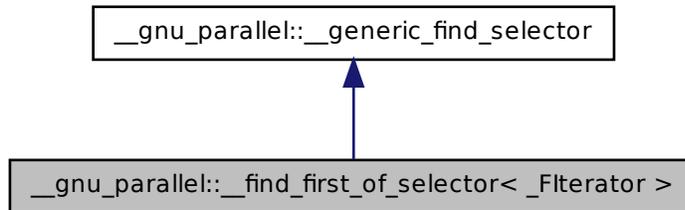
- [for\\_each\\_selectors.h](#)

## 5.99 `__gnu_parallel::__find_first_of_selector< _- FIterator >` Struct Template Reference

Test predicate on several elements.

## 5.99 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference 1049

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



### Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred> std::pair<_RAIter1, _RAIter2> __M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred> bool operator()` (`_RAIter1 __i1`, `_RAIter2 __i2`, `_Pred __pred`)

### Public Attributes

- `_FIterator __M_begin`
- `_FIterator __M_end`

#### 5.99.1 Detailed Description

```
template<typename _FIterator> struct __gnu_parallel::__find_first_of_selector<_FIterator>
```

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

## 5.99.2 Member Function Documentation

**5.99.2.1** `template<typename _FIterator > template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator >::__M_sequential_algorithm ( _RAIter1 begin1, _RAIter1 end1, _RAIter2 begin2, _Pred pred ) [inline]`

Corresponding sequential algorithm on a sequence.

### Parameters

- begin1* Begin iterator of first sequence.
- end1* End iterator of first sequence.
- begin2* Begin iterator of second sequence.
- pred* Find predicate.

Definition at line 186 of file `find_selectors.h`.

**5.99.2.2** `template<typename _FIterator > template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__find_first_of_selector<_FIterator >::operator() ( _RAIter1 i1, _RAIter2 i2, _Pred pred ) [inline]`

Test on one position.

### Parameters

- i1* Iterator on first sequence.
- i2* Iterator on second sequence (unused).
- pred* Find predicate.

Definition at line 169 of file `find_selectors.h`.

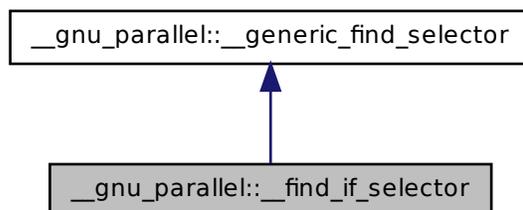
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.100 `__gnu_parallel::__find_if_selector` Struct Reference

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 5.100.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

### 5.100.2 Member Function Documentation

**5.100.2.1** `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> \_\_gnu\_parallel::\_\_find\_if\_selector::\_\_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

#### Parameters

`__begin1` Begin iterator of first sequence.

*\_\_end1* End iterator of first sequence.

*\_\_begin2* Begin iterator of second sequence.

*\_\_pred* Find predicate.

Definition at line 72 of file find\_selectors.h.

**5.100.2.2** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred > bool __gnu_parallel::__find_if_selector::operator() (  
_RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

#### Parameters

*\_\_i1* Iterator on first sequence.

*\_\_i2* Iterator on second sequence (unused).

*\_\_pred* Find predicate.

Definition at line 60 of file find\_selectors.h.

The documentation for this struct was generated from the following file:

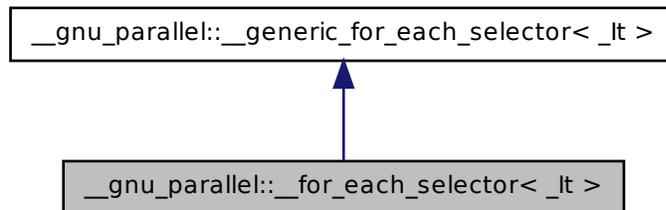
- [find\\_selectors.h](#)

## 5.101 `__gnu_parallel::__for_each_selector< _It >` Struct Template Reference

`std::for_each()` selector.

## 5.101 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



### Public Member Functions

- `template<typename _Op > bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.101.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__for_each_selector<_It >`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

#### 5.101.2 Member Function Documentation

**5.101.2.1** `template<typename _It> template<typename _Op > bool __gnu_parallel::__for_each_selector<_It >::operator() ( _Op & __o, _It __i ) [inline]`

Functor execution.

**Parameters**

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

**5.101.3 Member Data Documentation****5.101.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It >::__M_finish_iterator` [inherited]**

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

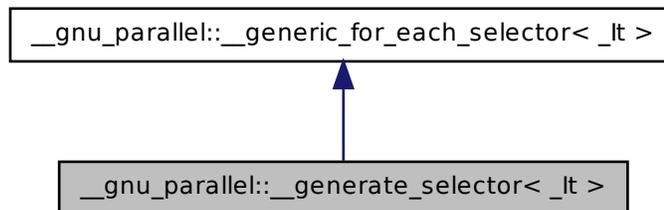
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.102 `__gnu_parallel::__generate_selector<_It >` Struct Template Reference**

`std::generate()` selector.

Inheritance diagram for `__gnu_parallel::__generate_selector<_It >`:



## 5.102 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

### Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &_o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.102.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector<_It >`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

#### 5.102.2 Member Function Documentation

**5.102.2.1** `template<typename _It> template<typename _Op > bool`  
`__gnu_parallel::__generate_selector<_It >::operator() ( _Op &`  
`__o, _It __i ) [inline]`

Functor execution.

##### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

#### 5.102.3 Member Data Documentation

**5.102.3.1** `template<typename _It > _It __gnu_parallel::__-`  
`generic_for_each_selector<_It >::_M_finish_iterator`  
`[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

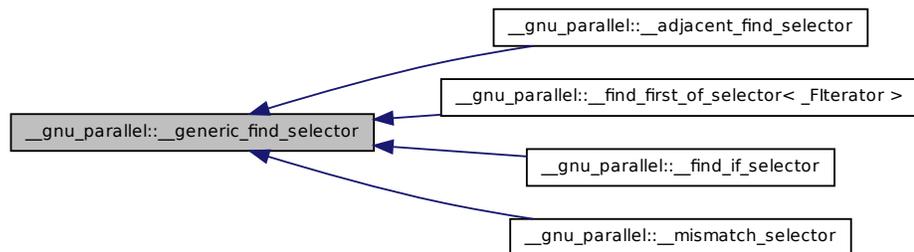
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.103 `__gnu_parallel::__generic_find_selector` Struct Reference

Base class of all `__gnu_parallel::__find_template` selectors.

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



#### 5.103.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

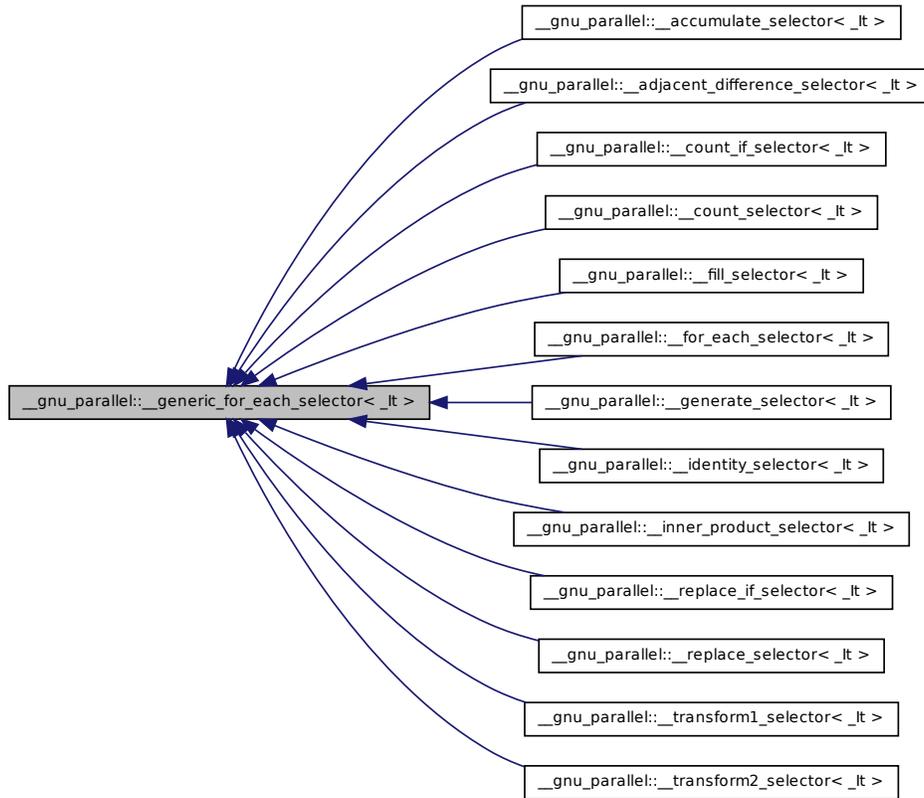
### 5.104 `__gnu_parallel::__generic_for_each_selector<_It >` Struct Template Reference

Generic `__selector` for embarrassingly parallel functions.

## 5.104 `__gnu_parallel::__generic_for_each_selector<_It>` Struct Template Reference

1057

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



### Public Attributes

- `_It_M_finish_iterator`

### 5.104.1 Detailed Description

```
template<typename _It> struct __gnu_parallel::__generic_for_each_selector<_It>
```

Generic `__selector` for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

## 5.104.2 Member Data Documentation

### 5.104.2.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It >::_M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

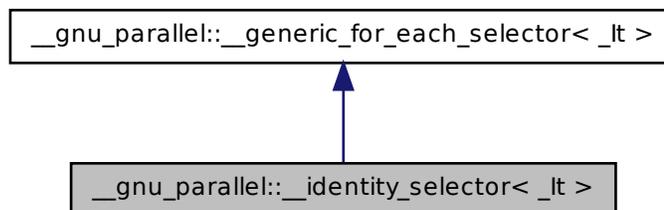
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.105 `__gnu_parallel::__identity_selector<_It >` Struct Template Reference

Selector that just returns the passed iterator.

Inheritance diagram for `__gnu_parallel::__identity_selector<_It >`:



## Public Member Functions

- `template<typename _Op > _It operator() (_Op __o, _It __i)`

## 5.105 \_\_gnu\_parallel::\_\_identity\_selector<\_It> Struct Template Reference1059

### Public Attributes

- [\\_It\\_M\\_finish\\_iterator](#)

### 5.105.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__identity_selector<_It>`

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

### 5.105.2 Member Function Documentation

**5.105.2.1** `template<typename _It> template<typename _Op > _It  
__gnu_parallel::__identity_selector<_It>::operator() ( _Op __o,  
_It __i ) [inline]`

Functor execution.

#### Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

#### Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

### 5.105.3 Member Data Documentation

**5.105.3.1** `template<typename _It > _It __gnu_parallel::__-  
generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

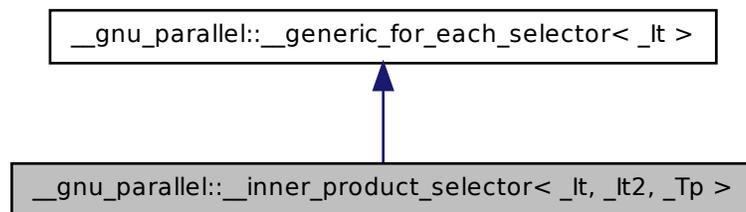
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.106 `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>` Struct Template Reference

[std::inner\\_product\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`:



### Public Member Functions

- [\\_\\_inner\\_product\\_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`  
`_Tp operator()` (`_Op __mult, _It __current`)

### Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

#### 5.106.1 Detailed Description

```
template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>
```

[std::inner\\_product\(\)](#) selector.

Definition at line 222 of file `for_each_selectors.h`.

## 5.106.2 Constructor & Destructor Documentation

5.106.2.1 `template<typename _It, typename _It2, typename _Tp >`  
`__gnu_parallel::__inner_product_selector<_It, _It2, _Tp`  
`>::__inner_product_selector( _It b1, _It2 b2 ) [inline,`  
`explicit]`

Constructor.

### Parameters

- b1* Begin iterator of first sequence.
- b2* Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

## 5.106.3 Member Function Documentation

5.106.3.1 `template<typename _It, typename _It2, typename _Tp >`  
`template<typename _Op > _Tp __gnu_parallel::__inner_product_`  
`selector<_It, _It2, _Tp >::operator() ( _Op mult, _It current )`  
`[inline]`

Functor execution.

### Parameters

- mult* Multiplication functor.
- current* iterator referencing object.

### Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp >::__begin1_`-  
iterator, and `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp >::__begin2_`-  
iterator.

## 5.106.4 Member Data Documentation

5.106.4.1 `template<typename _It, typename _It2, typename _Tp >`  
`_It __gnu_parallel::__inner_product_selector<_It, _It2, _Tp`  
`>::__begin1_iterator`

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator()`.

**5.106.4.2** `template<typename _It, typename _It2, typename _Tp >  
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp  
>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator()`.

**5.106.4.3** `template<typename _It > _It __gnu_parallel::__-  
generic_for_each_selector< _It >::__M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.107 `__gnu_parallel::__max_element_reduct< _- Compare, _It >` Struct Template Reference

Reduction for finding the maximum element, using a comparator.

### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

### 5.107.1 Detailed Description

`template<typename _Compare, typename _It> struct __gnu_parallel::__max_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.108 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

Reduction for finding the maximum element, using a comparator.

### Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

### 5.108.1 Detailed Description

`template<typename _Compare, typename _It> struct __gnu_parallel::__min_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 307 of file `for_each_selectors.h`.

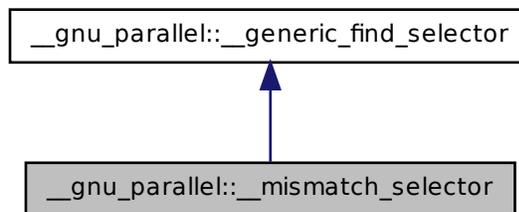
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.109 `__gnu_parallel::__mismatch_selector` Struct Reference

Test inverted predicate on a single element.

Inheritance diagram for `__gnu_parallel::__mismatch_selector`:



### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 5.109.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

**5.110** `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch<__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`  
Template Reference 1065  
**5.109.2** Member Function Documentation

---

**5.109.2.1** `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred ) [inline]`

Corresponding sequential algorithm on a sequence.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__pred` Find predicate.

Definition at line 143 of file `find_selectors.h`.

**5.109.2.2** `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

#### Parameters

`__i1` Iterator on first sequence.  
`__i2` Iterator on second sequence (unused).  
`__pred` Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

**5.110** `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`  
Template Reference

Switch for 3-way merging with `__sentinels` turned off.

## Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

### 5.110.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__-
multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _-
RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned off. Note that 3-way merging is always stable!

Definition at line 748 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 3-way merging with `__sentinels` turned on.

## Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

### 5.111.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _-
DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_-
merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _-
DifferenceTp, _Compare >
```

Switch for 3-way merging with `__sentinels` turned on. Note that 3-way merging is always stable!

**5.112 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch`**  
**`< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`**  
**Template Reference** **1067**  
Definition at line 768 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.112 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch`**  
**`< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`**  
**Template Reference**

Switch for 4-way merging with `__sentinels` turned off.

### Public Member Functions

- `_RAIter3 operator()` (`_RAIterIterator __seqs_begin`, `_RAIterIterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

#### 5.112.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__
multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _
RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned off. Note that 4-way merging is always stable!

Definition at line 791 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.113 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch`**  
**`< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`**  
**Template Reference**

Switch for 4-way merging with `__sentinels` turned on.

## Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

### 5.113.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
typename _Compare> struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true,
_RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on. Note that 4-way merging is always stable!

Definition at line 811 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.114 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Switch for k-way merging with `__sentinels` turned on.

## Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

**5.115** `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch`< `false`,  
`__stable`, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` > Struct  
Template Reference 1069

#### 5.114.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, type-  
name _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_  
parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable,  
_RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned on.

Definition at line 833 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.115** `__gnu_parallel::__multiway_merge_k_  
variant_sentinel_switch`< `false`, `__stable`, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` > Struct Template Reference

Switch for k-way merging with `__sentinels` turned off.

#### Public Member Functions

- `_RAIter3 operator()` (`_RAIterIterator __seqs_begin`, `_RAIterIterator __seqs_end`, `_RAIter3 __target`, `const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp`)

#### 5.115.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3,  
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__  
multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned off.

Definition at line 868 of file `multiway_merge.h`.

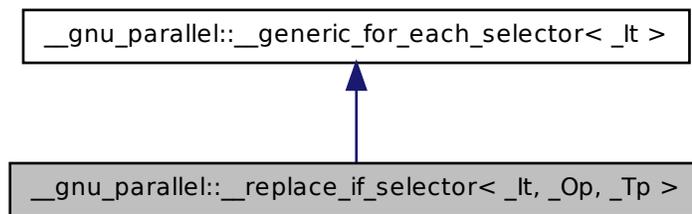
The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.116 `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`:



### Public Member Functions

- `__replace_if_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Op &__o, _It __i`)

### Public Attributes

- `const _Tp & __new_val`
- `_It __M_finish_iterator`

#### 5.116.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >
```

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

## 5.116.2 Constructor & Destructor Documentation

**5.116.2.1** `template<typename _It, typename _Op, typename _Tp  
> __gnu_parallel::__replace_if_selector<_It, _Op, _Tp  
>::__replace_if_selector( const _Tp & __new_val ) [inline,  
explicit]`

Constructor.

### Parameters

`__new_val` Value to replace with.

Definition at line 164 of file `for_each_selectors.h`.

## 5.116.3 Member Function Documentation

**5.116.3.1** `template<typename _It, typename _Op, typename _Tp > bool  
__gnu_parallel::__replace_if_selector<_It, _Op, _Tp >::operator()(  
_Op & __o, _It __i ) [inline]`

Functor execution.

### Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp >::__new_val`.

## 5.116.4 Member Data Documentation

**5.116.4.1** `template<typename _It, typename _Op, typename _Tp > const  
_Tp& __gnu_parallel::__replace_if_selector<_It, _Op, _Tp  
>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp >::operator()()`.

#### 5.116.4.2 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

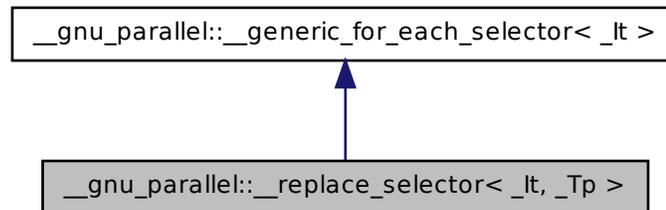
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.117 `__gnu_parallel::__replace_selector< _It, _Tp >` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_selector< _It, _Tp >`:



#### Public Member Functions

- [\\_\\_replace\\_selector](#) (`const _Tp &__new_val`)
- `bool operator() (_Tp &__v, _It __i)`

#### Public Attributes

- `const _Tp & __new_val`
- `_It __M_finish_iterator`

## 5.117 \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp > Struct Template Reference

### 5.117.1 Detailed Description

`template<typename _It, typename _Tp> struct __gnu_parallel::__replace_selector<_It, _Tp >`

std::replace() selector.

Definition at line 132 of file for\_each\_selectors.h.

### 5.117.2 Constructor & Destructor Documentation

**5.117.2.1** `template<typename _It, typename _Tp > __gnu_parallel::__replace_selector<_It, _Tp >::__replace_selector ( const _Tp & __new_val ) [inline, explicit]`

Constructor.

#### Parameters

`__new_val` Value to replace with.

Definition at line 140 of file for\_each\_selectors.h.

### 5.117.3 Member Function Documentation

**5.117.3.1** `template<typename _It, typename _Tp > bool __gnu_parallel::__replace_selector<_It, _Tp >::operator() ( _Tp & __v, _It __i ) [inline]`

Functor execution.

#### Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 146 of file for\_each\_selectors.h.

References `__gnu_parallel::__replace_selector<_It, _Tp >::__new_val`.

### 5.117.4 Member Data Documentation

**5.117.4.1** `template<typename _It, typename _Tp > const _Tp& __gnu_parallel::__replace_selector<_It, _Tp >::__new_val`

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It,_Tp>::operator()`.

#### 5.117.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

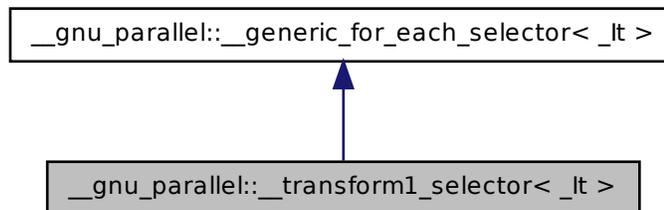
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.118 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, one input sequence variant.

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



#### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

## 5.118 [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector<\\_It>](#) Struct Template Reference 1075

### Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

### 5.118.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform1_selector<_It>`

`std::transform()` \_\_selector, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

### 5.118.2 Member Function Documentation

**5.118.2.1** `template<typename _It> template<typename _Op > bool  
__gnu_parallel::__transform1_selector<_It>::operator()( _Op &  
__o, _It __i ) [inline]`

Functor execution.

#### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

### 5.118.3 Member Data Documentation

**5.118.3.1** `template<typename _It > _It __gnu_parallel::__  
generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

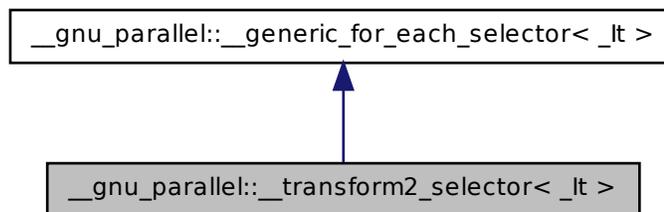
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.119 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

`std::transform()` \_\_selector, two input sequences variant.

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



### Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.119.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform2_selector<_It>`

`std::transform()` \_\_selector, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

## 5.119.2 Member Function Documentation

5.119.2.1 `template<typename _It> template<typename _Op > bool  
__gnu_parallel::__transform2_selector< _It >::operator() ( _Op &  
__o, _It __i ) [inline]`

Functor execution.

### Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

## 5.119.3 Member Data Documentation

5.119.3.1 `template<typename _It > _It __gnu_parallel::__-  
generic_for_each_selector< _It >::_M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

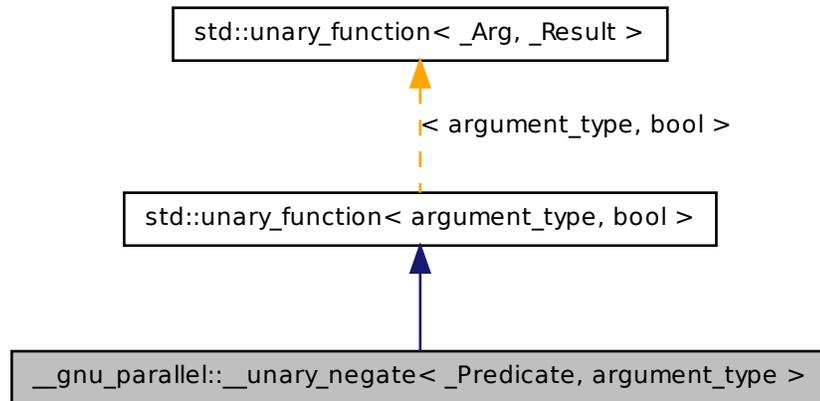
## 5.120 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

Similar to `std::unary_negate`, but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type`

---

&gt;:



## Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

## Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const argument_type &__x`)

## Protected Attributes

- `_Predicate _M_pred`

### 5.120.1 Detailed Description

```
template<typename _Predicate, typename argument_type> class __gnu_parallel::__unary_negate< _Predicate, argument_type >
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

## 5.121 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter > Struct` Template Reference 1079

---

Definition at line 173 of file `parallel/base.h`.

### 5.120.2 Member Typedef Documentation

#### 5.120.2.1 `typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

#### 5.120.2.2 `typedef bool std::unary_function< argument_type , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.121 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter > Struct` Template Reference

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

### Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator\_traits<_RAIter > _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

### Public Member Functions

- `\_DRandomShufflingGlobalData (_RAIter &__source)`

### Public Attributes

- `\_ThreadIndex * \_M\_bin\_proc`

- [\\_DifferenceType\\*\\*\\_M\\_dist](#)
- [int\\_M\\_num\\_bins](#)
- [int\\_M\\_num\\_bits](#)
- [\\_RAIter & \\_M\\_source](#)
- [\\_DifferenceType\\*\\_M\\_starts](#)
- [\\_ValueType\\*\\*\\_M\\_temporaries](#)

### 5.121.1 Detailed Description

```
template<typename _RAIter> struct __gnu_parallel::_-
DRandomShufflingGlobalData<_RAIter >
```

Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).

Definition at line 52 of file `random_shuffle.h`.

### 5.121.2 Constructor & Destructor Documentation

```
5.121.2.1 template<typename _RAIter> __gnu_parallel::_-
DRandomShufflingGlobalData<_RAIter
>::_DRandomShufflingGlobalData ( _RAIter & __source )
[inline]
```

Constructor.

Definition at line 83 of file `random_shuffle.h`.

### 5.121.3 Member Data Documentation

```
5.121.3.1 template<typename _RAIter> _ThreadIndex*
__gnu_parallel::_DRandomShufflingGlobalData<_RAIter
>::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**5.121 \_\_gnu\_parallel::\_DRandomShufflingGlobalData<\_RAIter> Struct**  
**Template Reference** **1081**

---

**5.121.3.2 template<typename \_RAIter> \_DifferenceType\*\*  
\_\_gnu\_parallel::\_DRandomShufflingGlobalData<\_RAIter  
>::\_M\_dist**

Two-dimensional array to hold the thread-bin distribution.

Dimensions (`_M_num_threads + 1`) `__x` (`_M_num_bins + 1`).

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.121.3.3 template<typename \_RAIter> int \_\_gnu\_  
parallel::\_DRandomShufflingGlobalData<\_RAIter  
>::\_M\_num\_bins**

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.121.3.4 template<typename \_RAIter> int \_\_gnu\_  
parallel::\_DRandomShufflingGlobalData<\_RAIter  
>::\_M\_num\_bits**

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.121.3.5 template<typename \_RAIter> \_RAIter& \_\_gnu\_  
parallel::\_DRandomShufflingGlobalData<\_RAIter  
>::\_M\_source**

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

### 5.121.3.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

### 5.121.3.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 5.122 `__gnu_parallel::_DRSSorterPU< _RAIter, _- RandomNumberGenerator > Struct Template Reference`

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

### Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData< \\_RAIter > \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

### 5.122.1 Detailed Description

`template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>`

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 91 of file `random_shuffle.h`.

### 5.122.2 Member Data Documentation

5.122.2.1 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__bins_end`

End index for bins taken care of by this thread.

Definition at line 100 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

5.122.2.2 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_bins_begin`

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

5.122.2.3 `template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_num_threads`

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.122.2.4** `template<typename _RAIter, typename  
_RandomNumberGenerator> _DRandomShufflingGlobalData<_  
RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter,  
_RandomNumberGenerator >::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_parallel_random_shuffle_drs_pu()`.

**5.122.2.5** `template<typename _RAIter, typename  
_RandomNumberGenerator> uint32_t __gnu_parallel::_  
DRSSorterPU< _RAIter, _RandomNumberGenerator  
>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_parallel_random_shuffle_drs_pu()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 5.123 `__gnu_parallel::_DummyReduct` Struct Reference

Reduction function doing nothing.

### Public Member Functions

- `bool operator()` (`bool`, `bool`) `const`

#### 5.123.1 Detailed Description

Reduction function doing nothing.

Definition at line 298 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

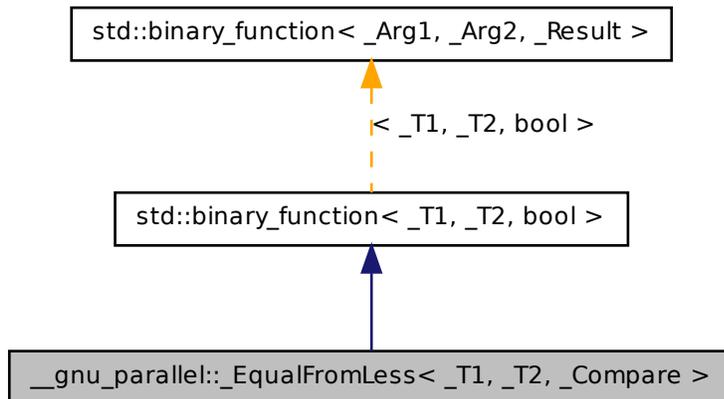
**5.124 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare >` Class Template Reference** 1085

- [for\\_each\\_selectors.h](#)

**5.124 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare >` Class Template Reference**

Constructs predicate for equality from strict weak ordering predicate.

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare >`:



**Public Types**

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

**Public Member Functions**

- `_EqualFromLess` (`_Compare &__comp`)
- `bool operator()` (`const _T1 &__a, const _T2 &__b`)

### 5.124.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file parallel/base.h.

### 5.124.2 Member Typedef Documentation

```
5.124.2.1 typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl\_function.h.

```
5.124.2.2 typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

type of the return type

Definition at line 118 of file stl\_function.h.

```
5.124.2.3 typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

the type of the second argument

Definition at line 117 of file stl\_function.h.

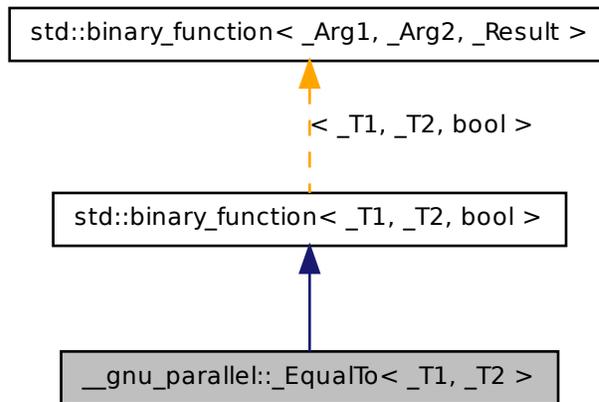
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.125 \_\_gnu\_parallel::\_EqualTo< \_T1, \_T2 > Struct Template Reference

Similar to [std::equal\\_to](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_EqualTo<_T1, _T2 >`:



## Public Types

- typedef `_T1` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_T2` `second_argument_type`

## Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

### 5.125.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_EqualTo<_T1, _T2 >
```

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

## 5.125.2 Member Typedef Documentation

**5.125.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.125.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.125.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.126 `__gnu_parallel::_GuardedIterator< _RAIter, _Compare >` Class Template Reference

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

### Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits< _RAIter >::value_type & operator*` ()
- `_GuardedIterator< _RAIter, _Compare > & operator++` ()

## Friends

- `bool operator< ( _GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`
- `bool operator<= ( _GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`

### 5.126.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator<_RAIter, _Compare >`

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons. The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 66 of file `multiway_merge.h`.

### 5.126.2 Constructor & Destructor Documentation

5.126.2.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator<_RAIter, _Compare >::_GuardedIterator ( _RAIter __begin, _RAIter __end, _Compare & __comp ) [inline]`

Constructor. Sets iterator to beginning of sequence.

#### Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__comp` Comparator provided for associated overloaded compare operators.

Definition at line 84 of file `multiway_merge.h`.

### 5.126.3 Member Function Documentation

5.126.3.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator<_RAIter, _Compare >::operator _RAIter ( ) [inline]`

Convert to wrapped iterator.

**Returns**

Wrapped iterator.

Definition at line 105 of file multiway\_merge.h.

```
5.126.3.2 template<typename _RAIter , typename _Compare
> std::iterator_traits<_RAIter>::value_type&
__gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator*( ) [inline]
```

Dereference operator.

**Returns**

Referenced element.

Definition at line 100 of file multiway\_merge.h.

```
5.126.3.3 template<typename _RAIter , typename _Compare
> _GuardedIterator<_RAIter, _Compare>&
__gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator++( ) [inline]
```

Pre-increment operator.

**Returns**

This.

Definition at line 91 of file multiway\_merge.h.

**5.126.4 Friends And Related Function Documentation**

```
5.126.4.1 template<typename _RAIter , typename _Compare > bool
operator<( _GuardedIterator< _RAIter, _Compare > & __bi1,
_GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

**Parameters**

*\_\_bi1* First iterator.

*\_\_bi2* Second iterator.

**5.127 `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` Class Template Reference** **1091**

---

**Returns**

`true` if less.

Definition at line 113 of file `multiway_merge.h`.

```
5.126.4.2 template<typename _RAIter, typename _Compare > bool  
operator<= ( _GuardedIterator<_RAIter, _Compare > & __bi1,  
_GuardedIterator<_RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

**Parameters**

`__bi1` First iterator.

`__bi2` Second iterator.

**Returns**

`True` if less equal.

Definition at line 128 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

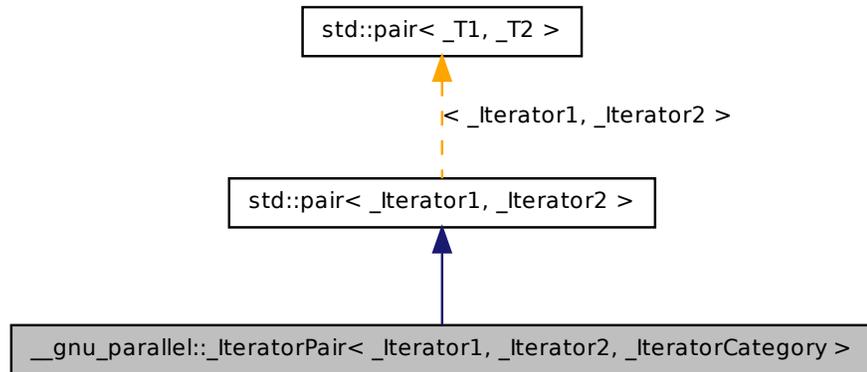
**5.127 `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` Class Template Reference**

A pair of iterators. The usual iterator operations are applied to both child iterators.

Inheritance diagram for `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _-`

---

IteratorCategory >:



## Public Types

- typedef `std::iterator_traits<_Iterator1 >_TraitsType`
- typedef `_TraitsType::difference_type difference_type`
- typedef `_Iterator1 first_type`
- typedef `_IteratorCategory iterator_category`
- typedef `_IteratorPair * pointer`
- typedef `_IteratorPair & reference`
- typedef `_Iterator2 second_type`
- typedef `void value_type`

## Public Member Functions

- `_IteratorPair` (`const _Iterator1 &__first, const _Iterator2 &__second`)
- `operator _Iterator2` () const
- `_IteratorPair operator+` (`difference_type __delta`) const
- `const _IteratorPair operator++` (int)
- `_IteratorPair & operator++` ()
- `difference_type operator-` (`const _IteratorPair &__other`) const
- `const _IteratorPair operator--` (int)
- `_IteratorPair & operator--` ()

## 5.127 `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` 1093 > Class Template Reference

---

- `_IteratorPair` & `operator=` (const `_IteratorPair` &\_\_other)
- void `swap` (pair &\_\_p)

### Public Attributes

- `_Iterator1` `first`
- `_Iterator2` `second`

### 5.127.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _-
IteratorCategory> class __gnu_parallel::IteratorPair<_Iterator1, _Iterator2,
_IteratorCategory >
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

### 5.127.2 Member Typedef Documentation

#### 5.127.2.1 `typedef _Iterator1 std::pair<_Iterator1, _Iterator2>::first_type` `[inherited]`

`first_type` is the first bound type

Definition at line 73 of file `stl_pair.h`.

#### 5.127.2.2 `typedef _Iterator2 std::pair<_Iterator1, _Iterator2>::second_type` `[inherited]`

`second_type` is the second bound type

Definition at line 74 of file `stl_pair.h`.

### 5.127.3 Member Data Documentation

#### 5.127.3.1 `_Iterator1 std::pair<_Iterator1, _Iterator2>::first` `[inherited]`

`first` is a copy of the first object

Definition at line 76 of file `stl_pair.h`.

### 5.127.3.2 `_Iterator2` `std::pair<_Iterator1, _Iterator2>::second` [inherited]

`second` is a copy of the second object

Definition at line 77 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.128 `__gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

A triple of iterators. The usual iterator operations are applied to all three child iterators.

### Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` `difference_type`
- typedef `_IteratorCategory` `iterator_category`
- typedef `_IteratorTriple` \* `pointer`
- typedef `_IteratorTriple` & `reference`
- typedef void `value_type`

### Public Member Functions

- `_IteratorTriple` (`const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third`)
- `operator _Iterator3` () const
- `_IteratorTriple` `operator+` (`difference_type __delta`) const
- `const _IteratorTriple` `operator++` (int)
- `_IteratorTriple` & `operator++` ()
- `difference_type` `operator-` (`const _IteratorTriple &__other`) const
- `_IteratorTriple` & `operator--` ()
- `const _IteratorTriple` `operator--` (int)
- `_IteratorTriple` & `operator=` (`const _IteratorTriple &__other`)

## Public Attributes

- `_Iterator1 _M_first`
- `_Iterator2 _M_second`
- `_Iterator3 _M_third`

### 5.128.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, type-  
name _IteratorCategory> class __gnu_parallel::_IteratorTriple< _Iterator1, _-  
Iterator2, _Iterator3, _IteratorCategory >
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.129 `__gnu_parallel::_Job<_DifferenceTp >` Struct Template Reference

One `__job` for a certain thread.

## Public Types

- `typedef _DifferenceTp _DifferenceType`

## Public Attributes

- `volatile _DifferenceType _M_first`
- `volatile _DifferenceType _M_last`
- `volatile _DifferenceType _M_load`

### 5.129.1 Detailed Description

```
template<typename _DifferenceTp> struct __gnu_parallel::_Job< _-  
DifferenceTp >
```

One `__job` for a certain thread.

Definition at line 54 of file workstealing.h.

## 5.129.2 Member Data Documentation

### 5.129.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file workstealing.h.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

### 5.129.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file workstealing.h.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

### 5.129.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load`

Number of elements, i.e. `_M_last-_M_first+1`.

Changed by owning thread only.

Definition at line 72 of file workstealing.h.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

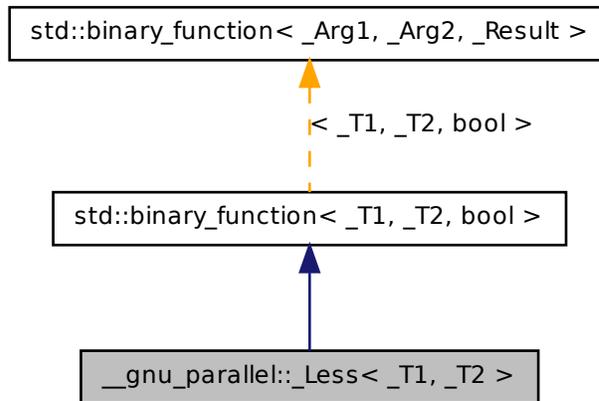
The documentation for this struct was generated from the following file:

- [workstealing.h](#)

## 5.130 `__gnu_parallel::_Less< _T1, _T2 > Struct Tem- plate Reference`

Similar to `std::less`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

### 5.130.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2 >
```

Similar to [std::less](#), but allows two different types.

Definition at line 252 of file `parallel/base.h`.

## 5.130.2 Member Typedef Documentation

**5.130.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool  
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.130.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.130.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

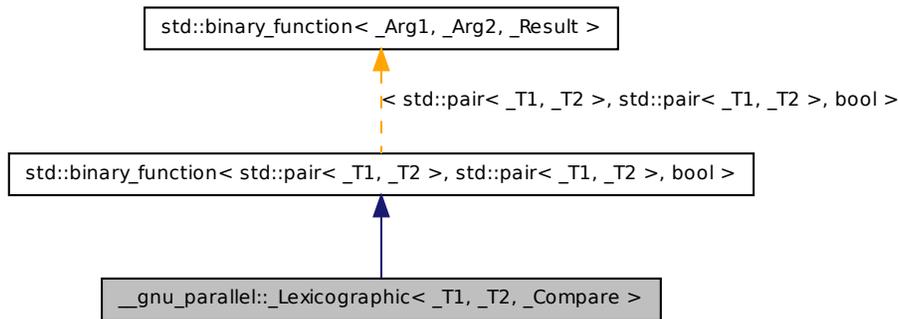
- [parallel/base.h](#)

## 5.131 `__gnu_parallel::_Lexicographic< _T1, _T2, _- Compare >` Class Template Reference

Compare \_\_a pair of types lexicographically, ascending.

## 5.131 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>` Class Template Reference 1099

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:



### Public Types

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

### Public Member Functions

- `_Lexicographic` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

#### 5.131.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`

Compare \_\_a pair of types lexicographically, ascending.

Definition at line 55 of file `multiseq_selection.h`.

## 5.131.2 Member Typedef Documentation

**5.131.2.1** `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::first_argument_type` **[inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.131.2.2** `typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::result_type` **[inherited]**

type of the return type

Definition at line 118 of file stl\_function.h.

**5.131.2.3** `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type` **[inherited]**

the type of the second argument

Definition at line 117 of file stl\_function.h.

The documentation for this class was generated from the following file:

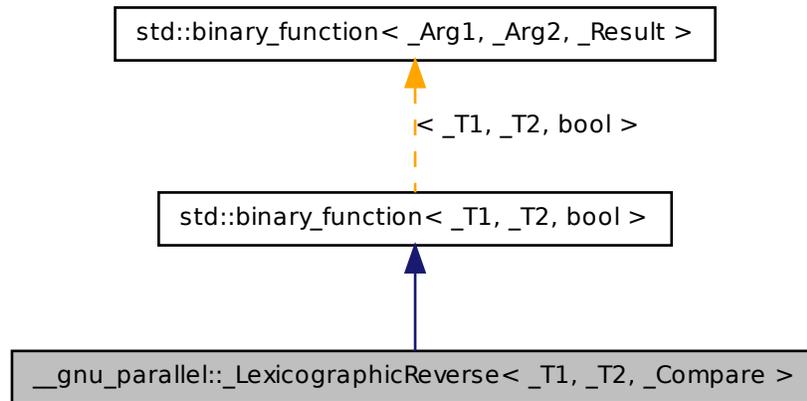
- [multiseq\\_selection.h](#)

## 5.132 `__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >` Class Template Reference

Compare \_\_a pair of types lexicographically, descending.

Inheritance diagram for `__gnu_parallel::_LexicographicReverse< _T1, _T2, _-`

Compare >:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

## Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

### 5.132.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`

Compare \_\_a pair of types lexicographically, descending.

Definition at line 82 of file `multiseq_selection.h`.

## 5.132.2 Member Typedef Documentation

**5.132.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.132.2.2** `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.132.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

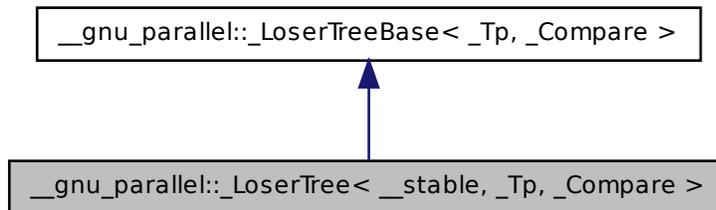
- [multiseq\\_selection.h](#)

## 5.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Stable `_LoserTree` variant.

## 5.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference 1103

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

### Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.133.1 Detailed Description

template<bool `__stable`, typename `_Tp`, typename `_Compare`> class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`

Stable `_LoserTree` variant. Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 165 of file losertree.h.

## 5.133.2 Member Function Documentation

**5.133.2.1** `template<bool __stable, typename _Tp , typename _Compare  
> void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare  
>::_delete_min_insert ( _Tp __key, bool __sup ) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 217 of file losertree.h.

References `std::swap()`.

**5.133.2.2** `template<typename _Tp , typename _Compare >  
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare  
>::_get_min_source ( ) [inline, inherited]`

### Returns

the index of the sequence with the smallest element.

Definition at line 151 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

**5.133.2.3** `template<typename _Tp , typename _Compare > void  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start  
( const _Tp & __key, int __source, bool __sup ) [inline,  
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

### Parameters

`__key` the element to insert

`__source` \_\_index of the `__source` \_\_sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file losertree.h.

## 5.133 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference 1105

---

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.133.3 Member Data Documentation

#### 5.133.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` [protected, inherited]

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

#### 5.133.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert` [protected, inherited]

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

#### 5.133.3.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` [protected, inherited]

`log_2{ _M_k }`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

#### 5.133.3.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` [protected, inherited]

`_LoserTree` `__elements`.

Definition at line 75 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

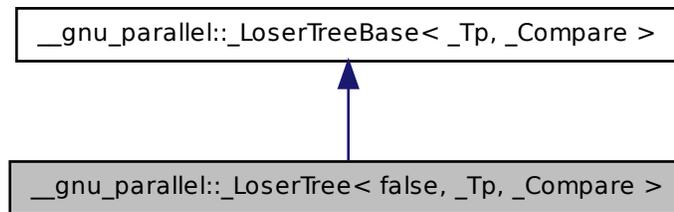
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.134 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Unstable `_LoserTree` variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



#### Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- `void __delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- `int __get_min_source` ()
- `void __init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- `void __insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

## Protected Attributes

- `_Compare` `_M_comp`
- `bool` `_M_first_insert`
- `unsigned int` `_M_ik`
- `unsigned int` `_M_k`
- `unsigned int` `_M_log_k`
- `_Loser *` `_M_losers`
- `unsigned int` `_M_offset`

### 5.134.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable `_LoserTree` variant. Stability (non-stable here) is selected with partial specialization.

Definition at line 255 of file `losertree.h`.

### 5.134.2 Member Function Documentation

5.134.2.1 `template<typename _Tp , typename _Compare > void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert ( _Tp __key, bool __sup ) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

#### Parameters

- `__key` the `_M_key` to insert
- `__sup` true iff `__key` is an explicitly marked supremum

Definition at line 317 of file `losertree.h`.

References `std::swap()`.

5.134.2.2 `template<typename _Tp , typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source ( ) [inline, inherited]`

#### Returns

- the index of the sequence with the smallest element.

Definition at line 151 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

**5.134.2.3** `template<typename _Tp, typename _Compare > unsigned int  
__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner  
( unsigned int __root ) [inline]`

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

#### Parameters

`__root` \_\_index of the "game" to start.

Definition at line 277 of file losertree.h.

**5.134.2.4** `template<typename _Tp, typename _Compare > void  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start  
( const _Tp & __key, int __source, bool __sup ) [inline,  
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

`__key` the element to insert

`__source` \_\_index of the `__source` \_\_sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.134.3 Member Data Documentation

**5.134.3.1** `template<typename _Tp , typename _Compare > _Compare  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp  
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

**5.134.3.2** `template<typename _Tp , typename _Compare >  
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare  
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

**5.134.3.3** `template<typename _Tp , typename _Compare > unsigned int  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  
[protected, inherited]`

`log_2{ _M_k }`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

**5.134.3.4** `template<typename _Tp , typename _Compare > _Loser*  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers  
[protected, inherited]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

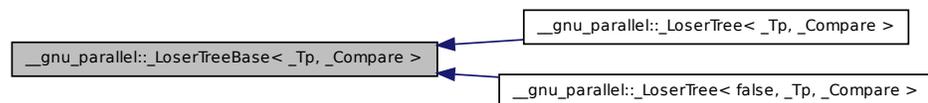
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.135 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Guarded loser/tournament tree.

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



#### Classes

- `struct _Loser`

*Internal representation of a `_LoserTree` element.*

#### Public Member Functions

- `_LoserTreeBase` (unsigned int `__k`, `_Compare __comp`)
- `~_LoserTreeBase` ()
- `int __get_min_source` ()
- `void __insert_start` (const `_Tp &__key`, int `__source`, bool `__sup`)

#### Protected Attributes

- `_Compare _M_comp`
- `bool _M_first_insert`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `unsigned int _M_log_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

### 5.135.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeBase<_Tp, _Compare >
```

Guarded loser/tournament tree. The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

#### Parameters

`_Tp` the element type

`_Compare` the comparator to use, defaults to `std::less<_Tp>`

Definition at line 55 of file `losertree.h`.

### 5.135.2 Constructor & Destructor Documentation

```
5.135.2.1 template<typename _Tp , typename _Compare >
        __gnu_parallel::_LoserTreeBase<_Tp, _Compare
        >::_LoserTreeBase ( unsigned int __k, _Compare __comp )
        [inline]
```

The constructor.

#### Parameters

`__k` The number of sequences to merge.

`__comp` The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::_rd_log2()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_M_losers`.

```
5.135.2.2 template<typename _Tp , typename _Compare >
        __gnu_parallel::_LoserTreeBase<_Tp, _Compare
        >::~~_LoserTreeBase ( ) [inline]
```

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_M_losers`.

### 5.135.3 Member Function Documentation

**5.135.3.1** `template<typename _Tp , typename _Compare >  
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::  
>::_get_min_source ( ) [inline]`

#### Returns

the index of the sequence with the smallest element.

Definition at line 151 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

**5.135.3.2** `template<typename _Tp , typename _Compare > void  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start  
( const _Tp & __key, int __source, bool __sup ) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

`__key` the element to insert

`__source` \_\_index of the `__source` \_\_sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.135.4 Member Data Documentation

**5.135.4.1** `template<typename _Tp , typename _Compare > _Compare  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp  
[protected]`

`_Compare` to use.

Definition at line 78 of file losertree.h.

## 5.136 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference 1113

---

**5.135.4.2** `template<typename _Tp, typename _Compare >  
bool __gnu_parallel::_LoserTreeBase<_Tp, _Compare  
>::_M_first_insert [protected]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_LoserTreeBase()`.

**5.135.4.3** `template<typename _Tp, typename _Compare > unsigned int  
__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_log_k  
[protected]`

`log_2{M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_LoserTreeBase()`.

**5.135.4.4** `template<typename _Tp, typename _Compare > _Loser*  
__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_losers  
[protected]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_get_min_source()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.136 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference

Internal representation of a `_LoserTree` element.

---

## Public Attributes

- [\\_Tp\\_M\\_key](#)
- [int\\_M\\_source](#)
- [bool\\_M\\_sup](#)

### 5.136.1 Detailed Description

**template<typename \_Tp, typename \_Compare> struct \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser**

Internal representation of a [\\_LoserTree](#) element.

Definition at line 59 of file losertree.h.

### 5.136.2 Member Data Documentation

**5.136.2.1 template<typename \_Tp , typename \_Compare > \_Tp \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_key**

[\\_M\\_key](#) of the element in the [\\_LoserTree](#).

Definition at line 66 of file losertree.h.

Referenced by [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_insert\\_start\(\)](#).

**5.136.2.2 template<typename \_Tp , typename \_Compare > int \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_source**

[\\_\\_index](#) of the [\\_\\_source](#) [\\_\\_sequence](#).

Definition at line 64 of file losertree.h.

Referenced by [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_get\\_min\\_source\(\)](#), and [\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_insert\\_start\(\)](#).

**5.136.2.3 template<typename \_Tp , typename \_Compare > bool \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_sup**

flag, true iff this is a "maximum" [\\_\\_sentinel](#).

Definition at line 62 of file losertree.h.

### 5.137 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference 1115

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

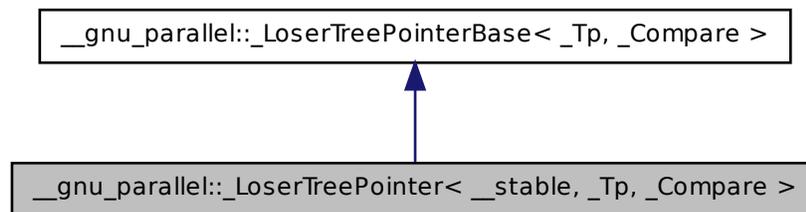
The documentation for this struct was generated from the following file:

- [losertree.h](#)

### 5.137 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Stable `_LoserTree` implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



#### Public Member Functions

- `_LoserTreePointer` (unsigned int `__k`, `_Compare` `__comp=std::less< _Tp >()`)
- `void __delete_min_insert` (const `_Tp` &`__key`, bool `__sup`)
- `int __get_min_source` ()
- `void __init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- `void __insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

#### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`

- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

### 5.137.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

Stable `_LoserTree` implementation. The unstable variant is implemented using partial instantiation below.

Definition at line 401 of file `losertree.h`.

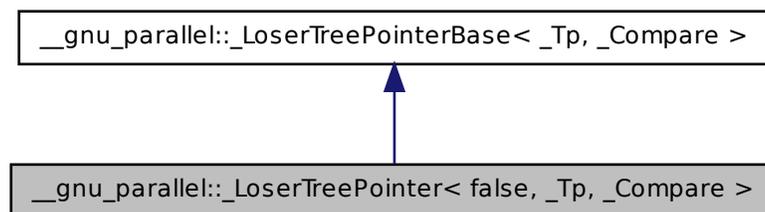
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.138 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Unstable `_LoserTree` implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointer` (unsigned int `_k`, `_Compare __comp=std::less< _Tp >()`)

## 5.139 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >` Class Template Reference

1117

- void `__delete_min_insert` (const `_Tp` &`__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

### 5.138.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`

Unstable `_LoserTree` implementation. The stable variant is above.

Definition at line 482 of file `losertree.h`.

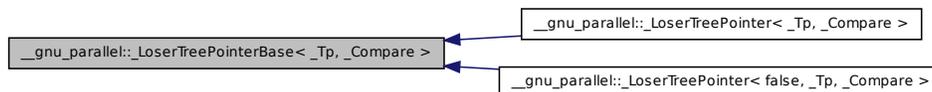
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.139 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >` Class Template Reference

Base class of `_Loser` Tree implementation using pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`:



## Classes

- struct [\\_Loser](#)

*Internal representation of [\\_LoserTree](#) \_\_elements.*

## Public Member Functions

- [\\_LoserTreePointerBase](#) (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- [\\_Compare](#) [\\_M\\_comp](#)
- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_k](#)
- [\\_Loser](#) \* [\\_M\\_losers](#)
- unsigned int [\\_M\\_offset](#)

### 5.139.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >
```

Base class of [\\_Loser](#) Tree implementation using pointers.

Definition at line 349 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.140 [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase](#)< [\\_Tp](#), [\\_Compare](#) >::[\\_Loser](#) Struct Reference

Internal representation of [\\_LoserTree](#) \_\_elements.

**5.141 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference** 1119

---

**Public Attributes**

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

**5.140.1 Detailed Description**

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`

Internal representation of [\\_LoserTree](#) \_\_elements.

Definition at line 353 of file `losertree.h`.

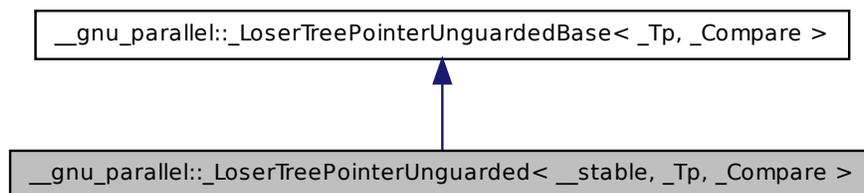
The documentation for this struct was generated from the following file:

- [losertree.h](#)

**5.141 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference**

Stable unguarded [\\_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=std::less<\_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- **\_Compare \_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- **\_Loser \* \_M\_losers**
- unsigned int **\_M\_offset**

### 5.141.1 Detailed Description

**template<bool \_\_stable, typename \_Tp, typename \_Compare> class \_\_gnu\_parallel::\_LoserTreePointerUnguarded< \_\_stable, \_Tp, \_Compare >**

Stable unguarded [\\_LoserTree](#) variant storing pointers. Unstable variant is implemented below using partial specialization.

Definition at line 868 of file losertree.h.

The documentation for this class was generated from the following file:

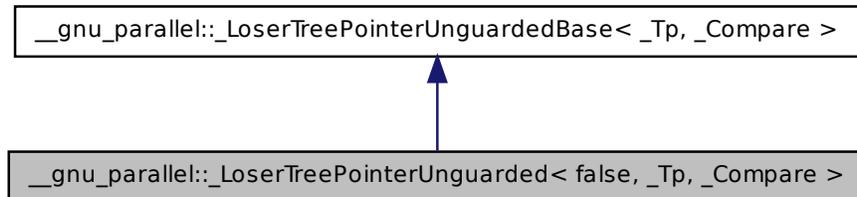
- [losertree.h](#)

### 5.142 **\_\_gnu\_parallel::\_LoserTreePointerUnguarded< false, \_Tp, \_Compare > Class Template Reference**

Unstable unguarded [\\_LoserTree](#) variant storing pointers.

## 5.142 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference 1121

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointerUnguarded` (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- void `__delete_min_insert` (const `_Tp` &`__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.142.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded [\\_LoserTree](#) variant storing pointers. Stable variant is above.

Definition at line 953 of file losertree.h.

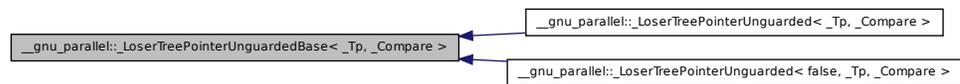
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.143 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare >` Class Template Reference

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare >`:



#### Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=`std::less<_Tp >()`)
- `int __get_min_source` ()
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

## 5.144 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference 1123

### 5.143.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare >
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure. No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 805 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.144 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

Traits for determining whether the loser tree should use pointers or copies.

### Static Public Attributes

- static const bool `_M_use_pointer`

### 5.144.1 Detailed Description

```
template<typename _Tp> struct __gnu_parallel::_LoserTreeTraits<_Tp >
```

Traits for determining whether the loser tree should use pointers or copies. The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
```

```
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

**Parameters**

***`_Tp`*** type to give the loser tree traits for.

Definition at line 727 of file `multiway_merge.h`.

**5.144.2 Member Data Documentation**
**5.144.2.1 `template<typename _Tp > const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer` [static]**

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

Definition at line 735 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

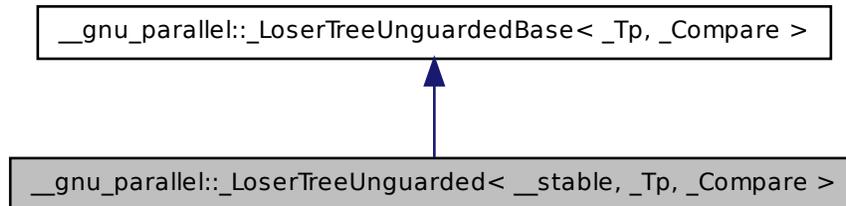
- [multiway\\_merge.h](#)

**5.145 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference**

Stable implementation of unguarded [\\_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _`

Compare >:



### Public Member Functions

- `_LoserTreeUnguarded` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- void `__delete_min_insert` (`_Tp` `__key`, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.145.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`. Unstable variant is selected below with partial specialization.

Definition at line 627 of file losertree.h.

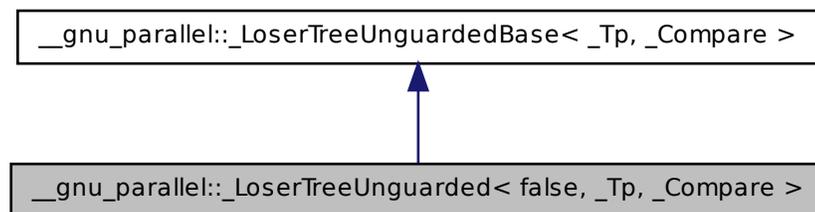
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.146 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Non-Stable implementation of unguarded [\\_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreeUnguarded` (unsigned int \_\_k, const \_Tp \_\_sentinel, \_Compare \_\_comp=`std::less<_Tp>`())
- void `__delete_min_insert` (\_Tp \_\_key, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare _M_comp`

## 5.147 `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>` Class Template Reference 1127

- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

### 5.146.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`

Non-Stable implementation of unguarded `_LoserTree`. Stable implementation is above.

Definition at line 713 of file `losertree.h`.

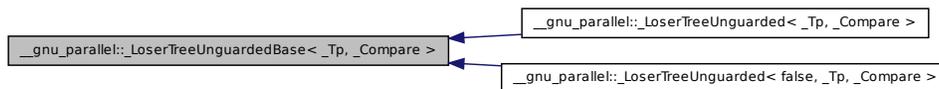
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.147 `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>` Class Template Reference

Base class for unguarded `_LoserTree` implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`:



### Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less<_Tp>()`)
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

## Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

### 5.147.1 Detailed Description

**template<typename `_Tp`, typename `_Compare`> class `__gnu_parallel::_LoserTreeUnguardedBase`< `_Tp`, `_Compare` >**

Base class for unguarded `_LoserTree` implementation. The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

Definition at line 564 of file `losertree.h`.

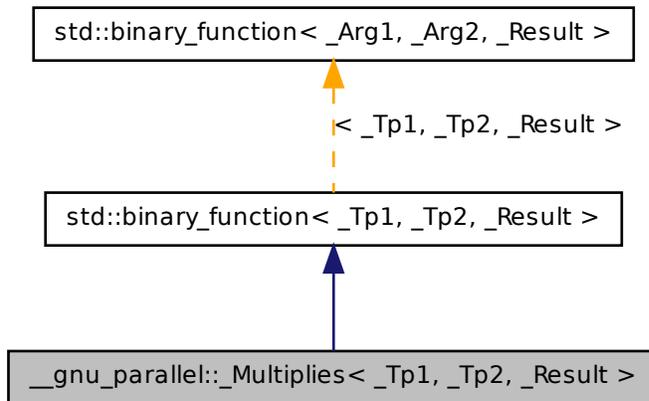
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.148 `__gnu_parallel::_Multiplies`< `_Tp1`, `_Tp2`, `_Result` > Struct Template Reference

Similar to `std::multiplies`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result >`:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result operator()` (const `_Tp1` &`__x`, const `_Tp2` &`__y`) const

### 5.148.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(
(*static_cast<_Tp1*>(NULL) * *static_cast<_Tp2*>(NULL)))> struct __gnu_
parallel::_Multiplies<_Tp1, _Tp2, _Result >
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

## 5.148.2 Member Typedef Documentation

### 5.148.2.1 `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

### 5.148.2.2 `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

### 5.148.2.3 `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.149 `__gnu_parallel::_Nothing` Struct Reference

Functor doing nothing.

### Public Member Functions

- `template<typename _It > void operator() (_It)`

### 5.149.1 Detailed Description

Functor doing nothing. For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

## 5.150 `__gnu_parallel::_Piece<_DifferenceTp > Struct` Template Reference 1131

### 5.149.2 Member Function Documentation

5.149.2.1 `template<typename _It > void __gnu_parallel::_Nothing::operator()`  
`( _It ) [inline]`

Functor execution.

#### Parameters

`__i` iterator referencing object.

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.150 `__gnu_parallel::_Piece<_DifferenceTp > Struct` Template Reference

Subsequence description.

### Public Types

- `typedef _DifferenceTp _DifferenceType`

### Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

### 5.150.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Piece< _-`  
`DifferenceTp >`

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

## 5.150.2 Member Data Documentation

### 5.150.2.1 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

### 5.150.2.2 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end`

End of subsequence.

Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

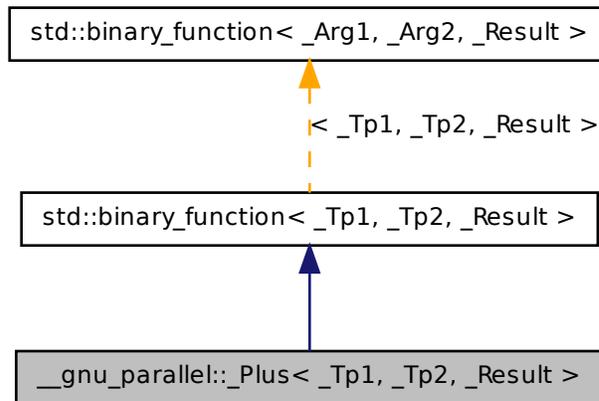
- [multiway\\_mergesort.h](#)

## 5.151 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

Similar to [std::plus](#), but allows two different types.

## 5.151 `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`:



### Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Tp2` `second_argument_type`

### Public Member Functions

- `_Result operator()` (const `_Tp1` &`__x`, const `_Tp2` &`__y`) const

#### 5.151.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof_  
(*static_cast<_Tp1*>(NULL) + *static_cast<_Tp2*>(NULL))> struct __gnu_  
parallel::_Plus<_Tp1, _Tp2, _Result >
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.

## 5.151.2 Member Typedef Documentation

**5.151.2.1** `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.151.2.2** `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.151.2.3** `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.152 `__gnu_parallel::PMWMSortingData< _RAIter >` Struct Template Reference

Data accessed by all threads.

### Public Types

- `typedef _TraitsType::difference_type DifferenceType`
- `typedef std::iterator\_traits< _RAIter > TraitsType`
- `typedef _TraitsType::value_type ValueType`

### Public Attributes

- [\\_ThreadIndex](#) `_M_num_threads`
- `_DifferenceType * _M_offsets`

- [std::vector<\\_Piece<\\_DifferenceType > > \\* \\_M\\_pieces](#)
- [\\_ValueType \\* \\_M\\_samples](#)
- [\\_RAIter \\_M\\_source](#)
- [\\_DifferenceType \\* \\_M\\_starts](#)
- [\\_ValueType \\*\\* \\_M\\_temporary](#)

### 5.152.1 Detailed Description

**template<typename \_RAIter> struct \_\_gnu\_parallel::\_PMWMSortingData<\_RAIter >**

Data accessed by all threads. PMWMS = parallel multiway mergesort

Definition at line 61 of file multiway\_mergesort.h.

### 5.152.2 Member Data Documentation

**5.152.2.1 template<typename \_RAIter> \_ThreadIndex  
\_\_gnu\_parallel::\_PMWMSortingData<\_RAIter  
>::\_M\_num\_threads**

Number of threads involved.

Definition at line 68 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

**5.152.2.2 template<typename \_RAIter> \_DifferenceType\*  
\_\_gnu\_parallel::\_PMWMSortingData<\_RAIter >::\_M\_offsets**

Offsets to add to the found positions.

Definition at line 83 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

**5.152.2.3 template<typename \_RAIter> std::vector<\_Piece<\_-  
DifferenceType> > \* \_\_gnu\_parallel::\_PMWMSortingData<  
\_RAIter >::\_M\_pieces**

Pieces of data to merge [thread][\_\_sequence].

Definition at line 86 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.152.2.4 `template<typename _RAIter> _ValueType*` `__gnu_parallel::PWMSSortingData<_RAIter>::_M_samples`

Samples.

Definition at line 80 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, and `__gnu_parallel::parallel_sort_mwms()`.

#### 5.152.2.5 `template<typename _RAIter> _RAIter` `__gnu_parallel::PWMSSortingData<_RAIter>::_M_source`

Input `__begin`.

Definition at line 71 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.152.2.6 `template<typename _RAIter> _DifferenceType*` `__gnu_parallel::PWMSSortingData<_RAIter>::_M_starts`

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.152.2.7 `template<typename _RAIter> _ValueType**` `__gnu_parallel::PWMSSortingData<_RAIter>::_M_temporary`

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.153 `__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >` Class Template Reference

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

### Public Types

- typedef `_DifferenceTp` `_DifferenceType`
- typedef `_PseudoSequenceIterator< _Tp, uint64_t >` `iterator`

### Public Member Functions

- `_PseudoSequence` (`const _Tp &__val, _DifferenceType __count`)
- `iterator begin` () const
- `iterator end` () const

#### 5.153.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >`

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

#### Parameters

- `_Tp` Sequence `_M_value` type.
- `_DifferenceType` Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

#### 5.153.2 Constructor & Destructor Documentation

5.153.2.1 `template<typename _Tp, typename _DifferenceTp> __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence ( const _Tp & __val, _DifferenceType __count ) [inline]`

Constructor.

**Parameters**

- \_M\_val* Element of the sequence.
- \_\_count* Number of (virtual) copies.

Definition at line 371 of file parallel/base.h.

**5.153.3 Member Function Documentation**

**5.153.3.1** `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp >::begin ( )  
const [inline]`

Begin iterator.

Definition at line 376 of file parallel/base.h.

**5.153.3.2** `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp >::end ( )  
const [inline]`

End iterator.

Definition at line 381 of file parallel/base.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

**5.154 \_\_gnu\_parallel::\_PseudoSequenceIterator< \_Tp, \_DifferenceTp > Class Template Reference**

*\_Iterator* associated with [\\_\\_gnu\\_parallel::\\_PseudoSequence](#). It features the usual random-access iterator functionality.

**Public Types**

- `typedef _DifferenceTp _DifferenceType`

**Public Member Functions**

- `_PseudoSequenceIterator` (const *\_Tp* &*\_\_val*, *\_DifferenceType* *\_\_pos*)

- `bool operator!= (const \_PseudoSequenceIterator &__i2)`
- `const _Tp & operator* () const`
- `\_PseudoSequenceIterator & operator++ ()`
- `\_PseudoSequenceIterator operator++ (int)`
- `_DifferenceType operator- (const \_PseudoSequenceIterator &__i2)`
- `bool operator== (const \_PseudoSequenceIterator &__i2)`
- `const _Tp & operator[] (_DifferenceType) const`

### 5.154.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp >`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

#### Parameters

- `_Tp` Sequence `_M_value` type.
- `_DifferenceType` Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.155 `__gnu_parallel::_QSSThreadLocal<_RAIter >` Struct Template Reference

Information local to one thread in the parallel quicksort run.

### Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::pair<_RAIter, _RAIter > _Piece`
- `typedef std::iterator_traits<_RAIter > _TraitsType`

### Public Member Functions

- `\_QSSThreadLocal (int __queue_size)`

## Public Attributes

- volatile `_DifferenceType` \* `_M_elements_leftover`
- `_Piece` `_M_global`
- `_Piece` `_M_initial`
- `_RestrictedBoundedConcurrentQueue`< `_Piece` > `_M_leftover_parts`
- `_ThreadIndex` `_M_num_threads`

### 5.155.1 Detailed Description

```
template<typename _RAIter> struct __gnu_parallel::QSBThreadLocal< _-
RAIter >
```

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

### 5.155.2 Member Typedef Documentation

```
5.155.2.1 template<typename _RAIter> typedef std::pair<_RAIter, _RAIter>
__gnu_parallel::QSBThreadLocal< _RAIter >::_Piece
```

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

### 5.155.3 Constructor & Destructor Documentation

```
5.155.3.1 template<typename _RAIter> __gnu_parallel::QSBThreadLocal<
_RAIter >::_QSBThreadLocal ( int __queue_size ) [inline]
```

Constructor.

#### Parameters

`__queue_size` size of the work-stealing queue.

Definition at line 88 of file `balanced_quicksort.h`.

## 5.155.4 Member Data Documentation

### 5.155.4.1 `template<typename _RAIter> volatile _DifferenceType* __gnu_parallel::__QSBThreadLocal<_RAIter >::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 5.155.4.2 `template<typename _RAIter> _Piece __gnu_parallel::_- QSBThreadLocal<_RAIter>::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

### 5.155.4.3 `template<typename _RAIter> _Piece __gnu_parallel::_- QSBThreadLocal<_RAIter>::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 5.155.4.4 `template<typename _RAIter> _- RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 5.155.4.5 `template<typename _RAIter> _ThreadIndex __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

## 5.156 `__gnu_parallel::_RandomNumber` Class Reference

Random number generator, based on the Mersenne twister.

### Public Member Functions

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_M\_supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)
- uint32\_t [operator](#)() ()
- uint32\_t [operator](#)() (uint64\_t local\_supremum)

### 5.156.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

### 5.156.2 Constructor & Destructor Documentation

#### 5.156.2.1 `__gnu_parallel::_RandomNumber::_RandomNumber ( )` [inline]

Default constructor. Seed with 0.

Definition at line 74 of file `random_number.h`.

#### 5.156.2.2 `__gnu_parallel::_RandomNumber::_RandomNumber ( uint32_t __seed, uint64_t _M_supremum = 0x100000000ULL )` [inline]

Constructor.

#### Parameters

`__seed` Random \_\_seed.

## 5.157 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference 1143

---

`_M_supremum` Generate integer random numbers in the interval  $[0, \_M\_supremum)$ .

Definition at line 85 of file `random_number.h`.

### 5.156.3 Member Function Documentation

**5.156.3.1** `unsigned long __gnu_parallel::_RandomNumber::_genrand_bits ( int bits ) [inline]`

Generate a number of random bits, run-time parameter.

#### Parameters

*bits* Number of bits to generate.

Definition at line 109 of file `random_number.h`.

**5.156.3.2** `uint32_t __gnu_parallel::_RandomNumber::operator() ( ) [inline]`

Generate unsigned random 32-bit integer.

Definition at line 94 of file `random_number.h`.

**5.156.3.3** `uint32_t __gnu_parallel::_RandomNumber::operator() ( uint64_t local_supremum ) [inline]`

Generate unsigned random 32-bit integer in the interval  $[0, \_local\_supremum)$ .

Definition at line 100 of file `random_number.h`.

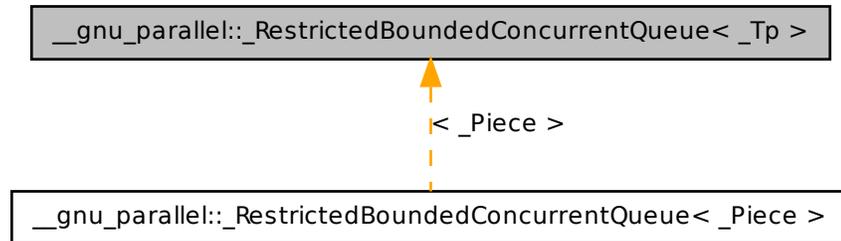
The documentation for this class was generated from the following file:

- [random\\_number.h](#)

## 5.157 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Inheritance diagram for `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`:



## Public Member Functions

- `_RestrictedBoundedConcurrentQueue` (`_SequenceIndex __max_size`)
- `~_RestrictedBoundedConcurrentQueue` ()
- `bool pop_back` (`_Tp &__t`)
- `bool pop_front` (`_Tp &__t`)
- `void push_front` (`const _Tp &__t`)

### 5.157.1 Detailed Description

`template<typename _Tp> class __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

#### Parameters

`_Tp` Contained element type.

Definition at line 52 of file `queue.h`.

## 5.157.2 Constructor & Destructor Documentation

**5.157.2.1** `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::_RestrictedBoundedConcurrentQueue ( _SequenceIndex _max_size ) [inline]`

Constructor. Not to be called concurrent, of course.

### Parameters

*M\_max\_size* Maximal number of elements to be contained.

Definition at line 68 of file `queue.h`.

**5.157.2.2** `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::~~_RestrictedBoundedConcurrentQueue ( ) [inline]`

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file `queue.h`.

## 5.157.3 Member Function Documentation

**5.157.3.1** `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back ( _Tp & t ) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop\\_front\(\)](#).

Definition at line 127 of file `queue.h`.

**5.157.3.2** `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_front ( _Tp & t ) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop\\_front\(\)](#).

Definition at line 100 of file `queue.h`.

Referenced by `__gnu_parallel::_qsb_local_sort_with_helping()`.

### 5.157.3.3 `template<typename _Tp> void __gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>::push_front ( const _Tp & __t ) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with `pop_front()`.

Definition at line 83 of file `queue.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this class was generated from the following file:

- [queue.h](#)

## 5.158 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference`

Stable sorting functor.

### Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

### 5.158.1 Detailed Description

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1004 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.159** `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference 1147

---

## **5.159** `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Non-`__stable` sorting functor.

### **Public Member Functions**

- void `operator()` (`_RAIter __first`, `_RAIter __last`, `_StrictWeakOrdering __comp`)

#### **5.159.1 Detailed Description**

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1017 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## **5.160** `__gnu_parallel::_Settings` Struct Reference

class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

### **Static Public Member Functions**

- static `_GLIBCXX_CONST` const `_Settings` & `get` () throw ()
- static void `set` (`_Settings` &) throw ()

### **Public Attributes**

- `_SequenceIndex` `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- `_AlgorithmStrategy` `algorithm_strategy`

- unsigned int `cache_line_size`
- `_SequenceIndex` `count_minimal_n`
- `_SequenceIndex` `fill_minimal_n`
- `_FindAlgorithm` **`find_algorithm`**
- double `find_increasing_factor`
- `_SequenceIndex` `find_initial_block_size`
- `_SequenceIndex` `find_maximum_block_size`
- `_SequenceIndex` `find_sequential_search_size`
- `_SequenceIndex` `for_each_minimal_n`
- `_SequenceIndex` `generate_minimal_n`
- unsigned long long `L1_cache_size`
- unsigned long long `L2_cache_size`
- `_SequenceIndex` `max_element_minimal_n`
- `_SequenceIndex` `merge_minimal_n`
- unsigned int `merge_oversampling`
- `_SplittingAlgorithm` **`merge_splitting`**
- `_SequenceIndex` `min_element_minimal_n`
- `_MultiwayMergeAlgorithm` **`multiway_merge_algorithm`**
- int `multiway_merge_minimal_k`
- `_SequenceIndex` `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- `_SplittingAlgorithm` **`multiway_merge_splitting`**
- `_SequenceIndex` `nth_element_minimal_n`
- `_SequenceIndex` `partial_sort_minimal_n`
- `_PartialSumAlgorithm` **`partial_sum_algorithm`**
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- `_SequenceIndex` `partition_chunk_size`
- `_SequenceIndex` `partition_minimal_n`
- `_SequenceIndex` `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- `_SequenceIndex` `replace_minimal_n`
- `_SequenceIndex` `search_minimal_n`
- `_SequenceIndex` `set_difference_minimal_n`
- `_SequenceIndex` `set_intersection_minimal_n`
- `_SequenceIndex` `set_symmetric_difference_minimal_n`
- `_SequenceIndex` `set_union_minimal_n`
- `_SortAlgorithm` **`sort_algorithm`**
- `_SequenceIndex` `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`
- unsigned int `sort_qs_num_samples_preset`
- `_SequenceIndex` `sort_qsb_base_case_maximal_n`

- [\\_SplittingAlgorithm](#) `sort_splitting`
- unsigned int `TLB_size`
- [\\_SequenceIndex](#) `transform_minimal_n`
- [\\_SequenceIndex](#) `unique_copy_minimal_n`
- [\\_SequenceIndex](#) `workstealing_chunk_size`

### 5.160.1 Detailed Description

class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

### 5.160.2 Member Function Documentation

#### 5.160.2.1 `static _GLIBCXX_CONST const _Settings& __gnu_parallel::_Settings::get( ) throw () [static]`

Get the global settings.

Referenced by `__gnu_parallel::_find_template()`, `__gnu_parallel::_for_each_template_random_access_workstealing()`, `__gnu_parallel::_parallel_nth_element()`, `__gnu_parallel::_parallel_partial_sum()`, `__gnu_parallel::_parallel_partial_sum_linear()`, `__gnu_parallel::_parallel_partition()`, `__gnu_parallel::_parallel_sort()`, `__gnu_parallel::_parallel_sort_qs_conquer()`, `__gnu_parallel::_qsb_local_sort_with_helping()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.160.2.2 `static void __gnu_parallel::_Settings::set ( _Settings & ) throw () [static]`

Set the global settings.

### 5.160.3 Member Data Documentation

#### 5.160.3.1 `_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n`

Minimal input size for accumulate.

Definition at line 139 of file `settings.h`.

**5.160.3.2 unsigned int \_\_gnu\_parallel::\_Settings::adjacent\_difference\_minimal\_n**

Minimal input size for adjacent\_difference.

Definition at line 142 of file settings.h.

**5.160.3.3 unsigned int \_\_gnu\_parallel::\_Settings::cache\_line\_size**

Overestimation of cache line size. Used to avoid false /// sharing, i.e. elements of different threads are at least this /// amount apart.

Definition at line 265 of file settings.h.

Referenced by \_\_gnu\_parallel::\_for\_each\_template\_random\_access\_workstealing().

**5.160.3.4 \_SequenceIndex \_\_gnu\_parallel::\_Settings::count\_minimal\_n**

Minimal input size for count and count\_if.

Definition at line 145 of file settings.h.

**5.160.3.5 \_SequenceIndex \_\_gnu\_parallel::\_Settings::fill\_minimal\_n**

Minimal input size for fill.

Definition at line 148 of file settings.h.

**5.160.3.6 double \_\_gnu\_parallel::\_Settings::find\_increasing\_factor**

Block size increase factor for find.

Definition at line 151 of file settings.h.

Referenced by \_\_gnu\_parallel::\_find\_template().

**5.160.3.7 \_SequenceIndex \_\_gnu\_parallel::\_Settings::find\_initial\_block\_size**

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by \_\_gnu\_parallel::\_find\_template().

**5.160.3.8** `_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size`

Maximal block size for `find`.

Definition at line 157 of file `settings.h`.

Referenced by `__gnu_parallel::_find_template()`.

**5.160.3.9** `_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size`

Start with looking for this many elements sequentially, for `find`.

Definition at line 160 of file `settings.h`.

Referenced by `__gnu_parallel::_find_template()`.

**5.160.3.10** `_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n`

Minimal input size for `for_each`.

Definition at line 163 of file `settings.h`.

**5.160.3.11** `_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for `generate`.

Definition at line 166 of file `settings.h`.

**5.160.3.12** `unsigned long long __gnu_parallel::_Settings::L1_cache_size`

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file `settings.h`.

**5.160.3.13** `unsigned long long __gnu_parallel::_Settings::L2_cache_size`

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_sequential_random_shuffle()`.

**5.160.3.14** `_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for `max_element`.

Definition at line 169 of file `settings.h`.

**5.160.3.15** `_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for `merge`.

Definition at line 172 of file `settings.h`.

**5.160.3.16** `unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for `merge`.

Definition at line 175 of file `settings.h`.

Referenced by `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

**5.160.3.17** `_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for `min_element`.

Definition at line 178 of file `settings.h`.

**5.160.3.18** `int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for `multiway_merge`.

Definition at line 184 of file `settings.h`.

**5.160.3.19** `_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n`

Minimal input size for `multiway_merge`.

Definition at line 181 of file `settings.h`.

**5.160.3.20** `unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`

Oversampling factor for `multiway_merge`.

Definition at line 187 of file `settings.h`.

**5.160.3.21** `_SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n`

Minimal input size for `nth_element`.

Definition at line 190 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_nth_element()`.

**5.160.3.22** `_SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n`

Minimal input size for `partial_sort`.

Definition at line 203 of file `settings.h`.

**5.160.3.23** `float __gnu_parallel::_Settings::partial_sum_dilation`

Ratio for `partial_sum`. Assume "sum and write result" to be `///` this factor slower than just "sum".

Definition at line 207 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_partial_sum_linear()`.

**5.160.3.24** `unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`

Minimal input size for `partial_sum`.

Definition at line 210 of file `settings.h`.

**5.160.3.25** `double __gnu_parallel::_Settings::partition_chunk_share`

Chunk size for `partition`, relative to input size. If `> 0.0`, `///` this value overrides `partition_chunk_size`.

Definition at line 197 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_partition()`.

**5.160.3.26** `_SequenceIndex __gnu_parallel::_Settings::partition_chunk_size`

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by `__gnu_parallel::_parallel_partition()`.

**5.160.3.27** `_SequenceIndex __gnu_parallel::_Settings::partition_minimal_n`

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by `__gnu_parallel::_parallel_nth_element()`.

**5.160.3.28** `_SequenceIndex __gnu_parallel::_Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

**5.160.3.29** `unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n`

Minimal input size for `random_shuffle`.

Definition at line 213 of file settings.h.

**5.160.3.30** `_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Definition at line 216 of file settings.h.

**5.160.3.31** `_SequenceIndex __gnu_parallel::_Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Definition at line 273 of file settings.h.

**5.160.3.32** `_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file settings.h.

**5.160.3.33** `_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

**5.160.3.34** `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

**5.160.3.35** `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

**5.160.3.36** `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

**5.160.3.37** `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

**5.160.3.38** `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

### 5.160.3.39 `_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. */// Applies to `std::sort` with dynamically load-balanced quicksort.*

Definition at line 241 of file `settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 5.160.3.40 `unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

### 5.160.3.41 `_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Definition at line 244 of file `settings.h`.

### 5.160.3.42 `_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Definition at line 247 of file `settings.h`.

The documentation for this struct was generated from the following file:

- [settings.h](#)

## 5.161 `__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Split consistently.

### 5.161.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< __exact, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.162 `__gnu_parallel::_SplitConsistently`< false, `_RAIter`, `_Compare`, `_SortingPlacesIterator` > Struct Template Reference

Split by sampling.

### Public Member Functions

- void `operator()` (const `_ThreadIndex` `__iam`, `_PMWMSortingData`< `_RAIter` > `*__sd`, `_Compare` & `__comp`, const typename `std::iterator_traits`< `_RAIter` >::`difference_type` `__num_samples`) const

### 5.162.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< false,
_RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.163 `__gnu_parallel::_SplitConsistently`< `true`, `_RAIter`, `_Compare`, `_SortingPlacesIterator` > Struct Template Reference

Split by exact splitting.

#### Public Member Functions

- void `operator()` (const `_ThreadIndex` `__iam`, `_PMWMSortingData`< `_RAIter` > \*`__sd`, `_Compare` &`__comp`, const typename `std::iterator_traits`< `_RAIter` >::`difference_type` `__num_samples`) const

#### 5.163.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< true, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

Definition at line 128 of file `multiway_mergesort.h`.

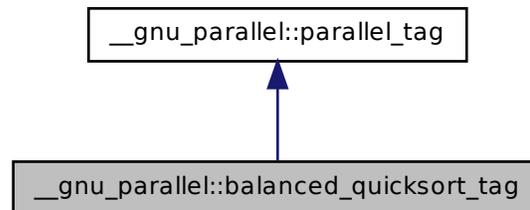
The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.164 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Forces parallel sorting using balanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



## Public Member Functions

- `balanced_quicksort_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 5.164.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

### 5.164.2 Member Function Documentation

#### 5.164.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

### 5.164.2.2 void \_\_gnu\_parallel::parallel\_tag::set\_num\_threads ( \_ThreadIndex \_\_num\_threads ) [inline, inherited]

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

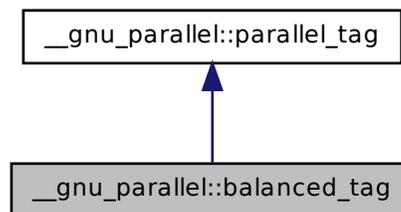
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.165 \_\_gnu\_parallel::balanced\_tag Struct Reference

Recommends parallel execution using dynamic load-balancing at compile time.

Inheritance diagram for \_\_gnu\_parallel::balanced\_tag:



### Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

#### 5.165.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

## 5.165.2 Member Function Documentation

### 5.165.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.165.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

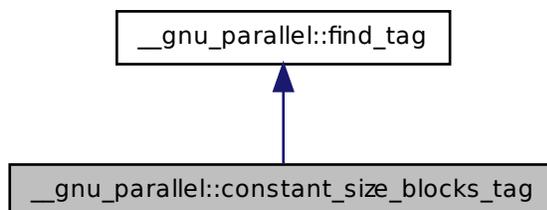
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.166 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Selects the constant block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



### 5.166.1 Detailed Description

Selects the constant block size variant for `std::find()`.

#### See also

[\\_GLIBCXX\\_FIND\\_CONSTANT\\_SIZE\\_BLOCKS](#)

Definition at line 178 of file `tags.h`.

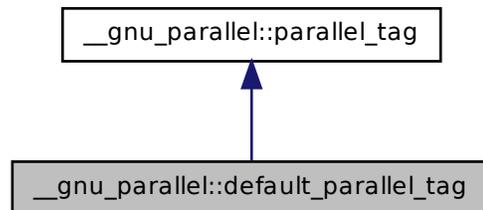
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.167 `__gnu_parallel::default_parallel_tag` Struct Reference

Recommends parallel execution using the default parallel algorithm.

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



## Public Member Functions

- `default_parallel_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 5.167.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file tags.h.

### 5.167.2 Member Function Documentation

#### 5.167.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

**5.167.2.2** `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex  
__num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

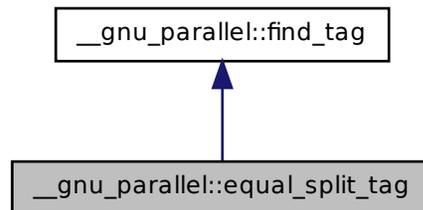
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.168 `__gnu_parallel::equal_split_tag` Struct Reference

Selects the equal splitting variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



### 5.168.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

#### See also

[\\_GLIBCXX\\_FIND\\_EQUAL\\_SPLIT](#)

Definition at line 182 of file tags.h.

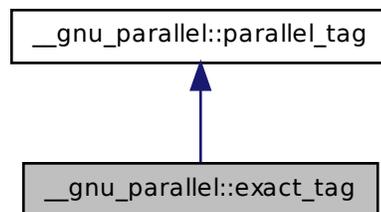
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.169 `__gnu_parallel::exact_tag` Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for `__gnu_parallel::exact_tag`:



### Public Member Functions

- `exact_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

#### 5.169.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

## 5.169.2 Member Function Documentation

### 5.169.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.169.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

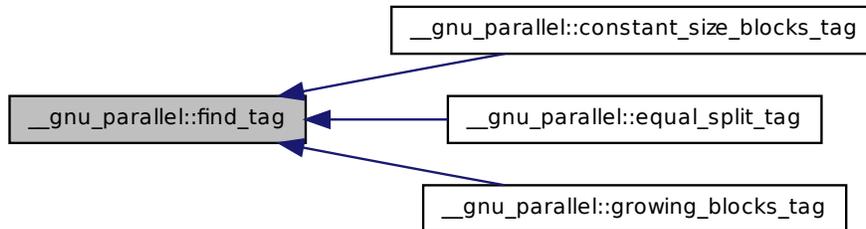
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.170 `__gnu_parallel::find_tag` Struct Reference

Base class for for `std::find()` variants.

Inheritance diagram for `__gnu_parallel::find_tag`:



### 5.170.1 Detailed Description

Base class for for `std::find()` variants.

Definition at line 104 of file `tags.h`.

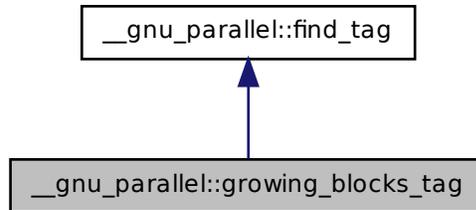
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.171 `__gnu_parallel::growing_blocks_tag` Struct Reference

Selects the growing block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



### 5.171.1 Detailed Description

Selects the growing block size variant for `std::find()`.

**See also**

[\\_GLIBCXX\\_FIND\\_GROWING\\_BLOCKS](#)

Definition at line 174 of file `tags.h`.

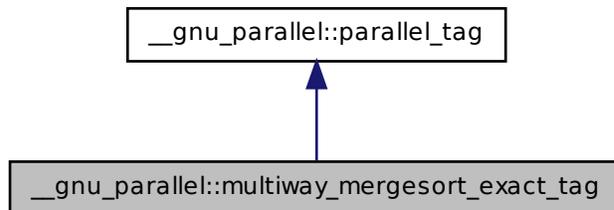
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.172 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



## Public Member Functions

- `multiway_mergesort_exact_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 5.172.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

### 5.172.2 Member Function Documentation

#### 5.172.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.172.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

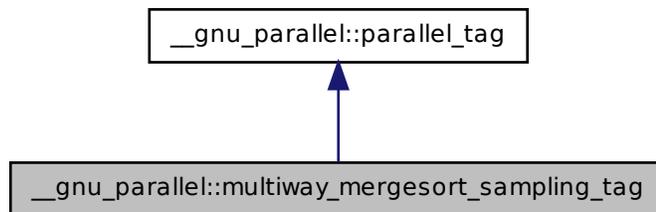
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.173 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



#### Public Member Functions

- `multiway_mergesort_sampling_tag ( _ThreadIndex __num_threads )`
- `_ThreadIndex __get_num_threads ( )`
- `void set_num_threads ( _ThreadIndex __num_threads )`

### 5.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

### 5.173.2 Member Function Documentation

#### 5.173.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.173.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

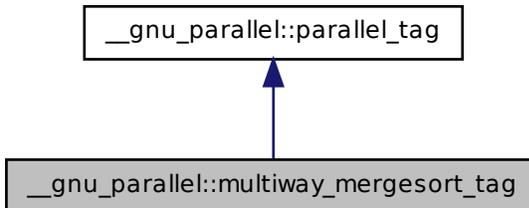
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.174 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Forces parallel sorting using multiway mergesort at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



## Public Member Functions

- `multiway_mergesort_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 5.174.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

### 5.174.2 Member Function Documentation

#### 5.174.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.174.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

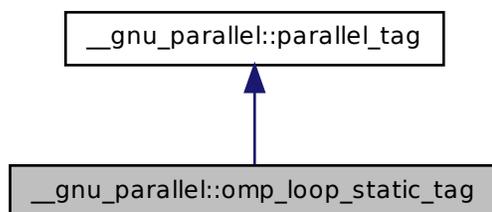
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.175 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Recommends parallel execution using OpenMP static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



#### Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

### 5.175.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.  
Definition at line 100 of file tags.h.

### 5.175.2 Member Function Documentation

#### 5.175.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.175.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

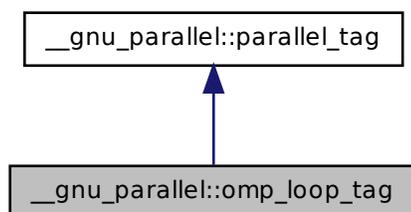
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.176 `__gnu_parallel::omp_loop_tag` Struct Reference

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:



## Public Member Functions

- `_ThreadIndex __get_num_threads()`
- `void set_num_threads(_ThreadIndex __num_threads)`

### 5.176.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file `tags.h`.

### 5.176.2 Member Function Documentation

#### 5.176.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

**5.176.2.2** `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex  
__num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

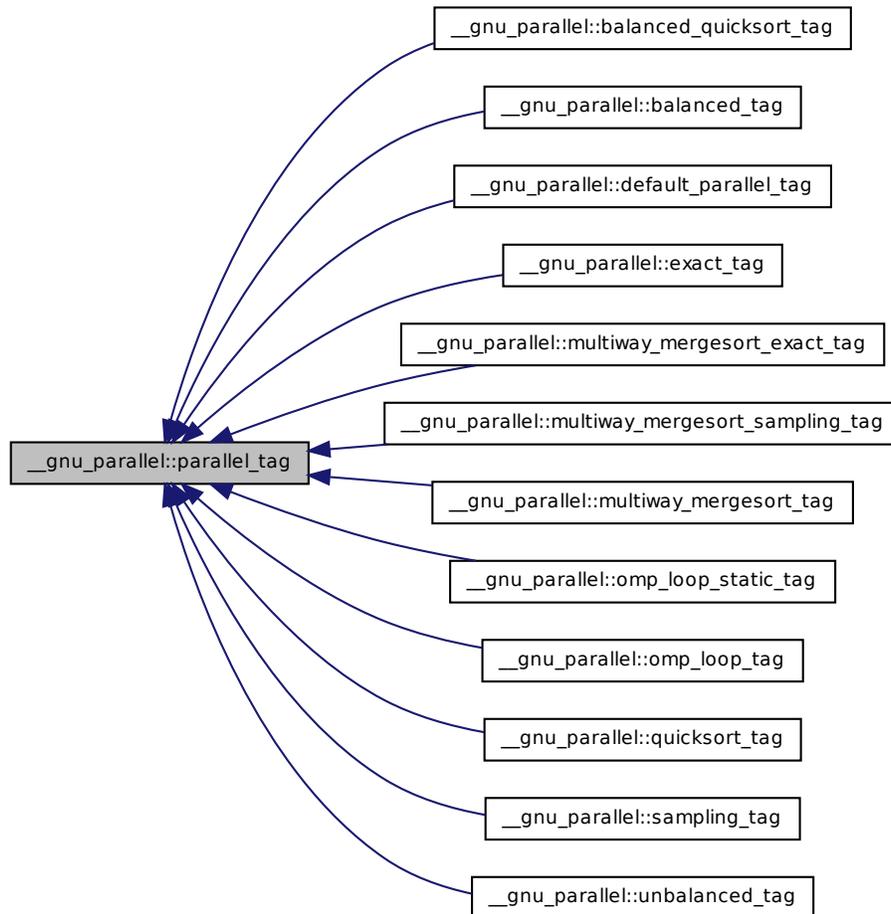
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.177 `__gnu_parallel::parallel_tag` Struct Reference

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:



## Public Member Functions

- [parallel\\_tag](#) ()
- [parallel\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.177.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

### 5.177.2 Constructor & Destructor Documentation

#### 5.177.2.1 `__gnu_parallel::parallel_tag::parallel_tag ( ) [inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

#### 5.177.2.2 `__gnu_parallel::parallel_tag::parallel_tag ( _ThreadIndex __num_threads ) [inline]`

Default constructor. Recommend number of threads to use.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 58 of file tags.h.

### 5.177.3 Member Function Documentation

#### 5.177.3.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline]`

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.177.3.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline]`

Set the desired number of threads.

**Parameters**

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

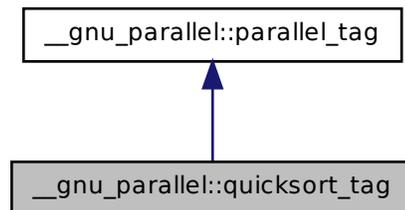
The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.178 `__gnu_parallel::quicksort_tag` Struct Reference**

Forces parallel sorting using unbalanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::quicksort_tag`:

**Public Member Functions**

- `quicksort_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

**5.178.1 Detailed Description**

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

## 5.178.2 Member Function Documentation

### 5.178.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.178.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

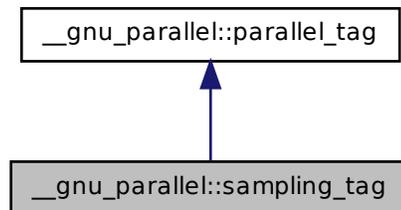
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.179 `__gnu_parallel::sampling_tag` Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for `__gnu_parallel::sampling_tag`:



## Public Member Functions

- `sampling_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

### 5.179.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file `tags.h`.

### 5.179.2 Member Function Documentation

#### 5.179.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [`inline`, `inherited`]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

**5.179.2.2** `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex  
__num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.180 `__gnu_parallel::sequential_tag` Struct Reference

Forces sequential execution at compile time.

### 5.180.1 Detailed Description

Forces sequential execution at compile time.

Definition at line 42 of file tags.h.

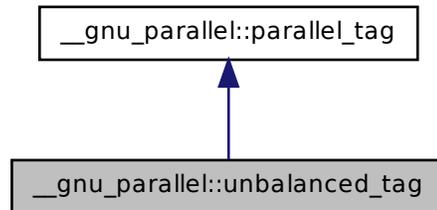
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.181 `__gnu_parallel::unbalanced_tag` Struct Reference

Recommends parallel execution using static load-balancing at compile time.

Inheritance diagram for \_\_gnu\_parallel::unbalanced\_tag:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \(\)](#)
- void [set\\_num\\_threads \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

### 5.181.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

### 5.181.2 Member Function Documentation

#### 5.181.2.1 [\\_ThreadIndex \\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads \( \)](#) [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.181.2.2 void \_\_gnu\_parallel::parallel\_tag::set\_num\_threads ( \_ThreadIndex \_\_num\_threads ) [inline, inherited]

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

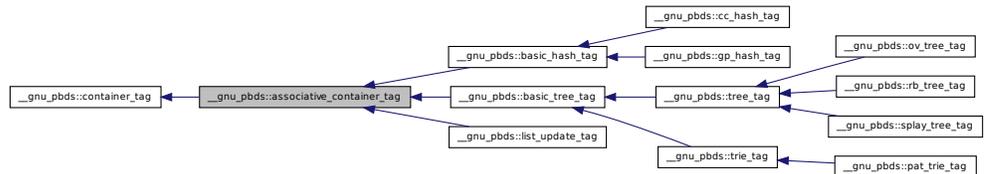
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.182 \_\_gnu\_pbds::associative\_container\_tag Struct Reference

Basic associative-container.

Inheritance diagram for \_\_gnu\_pbds::associative\_container\_tag:



### 5.182.1 Detailed Description

Basic associative-container.

Definition at line 102 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.183 `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator > Class Template

Reference ~~5.183~~ `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator > Class Template Reference <sup>1185</sup>

An abstract basic hash-based associative container.

Inheritance diagram for `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator >:



## Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.183.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_
Fn, typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_
TL, typename Allocator> class __gnu_pbds::basic_hash_table< Key, Mapped,
Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >
```

An abstract basic hash-based associative container.

Definition at line 143 of file `assoc_container.hpp`.

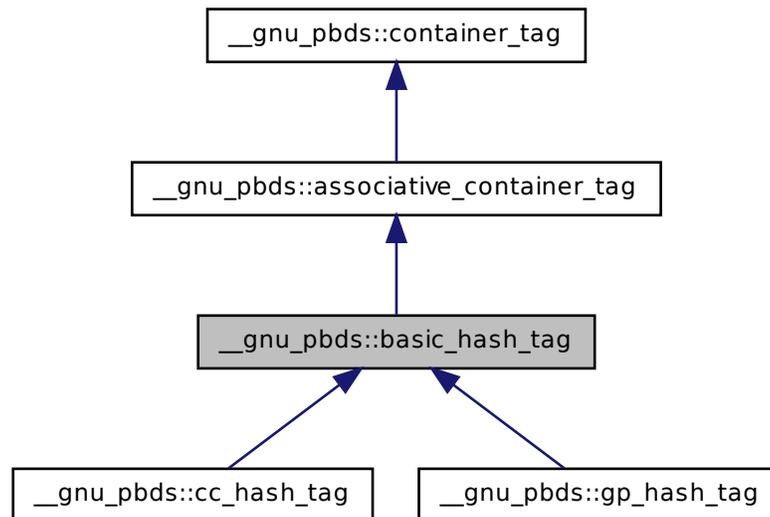
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.184 `__gnu_pbds::basic_hash_tag` Struct Reference

Basic hash.

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



### 5.184.1 Detailed Description

Basic hash.

Definition at line 105 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.185 `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >` Class Template Reference

An abstract basic tree-like (tree, trie) associative container.

Inheritance diagram for `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`:



### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**

- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef Node\_Update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.185.1 Detailed Description

**template<typename Key, typename Mapped, typename Tag, typename Node\_Update, typename Policy\_Tl, typename Allocator> class `__gnu_pbds::basic_tree`< Key, Mapped, Tag, Node\_Update, Policy\_Tl, Allocator >**

An abstract basic tree-like (tree, trie) associative container.

Definition at line 476 of file `assoc_container.hpp`.

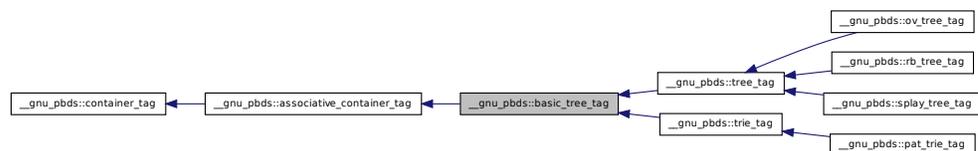
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.186 `__gnu_pbds::basic_tree_tag` Struct Reference

Basic tree.

Inheritance diagram for `__gnu_pbds::basic_tree_tag`:



### 5.186.1 Detailed Description

Basic tree.

Definition at line 114 of file `tag_and_trait.hpp`.

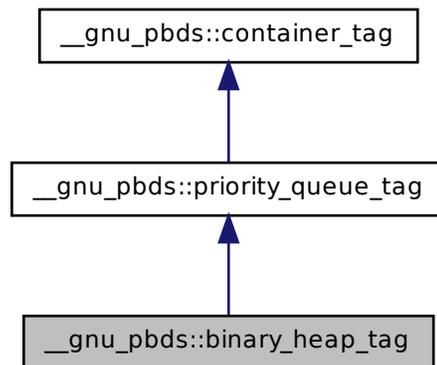
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.187 `__gnu_pbds::binary_heap_tag` Struct Reference

Binary-heap (array-based).

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



### 5.187.1 Detailed Description

Binary-heap (array-based).

Definition at line 150 of file `tag_and_trait.hpp`.

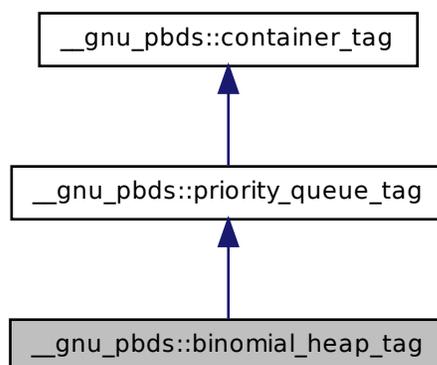
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.188 `__gnu_pbds::binomial_heap_tag` Struct Reference

Binomial-heap.

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



### 5.188.1 Detailed Description

Binomial-heap.

Definition at line 144 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.189 `__gnu_pbds::cc_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`,  
`Comb_Hash_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` > Class Template

Reference  
5.189 `__gnu_pbds::cc_hash_table`< `Key`, `Mapped`,  
`Hash_Fn`, `Eq_Fn`, `Comb_Hash_Fn`, `Resize_`  
`Policy`, `Store_Hash`, `Allocator` > Class Template  
Reference

A concrete collision-chaining hash-based associative container.

Inheritance diagram for `__gnu_pbds::cc_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`,  
`Comb_Hash_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` >:



## Public Types

- typedef Allocator **allocator\_type**
- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef `cc_hash_tag` **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_**  
**rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**

- typedef Resize\_Policy **resize\_policy**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- **cc\_hash\_table** (const hash\_fn &h)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It >  
**cc\_hash\_table** (It first, It last, const hash\_fn &h, const eq\_fn &e)
- **cc\_hash\_table** (const [cc\\_hash\\_table](#) &other)
- template<typename It >  
**cc\_hash\_table** (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It >  
**cc\_hash\_table** (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It >  
**cc\_hash\_table** (It first, It last, const hash\_fn &h)
- template<typename It >  
**cc\_hash\_table** (It first, It last)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e)
- [cc\\_hash\\_table](#) & **operator=** (const [cc\\_hash\\_table](#) &other)
- void **swap** ([cc\\_hash\\_table](#) &other)

### 5.189.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = typename
detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_
comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename Allocator = std::allocator<char>> class __gnu_pbds::cc_hash_table<
Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Al-
locator >
```

A concrete collision-chaining hash-based associative container.

Definition at line 179 of file `assoc_container.hpp`.

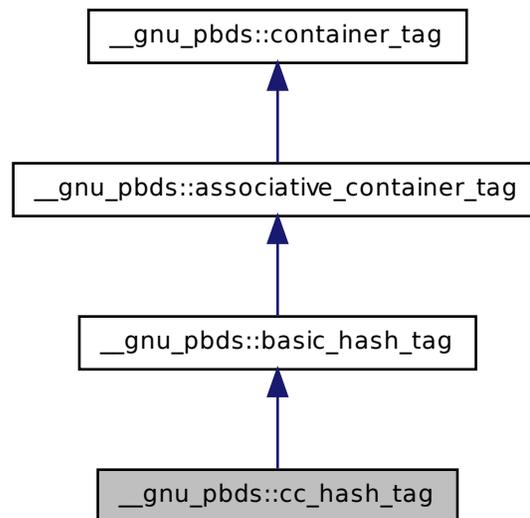
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.190 `__gnu_pbds::cc_hash_tag` Struct Reference

Collision-chaining hash.

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



### 5.190.1 Detailed Description

Collision-chaining hash.

Definition at line 108 of file `tag_and_trait.hpp`.

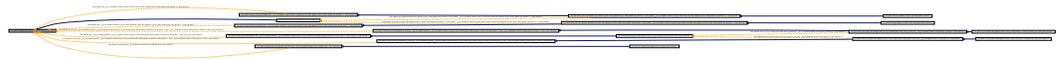
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.191 `__gnu_pbds::container_base`< `Key`, `Mapped`, `Tag`, `Policy_Tl`, `Allocator` > Class Template Reference

An abstract basic associative container.

Inheritance diagram for `__gnu_pbds::container_base`< `Key`, `Mapped`, `Tag`, `Policy_Tl`, `Allocator` >:



### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.191.1 Detailed Description

```
template<typename Key, typename Mapped, typename Tag, typename Policy_
-TI, typename Allocator> class __gnu_pbds::container_base< Key, Mapped, Tag,
Policy_TI, Allocator >
```

An abstract basic associative container.

Definition at line 76 of file `assoc_container.hpp`.

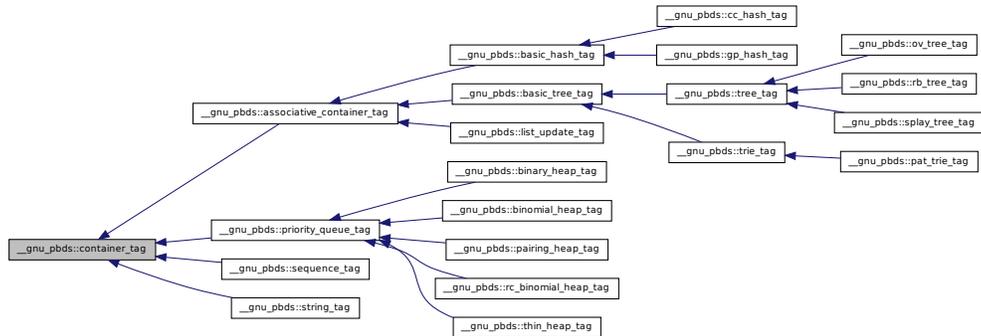
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.192 `__gnu_pbds::container_tag` Struct Reference

Base data structure tag.

Inheritance diagram for `__gnu_pbds::container_tag`:



### 5.192.1 Detailed Description

Base data structure tag.

Definition at line 92 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.193 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

[container\\_traits](#)

Inherits `container_traits_base< Cntnr::container_category >`.

### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `container_traits_base< container_category >` **base\_type**
- typedef `Cntnr::container_category` **container\_category**
- typedef `Cntnr` **container\_type**
- typedef `base_type::invalidation_guarantee` **invalidation\_guarantee**

### 5.193.1 Detailed Description

`template<typename Cntnr> struct __gnu_pbds::container_traits< Cntnr >`

[container\\_traits](#)

Definition at line 345 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.194 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >` Struct Template Reference

### Public Types

- typedef `mapped_type_allocator::const_pointer` **const\_mapped\_pointer**
- typedef `mapped_type_allocator::const_reference` **const\_mapped\_reference**
- typedef `value_type_allocator::const_pointer` **const\_pointer**
- typedef `value_type_allocator::const_reference` **const\_reference**
- typedef `mapped_type_allocator::pointer` **mapped\_pointer**
- typedef `mapped_type_allocator::reference` **mapped\_reference**
- typedef `mapped_type_allocator::value_type` **mapped\_type**

## 5.195 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >` Struct Template Reference 1197

- `typedef Allocator::template rebind< Mapped >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef value_type_allocator::value_type value_type`
- `typedef Allocator::template rebind< std::pair< const Key, Mapped > >::other value_type_allocator`

### 5.194.1 Detailed Description

`template<typename Key, typename Mapped, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 61 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

## 5.195 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >` Struct Template Reference

### Public Types

- `typedef mapped_type_allocator::const_pointer const_mapped_pointer`
- `typedef mapped_type_allocator::const_reference const_mapped_reference`
- `typedef value_type_allocator::const_pointer const_pointer`
- `typedef value_type_allocator::const_reference const_reference`
- `typedef mapped_type_allocator::pointer mapped_pointer`
- `typedef mapped_type_allocator::reference mapped_reference`
- `typedef mapped_type_allocator::value_type mapped_type`
- `typedef Allocator::template rebind< Mapped >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef value_type_allocator::value_type value_type`
- `typedef Allocator::template rebind< std::pair< const Key, Mapped > >::other value_type_allocator`

### 5.195.1 Detailed Description

```
template<typename Key, typename Mapped, typename Allocator> struct __-
gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >
```

Specialization of `value_type_base` for the case where the hash value is stored alongside each value.

Definition at line 88 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

## 5.196 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >` Struct Template Reference

### Public Types

- typedef `mapped_type_allocator::const_pointer` **const\_mapped\_pointer**
- typedef `mapped_type_allocator::const_reference` **const\_mapped\_reference**
- typedef `value_type_allocator::const_pointer` **const\_pointer**
- typedef `value_type_allocator::const_reference` **const\_reference**
- typedef `mapped_type_allocator::pointer` **mapped\_pointer**
- typedef `mapped_type_allocator::reference` **mapped\_reference**
- typedef `mapped_type_allocator::value_type` **mapped\_type**
- typedef `Allocator::template rebind< null_mapped_type >::other` **mapped\_type\_allocator**
- typedef `value_type_allocator::pointer` **pointer**
- typedef `value_type_allocator::reference` **reference**
- typedef `Key` **value\_type**
- typedef `Allocator::template rebind< value_type >::other` **value\_type\_allocator**

### Static Public Attributes

- static `null_mapped_type` **s\_null\_mapped**

### 5.196.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >
```

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 122 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

## 5.197 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >` Struct Template Reference

### Public Types

- typedef `mapped_type_allocator::const_pointer` **const\_mapped\_pointer**
- typedef `mapped_type_allocator::const_reference` **const\_mapped\_reference**
- typedef `value_type_allocator::const_pointer` **const\_pointer**
- typedef `value_type_allocator::const_reference` **const\_reference**
- typedef `mapped_type_allocator::pointer` **mapped\_pointer**
- typedef `mapped_type_allocator::reference` **mapped\_reference**
- typedef `mapped_type_allocator::value_type` **mapped\_type**
- typedef `Allocator::template rebind< null_mapped_type >::other` **mapped\_type\_allocator**
- typedef `value_type_allocator::pointer` **pointer**
- typedef `value_type_allocator::reference` **reference**
- typedef `Key` **value\_type**
- typedef `Allocator::template rebind< value_type >::other` **value\_type\_allocator**

### Static Public Attributes

- static `null_mapped_type` **s\_null\_mapped**

### 5.197.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbd::detail::value_type_base< Key, null_mapped_type, Allocator, true >
```

Specialization of value\_type\_base for the case where the hash value is stored alongside each value.

Definition at line 164 of file basic\_types.hpp.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

## 5.198 \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, Allocator > Class Template Reference

A concrete general-probing hash-based associative container.

Inheritance diagram for \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, Allocator >:



### Public Types

- typedef Allocator **allocator\_type**
- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**

**5.198 `__gnu_pbds::gp_hash_table`< `Key`, `Mapped`, `Hash_Fn`, `Eq_Fn`, `Comb_Probe_Fn`, `Probe_Fn`, `Resize_Policy`, `Store_Hash`, `Allocator` > Class**

**Template Reference**

1201

- `typedef Hash_Fn hash_fn`
- `typedef base_type::iterator iterator`
- `typedef key_rebind::pointer key_pointer`
- `typedef allocator_type::template rebind< key_type >::other key_rebind`
- `typedef key_rebind::reference key_reference`
- `typedef allocator_type::template rebind< Key >::other::value_type key_type`
- `typedef mapped_rebind::pointer mapped_pointer`
- `typedef allocator_type::template rebind< mapped_type >::other mapped_rebind`
- `typedef mapped_rebind::reference mapped_reference`
- `typedef Mapped mapped_type`
- `typedef base_type::point_iterator point_iterator`
- `typedef value_rebind::pointer pointer`
- `typedef Probe_Fn probe_fn`
- `typedef value_rebind::reference reference`
- `typedef Resize_Policy resize_policy`
- `typedef allocator_type::size_type size_type`
- `typedef allocator_type::template rebind< value_type >::other value_rebind`
- `typedef base_type::value_type value_type`

**Public Member Functions**

- `gp_hash_table (const hash_fn &h)`
- `gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)`
- `template<typename It >`  
`gp_hash_table (It first, It last, const hash_fn &h)`
- `gp_hash_table (const gp\_hash\_table &other)`
- `template<typename It >`  
`gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)`
- `template<typename It >`  
`gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)`
- `template<typename It >`  
`gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)`
- `template<typename It >`  
`gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)`
- `gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)`
- `template<typename It >`  
`gp_hash_table (It first, It last)`

- **gp\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- **gp\_hash\_table** (const hash\_fn &h, const eq\_fn &e)
- **gp\_hash\_table** & **operator=** (const [gp\\_hash\\_table](#) &other)
- void **swap** ([gp\\_hash\\_table](#) &other)

### 5.198.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = type-
name detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename
detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy
= typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_
Hash = detail::default_store_hash, typename Allocator = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_
Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >
```

A concrete general-probing hash-based associative container.

Definition at line 317 of file `assoc_container.hpp`.

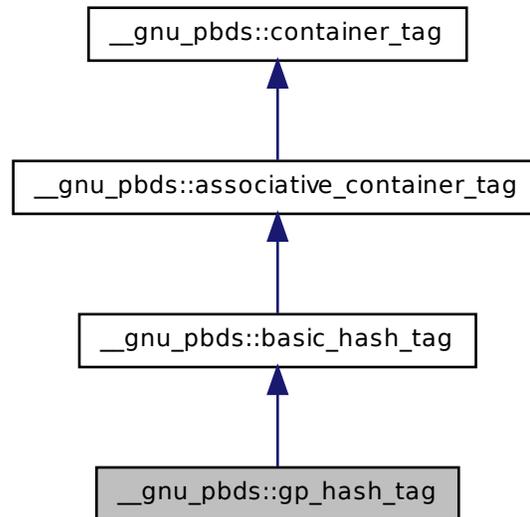
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.199 `__gnu_pbds::gp_hash_tag` Struct Reference

General-probing hash.

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



### 5.199.1 Detailed Description

General-probing hash.

Definition at line 111 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.200 `__gnu_pbds::list_update`< Key, Mapped, Eq\_Fn, Update\_Policy, Allocator > Class Template Reference

A list-update based associative container.

Inheritance diagram for `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`:



## Public Types

- typedef Allocator **allocator**
- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef [list\\_update\\_tag](#) **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef Update\_Policy **update\_policy**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- `template<typename It >`  
`list_update` (It first, It last)
- `list_update` (const `list_update` &other)
- `list_update` & `operator=` (const `list_update` &other)
- void `swap` (`list_update` &other)

### 5.200.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename
detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_
update_policy::type, class Allocator = std::allocator<char>> class __gnu_
pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >
```

A list-update based associative container.

Definition at line 653 of file `assoc_container.hpp`.

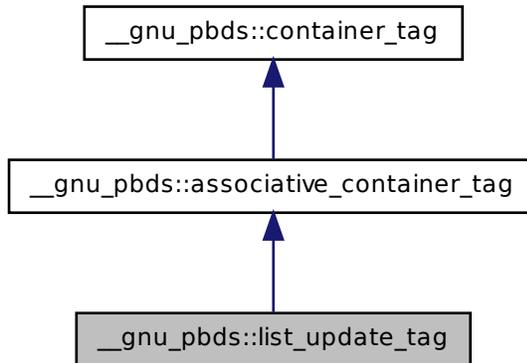
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.201 `__gnu_pbds::list_update_tag` Struct Reference

List-update.

Inheritance diagram for `__gnu_pbds::list_update_tag`:



### 5.201.1 Detailed Description

List-update.

Definition at line 135 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.202 `__gnu_pbds::null_mapped_type` Struct Reference

A mapped-policy indicating that an associative container is a set.

### 5.202.1 Detailed Description

A mapped-policy indicating that an associative container is a set.

Definition at line 88 of file `tag_and_trait.hpp`.

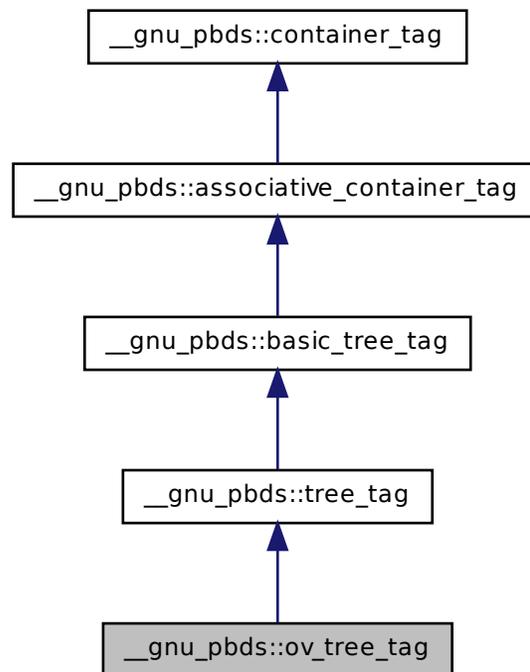
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.203 `__gnu_pbds::ov_tree_tag` Struct Reference

Ordered-vector tree.

Inheritance diagram for `__gnu_pbds::ov_tree_tag`:



### 5.203.1 Detailed Description

Ordered-vector tree.

Definition at line 126 of file `tag_and_trait.hpp`.

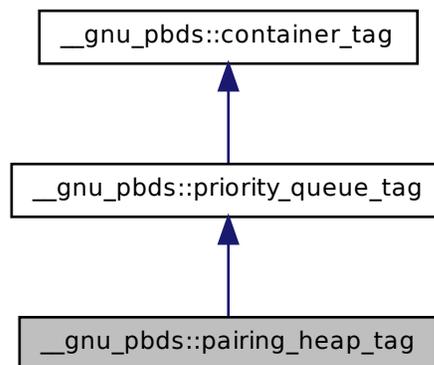
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.204 `__gnu_pbds::pairing_heap_tag` Struct Reference

Pairing-heap.

Inheritance diagram for `__gnu_pbds::pairing_heap_tag`:



### 5.204.1 Detailed Description

Pairing-heap.

Definition at line 141 of file `tag_and_trait.hpp`.

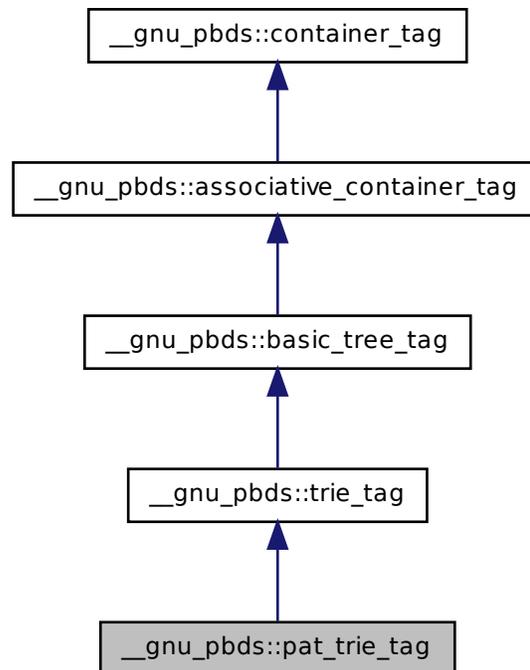
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.205 `__gnu_pbds::pat_trie_tag` Struct Reference

PATRICIA trie.

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



### 5.205.1 Detailed Description

PATRICIA trie.

Definition at line 132 of file `tag_and_trait.hpp`.

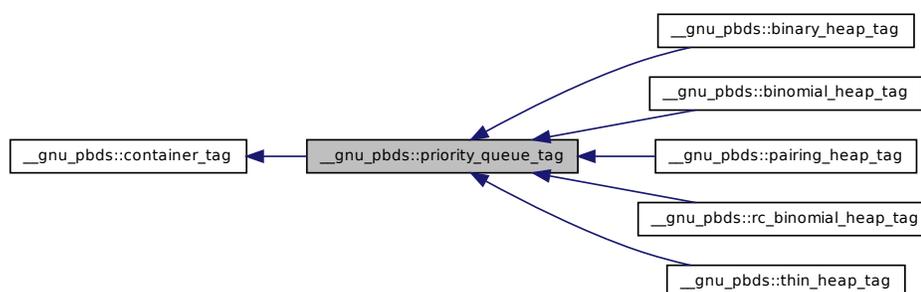
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.206 `__gnu_pbds::priority_queue_tag` Struct Reference

Basic priority-queue.

Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



### 5.206.1 Detailed Description

Basic priority-queue.

Definition at line 138 of file `tag_and_trait.hpp`.

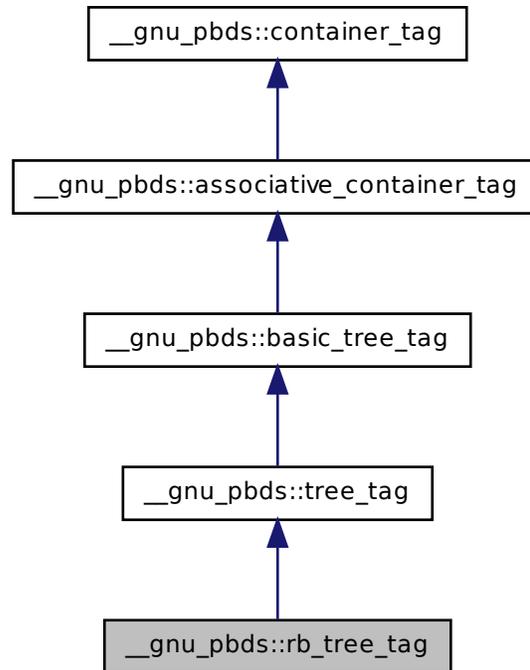
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.207 `__gnu_pbds::rb_tree_tag` Struct Reference

Red-black tree.

Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



### 5.207.1 Detailed Description

Red-black tree.

Definition at line 120 of file `tag_and_trait.hpp`.

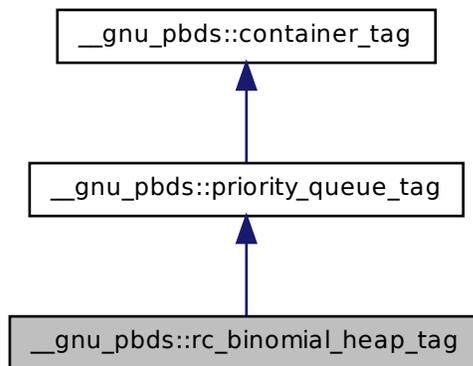
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.208 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Redundant-counter binomial-heap.

Inheritance diagram for `__gnu_pbds::rc_binomial_heap_tag`:



### 5.208.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 147 of file `tag_and_trait.hpp`.

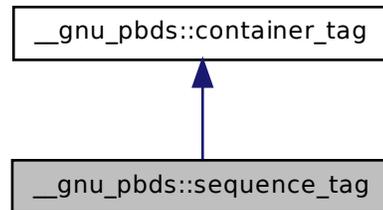
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.209 `__gnu_pbds::sequence_tag` Struct Reference

Basic sequence.

Inheritance diagram for `__gnu_pbds::sequence_tag`:



### 5.209.1 Detailed Description

Basic sequence.

Definition at line 99 of file `tag_and_trait.hpp`.

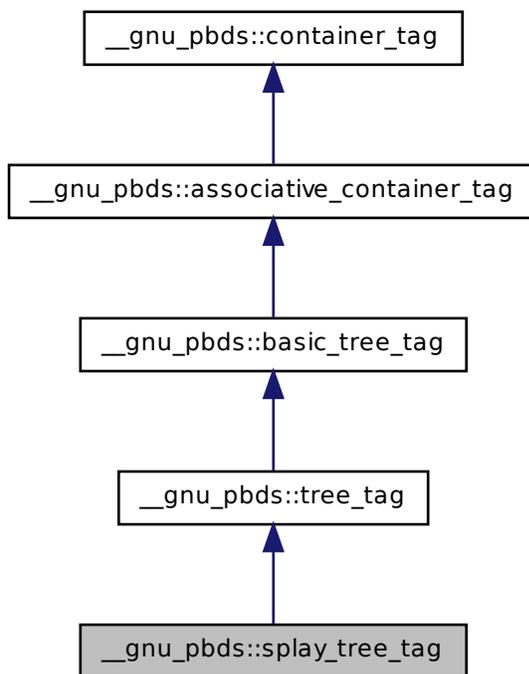
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.210 `__gnu_pbds::splay_tree_tag` Struct Reference

Splay tree.

Inheritance diagram for `__gnu_pbds::splay_tree_tag`:



### 5.210.1 Detailed Description

Splay tree.

Definition at line 123 of file `tag_and_trait.hpp`.

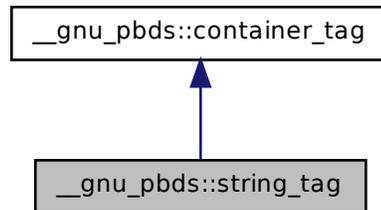
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.211 `__gnu_pbds::string_tag` Struct Reference

Basic string container, inclusive of strings, ropes, etc.

Inheritance diagram for `__gnu_pbds::string_tag`:



### 5.211.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 96 of file `tag_and_trait.hpp`.

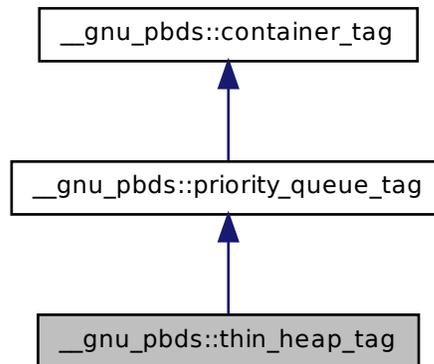
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.212 `__gnu_pbds::thin_heap_tag` Struct Reference

Thin heap.

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



### 5.212.1 Detailed Description

Thin heap.

Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.213 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic tree-based associative container.

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`:



## Public Types

- typedef Allocator **allocator\_type**
- typedef Cmp\_Fn **cmp\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Tag, Allocator >::node\_update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- **tree** (const cmp\_fn &c)
- template<typename It >  
  **tree** (It first, It last, const cmp\_fn &c)
- **tree** (const tree &other)
- template<typename It >  
  **tree** (It first, It last)
- tree & **operator=** (const tree &other)
- void **swap** (tree &other)

### 5.213.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn =
std::less<Key>, typename Tag = rb_tree_tag, template< typename Const_
Node_Iterator, typename Node_Iterator, typename Cmp_Fn_, typename
Allocator_ > class Node_Update = __gnu_pbds::null_tree_node_update, type-
name Allocator = std::allocator<char>> class __gnu_pbds::tree< Key, Mapped,
Cmp_Fn, Tag, Node_Update, Allocator >
```

A concrete basic tree-based associative container.

Definition at line 509 of file `assoc_container.hpp`.

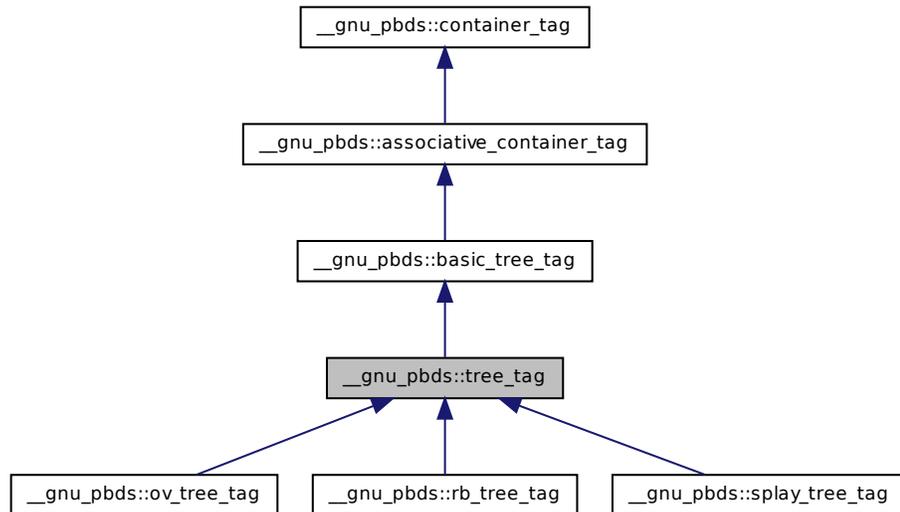
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.214 \_\_gnu\_pbds::tree\_tag Struct Reference

tree.

Inheritance diagram for `__gnu_pbds::tree_tag`:



### 5.214.1 Detailed Description

tree.

Definition at line 117 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.215 \_\_gnu\_pbds::trie< Key, Mapped, E\_Access\_Traits, Tag, Node\_Update, Allocator > Class Template Reference

A concrete basic trie-based associative container.

Inheritance diagram for \_\_gnu\_pbds::trie< Key, Mapped, E\_Access\_Traits, Tag, Node\_Update, Allocator >:



### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef E\_Access\_Traits **e\_access\_traits**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**

- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef detail::trie\_traits< Key, Mapped, E\_Access\_Traits, Node\_Update, Tag, Allocator >::node\_update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- **trie** (const e\_access\_traits &t)
- template<typename It >  
**trie** (It first, It last, const e\_access\_traits &t)
- **trie** (const [trie](#) &other)
- template<typename It >  
**trie** (It first, It last)
- **trie & operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

### 5.215.1 Detailed Description

```
template<typename Key, typename Mapped, typename E_Access_Traits = type-
name detail::default_trie_e_access_traits<Key>::type, typename Tag = pat_
trie_tag, template< typename Const_Node_Iterator, typename Node_Iterator,
typename E_Access_Traits_, typename Allocator_ > class Node_Update = null_
trie_node_update, typename Allocator = std::allocator<char>> class __gnu_
pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >
```

A concrete basic trie-based associative container.

Definition at line 585 of file `assoc_container.hpp`.

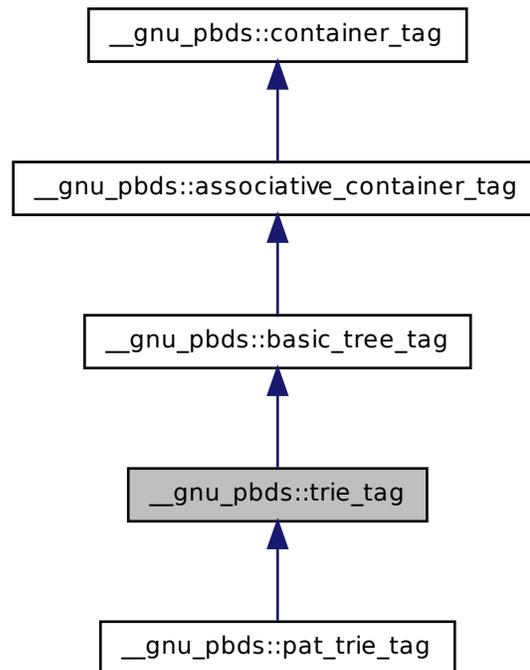
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.216 \_\_gnu\_pbds::trie\_tag Struct Reference

trie.

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



### 5.216.1 Detailed Description

trie.

Definition at line 129 of file `tag_and_trait.hpp`.

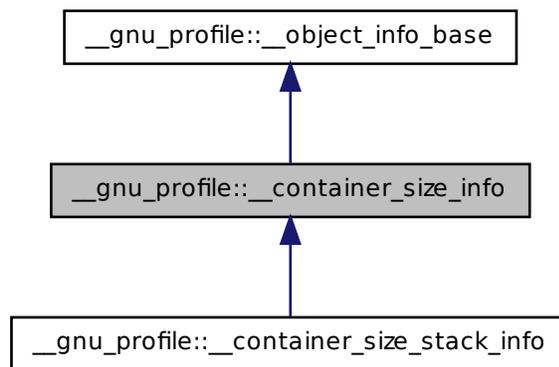
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.217 `__gnu_profile::__container_size_info` Class Reference

A container size instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__container_size_info`:



### Public Member Functions

- `__container_size_info` (const `__container_size_info` &\_\_o)
- `__container_size_info` (`__stack_t` \_\_stack, `size_t` \_\_num)
- `const char *` `__advice` () const
- `void` `__destruct` (`size_t` \_\_num, `size_t` \_\_inum)
- `bool` `__is_valid` () const
- `float` `__magnitude` () const
- `void` `__merge` (const `__container_size_info` &\_\_o)
- `void` `__resize` (`size_t` \_\_from, `size_t` \_\_to)
- `float` `__resize_cost` (`size_t` \_\_from, `size_t` \_\_to)
- `__stack_t` `__stack` () const
- `void` `__write` (FILE \*f) const

### Protected Attributes

- `__stack_t` `_M_stack`

- `bool _M_valid`

### 5.217.1 Detailed Description

A container size instrumentation line in the object table.

Definition at line 60 of file `profiler_container_size.h`.

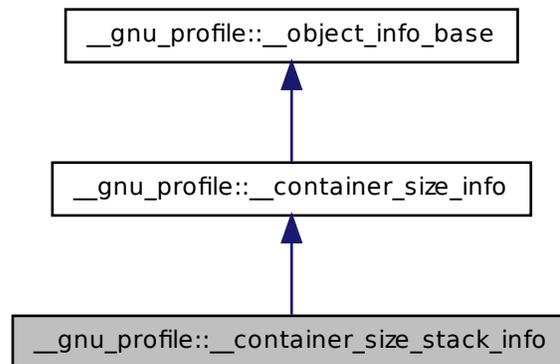
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

## 5.218 `__gnu_profile::__container_size_stack_info` Class Reference

A container size instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__container_size_stack_info`:



### Public Member Functions

- `__container_size_stack_info` (const `__container_size_info` &`_o`)
- `const char * __advice` () const

- void **\_\_destruct** (size\_t \_\_num, size\_t \_\_inum)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void **\_\_resize** (size\_t \_\_from, size\_t \_\_to)
- float **\_\_resize\_cost** (size\_t \_\_from, size\_t \_\_to)
- `__stack_t` **\_\_stack** () const
- void **\_\_write** (FILE \*f) const

### Protected Attributes

- `__stack_t` **\_M\_stack**
- bool **\_M\_valid**

#### 5.218.1 Detailed Description

A container size instrumentation line in the stack table.

Definition at line 176 of file `profiler_container_size.h`.

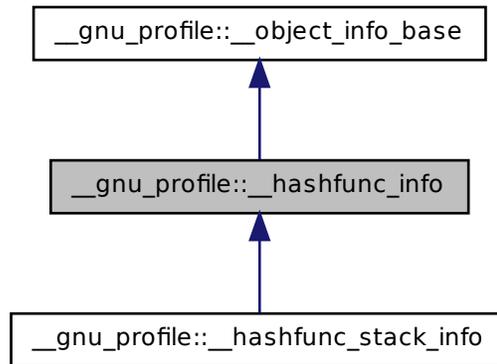
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

#### 5.219 `__gnu_profile::__hashfunc_info` Class Reference

A hash performance instrumentation line in the object table.

Inheritance diagram for \_\_gnu\_profile::\_\_hashfunc\_info:



## Public Member Functions

- `__hashfunc_info` (const `__hashfunc_info` &o)
- `__hashfunc_info` (`__stack_t` \_\_stack)
- `const char * __advice` () const
- `void __destruct` (size\_t \_\_chain, size\_t \_\_accesses, size\_t \_\_hops)
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__hashfunc_info` &o)
- `__stack_t __stack` () const
- `void __write` (FILE \* \_\_f) const

## Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

### 5.219.1 Detailed Description

A hash performance instrumentation line in the object table.

Definition at line 57 of file profiler\_hash\_func.h.

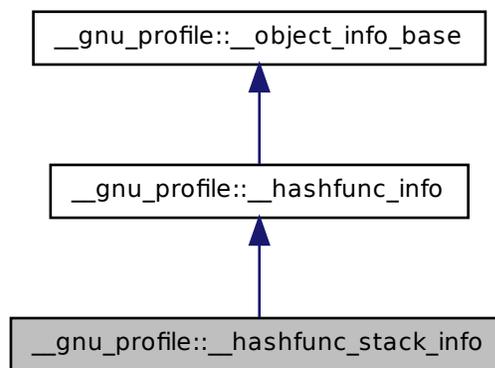
The documentation for this class was generated from the following file:

- profiler\_hash\_func.h

## 5.220 `__gnu_profile::__hashfunc_stack_info` Class Reference

A hash performance instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



### Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &\_\_o)
- `const char * __advice` () const
- `void __destruct` (size\_t \_\_chain, size\_t \_\_accesses, size\_t \_\_hops)
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__hashfunc_info` &\_\_o)
- `__stack_t __stack` () const
- `void __write` (FILE \* \_\_f) const

## Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

### 5.220.1 Detailed Description

A hash performance instrumentation line in the stack table.

Definition at line 104 of file `profiler_hash_func.h`.

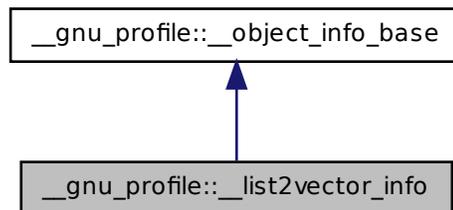
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

## 5.221 `__gnu_profile::__list2vector_info` Class Reference

A list-to-vector instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__list2vector_info`:



## Public Member Functions

- `__list2vector_info` (`__stack_t __stack`)
- `__list2vector_info` (`const __list2vector_info &__o`)
- `const char * __advice` () `const`
- `bool __is_valid` () `const`

- `bool __is_valid ()`
- `size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __list2vector_info &__o)`
- `void __opr_insert (size_t __shift, size_t __size)`
- `void __opr_iterate (size_t __num)`
- `size_t __resize ()`
- `void __resize (size_t __from, size_t __to)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE * __f) const`

### Protected Attributes

- `__stack_t _M_stack`

#### 5.221.1 Detailed Description

A list-to-vector instrumentation line in the object table.

Definition at line 59 of file `profiler_list_to_vector.h`.

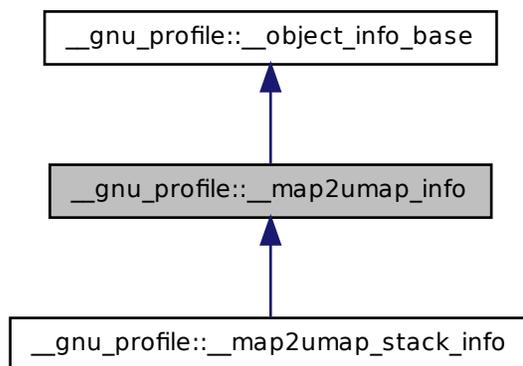
The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)

#### 5.222 `__gnu_profile::__map2umap_info` Class Reference

A map-to-unordered\_map instrumentation line in the object table.

Inheritance diagram for \_\_gnu\_profile::\_\_map2umap\_info:



## Public Member Functions

- **\_\_map2umap\_info** (\_\_stack\_t \_\_stack)
- **\_\_map2umap\_info** (const [\\_\\_map2umap\\_info](#) &o)
- const char \* **\_\_advice** () const
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_map2umap\\_info](#) &o)
- void **\_\_record\_erase** (size\_t \_\_size, size\_t \_\_count)
- void **\_\_record\_find** (size\_t \_\_size)
- void **\_\_record\_insert** (size\_t \_\_size, size\_t \_\_count)
- void **\_\_record\_invalidate** ()
- void **\_\_record\_iterate** (size\_t \_\_count)
- **\_\_stack\_t \_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

## Protected Attributes

- **\_\_stack\_t \_M\_stack**

### 5.222.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

Definition at line 86 of file profiler\_map\_to\_unordered\_map.h.

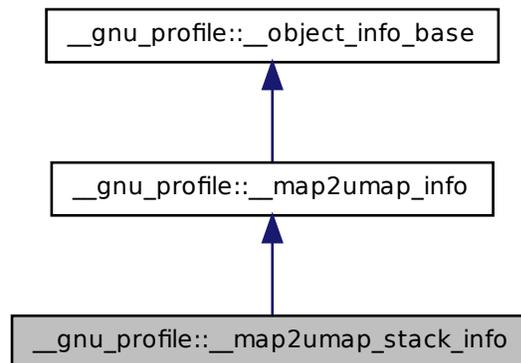
The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

### 5.223 \_\_gnu\_profile::\_\_map2umap\_stack\_info Class Reference

A map-to-unordered\_map instrumentation line in the stack table.

Inheritance diagram for \_\_gnu\_profile::\_\_map2umap\_stack\_info:



#### Public Member Functions

- `__map2umap_stack_info` (const [\\_\\_map2umap\\_info](#) &o)
- `const char * __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const [\\_\\_map2umap\\_info](#) &o)

- void `__record_erase` (size\_t \_\_size, size\_t \_\_count)
- void `__record_find` (size\_t \_\_size)
- void `__record_insert` (size\_t \_\_size, size\_t \_\_count)
- void `__record_invalidate` ()
- void `__record_iterate` (size\_t \_\_count)
- `__stack_t __stack` () const
- void `__write` (FILE \* \_\_f) const

## Protected Attributes

- `__stack_t _M_stack`

### 5.223.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

Definition at line 189 of file `profiler_map_to_unordered_map.h`.

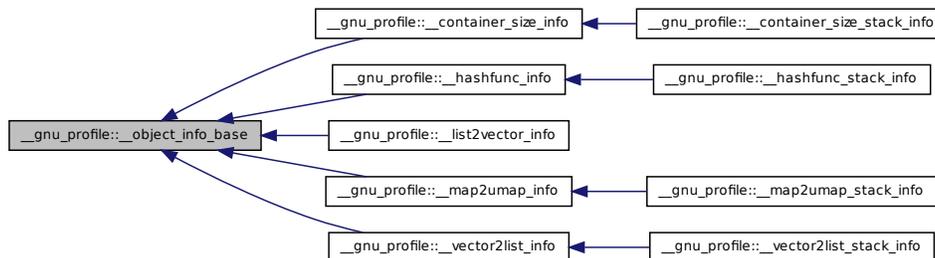
The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.224 `__gnu_profile::__object_info_base` Class Reference

Base class for a line in the object table.

Inheritance diagram for `__gnu_profile::__object_info_base`:



## Public Member Functions

- `__object_info_base (__stack_t __stack)`
- `__object_info_base (const __object_info_base &o)`
- `bool __is_valid () const`
- `__stack_t __stack () const`
- `virtual void __write (FILE *f) const =0`

## Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

### 5.224.1 Detailed Description

Base class for a line in the object table.

Definition at line 130 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

### 5.225 `__gnu_profile::__reentrance_guard` Struct Reference

Reentrance guard.

## Static Public Member Functions

- `static bool __get_in ()`
- `static bool & __inside ()`

### 5.225.1 Detailed Description

Reentrance guard. Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 69 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

## 5.226 `__gnu_profile::__stack_hash` Class Reference

Hash function for summary trace using call stack as index.

### Public Member Functions

- `size_t operator()` (const `__stack_t __s`) const
- `bool operator()` (const `__stack_t __stack1`, const `__stack_t __stack2`) const

#### 5.226.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 101 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.227 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

Base class for a line in the stack table.

### Public Member Functions

- `__stack_info_base` (const `__object_info &__info`)=0
- virtual const char \* `__get_id` () const =0
- virtual float `__magnitude` () const =0
- void `__merge` (const `__object_info &__info`)=0

#### 5.227.1 Detailed Description

```
template<typename __object_info> class __gnu_profile::__stack_info_base< __-  
object_info >
```

Base class for a line in the stack table.

Definition at line 160 of file `profiler_node.h`.

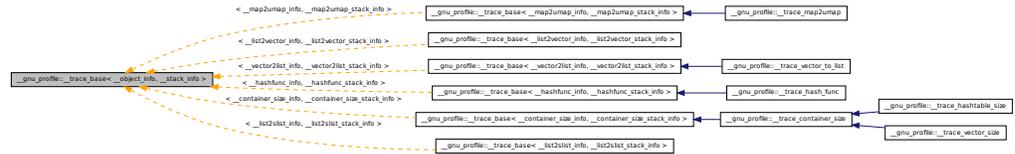
The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.228 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Base class for all trace producers.

Inheritance diagram for `__gnu_profile::__trace_base< __object_info, __stack_info >`:



### Public Member Functions

- `void __add_object (__object_t object, __object_info __info)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `__object_info * __get_object_info (__object_t __object)`
- `void __retire_object (__object_t __object)`
- `void __write (FILE *f)`

### Protected Attributes

- `const char * __id`

#### 5.228.1 Detailed Description

`template<typename __object_info, typename __stack_info> class __gnu_profile::__trace_base< __object_info, __stack_info >`

Base class for all trace producers.

Definition at line 195 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

## 5.229 `__gnu_profile::__trace_container_size` Class Reference

Container size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_container_size`:



### Public Member Functions

- void `__add_object` (`__object_t` object, `__container_size_info` info)
- void `__collect_warnings` (`__warning_vector_t` &`__warnings`)
- void `__construct` (const void \*`__obj`, `size_t` `__inum`)
- void `__destruct` (const void \*`__obj`, `size_t` `__num`, `size_t` `__inum`)
- `__container_size_info` \* `__get_object_info` (`__object_t` `__object`)
- void `__insert` (const `__object_t` `__obj`, `__stack_t` `__stack`, `size_t` `__num`)
- void `__resize` (const void \*`__obj`, int `__from`, int `__to`)
- void `__retire_object` (`__object_t` `__object`)
- void `__write` (FILE \*`f`)

### Protected Attributes

- const char \* `__id`

#### 5.229.1 Detailed Description

Container size instrumentation trace producer.

Definition at line 184 of file `profiler_container_size.h`.

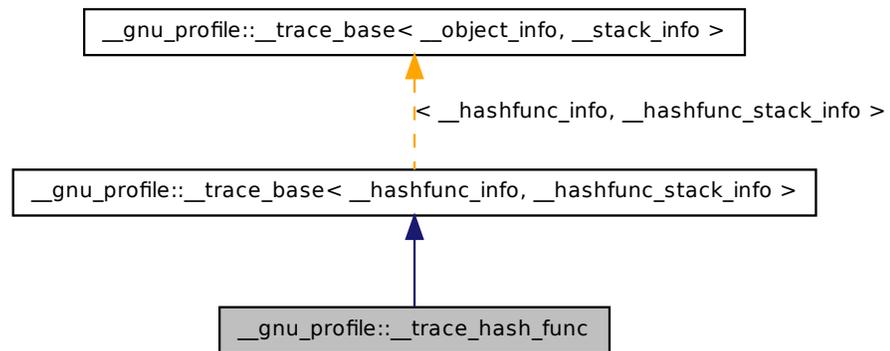
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

## 5.230 `__gnu_profile::__trace_hash_func` Class Reference

Hash performance instrumentation producer.

Inheritance diagram for `__gnu_profile::__trace_hash_func`:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_hashfunc\_info \_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **\_\_destruct** (const void \*\_\_obj, size\_t \_\_chain, size\_t \_\_accesses, size\_t \_\_hops)
- [\\_\\_hashfunc\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_insert** (\_\_object\_t \_\_obj, \_\_stack\_t \_\_stack)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*f)

## Protected Attributes

- const char \* **\_\_id**

### 5.230.1 Detailed Description

Hash performance instrumentation producer.

Definition at line 110 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

## 5.231 `__gnu_profile::__trace_hashtable_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



### Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info` \_\_info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- void **\_\_construct** (const void \*\_\_obj, `size_t` \_\_inum)
- void **\_\_destruct** (const void \*\_\_obj, `size_t` \_\_num, `size_t` \_\_inum)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_insert** (const `__object_t` \_\_obj, `__stack_t` \_\_stack, `size_t` \_\_num)
- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (FILE \*f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.231.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 59 of file `profiler_hashtable_size.h`.

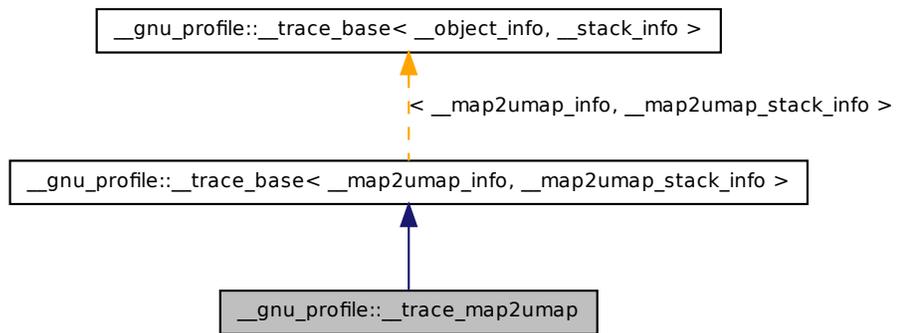
The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

## 5.232 `__gnu_profile::__trace_map2umap` Class Reference

Map-to-unordered\_map instrumentation producer.

Inheritance diagram for `__gnu_profile::__trace_map2umap`:



## Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_map2umap\_info\_\_info)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- [\\_\\_map2umap\\_info](#) \* **\_\_get\_object\_info** (\_\_object\_t \_\_object)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*f)

## Protected Attributes

- const char \* **\_\_id**

### 5.232.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

Definition at line 196 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.233 `__gnu_profile::__trace_vector_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



### Public Member Functions

- void `__add_object` (`__object_t` object, `__container_size_info` \_\_info)
- void `__collect_warnings` (`__warning_vector_t` &\_\_warnings)
- void `__construct` (const void \*\_\_obj, size\_t \_\_inum)
- void `__destruct` (const void \*\_\_obj, size\_t \_\_num, size\_t \_\_inum)
- `__container_size_info` \* `__get_object_info` (`__object_t` \_\_object)
- void `__insert` (const `__object_t` \_\_obj, `__stack_t` \_\_stack, size\_t \_\_num)
- void `__resize` (const void \*\_\_obj, int \_\_from, int \_\_to)
- void `__retire_object` (`__object_t` \_\_object)
- void `__write` (FILE \*f)

### Protected Attributes

- const char \* `__id`

#### 5.233.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 59 of file `profiler_vector_size.h`.

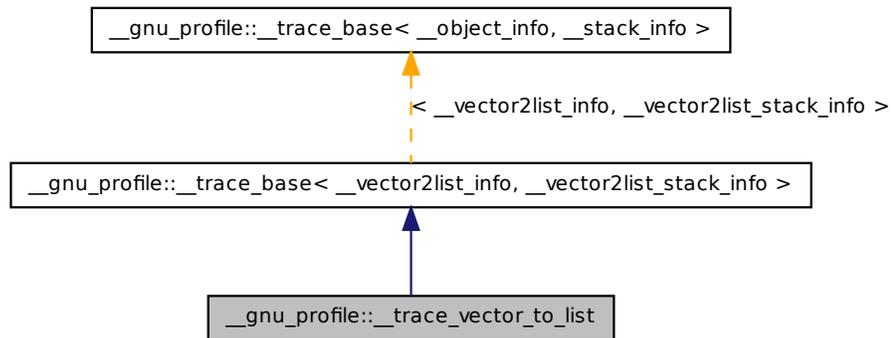
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)

## 5.234 `__gnu_profile::__trace_vector_to_list` Class Reference

Vector-to-list instrumentation producer.

Inheritance diagram for `__gnu_profile::__trace_vector_to_list`:



## Public Member Functions

- `void __add_object (__object_t object, __vector2list_info __info)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `void __destruct (const void *__obj)`
- `__vector2list_info * __find (const void *__obj)`
- `__vector2list_info * __get_object_info (__object_t __object)`
- `void __insert (__object_t __obj, __stack_t __stack)`
- `void __invalid_operator (const void *__obj)`
- `float __list_cost (size_t __shift, size_t __iterate, size_t __resize)`
- `void __opr_find (const void *__obj, size_t __size)`
- `void __opr_insert (const void *__obj, size_t __pos, size_t __num)`
- `void __opr_iterate (const void *__obj, size_t __num)`
- `void __resize (const void *__obj, size_t __from, size_t __to)`
- `void __retire_object (__object_t __object)`
- `float __vector_cost (size_t __shift, size_t __iterate, size_t __resize)`
- `void __write (FILE *f)`

## Protected Attributes

- `const char * __id`

### 5.234.1 Detailed Description

Vector-to-list instrumentation producer.

Definition at line 147 of file `profiler_vector_to_list.h`.

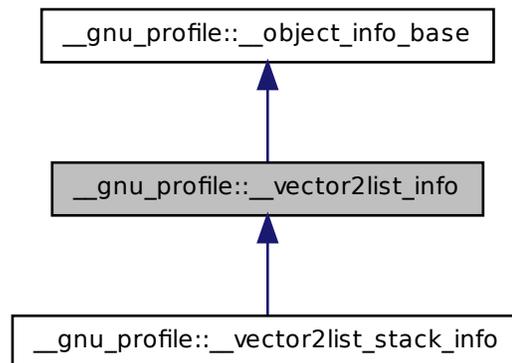
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.235 `__gnu_profile::__vector2list_info` Class Reference

A vector-to-list instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__vector2list_info`:



### Public Member Functions

- `__vector2list_info` (`__stack_t __stack`)
- `__vector2list_info` (`const __vector2list_info &__o`)
- `const char * __advice` () const
- `bool __is_valid` () const
- `bool __is_valid` ()

- `size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __vector2list_info &__o)`
- `void __opr_find (size_t __size)`
- `void __opr_insert (size_t __pos, size_t __num)`
- `void __opr_iterate (size_t __num)`
- `void __resize (size_t __from, size_t __to)`
- `size_t __resize ()`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE * __f) const`

### Protected Attributes

- `__stack_t _M_stack`

#### 5.235.1 Detailed Description

A vector-to-list instrumentation line in the object table.

Definition at line 57 of file `profiler_vector_to_list.h`.

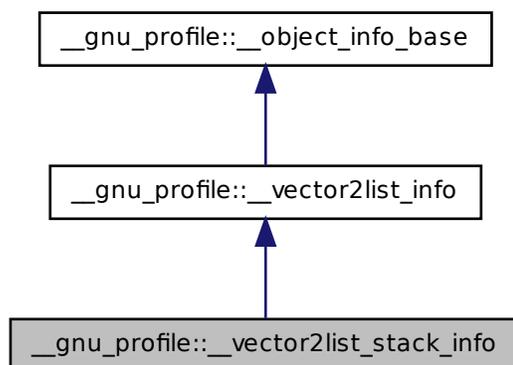
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

#### 5.236 `__gnu_profile::__vector2list_stack_info` Class Reference

A vector-to-list instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



## Public Member Functions

- `__vector2list_stack_info` (const `__vector2list_info` &`_o`)
- `const char * __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const `__vector2list_info` &`_o`)
- `void __opr_find` (size\_t `_size`)
- `void __opr_insert` (size\_t `_pos`, size\_t `_num`)
- `void __opr_iterate` (size\_t `_num`)
- `void __resize` (size\_t `_from`, size\_t `_to`)
- `size_t __resize` ()
- `void __set_invalid` ()
- `void __set_list_cost` (float `_lc`)
- `void __set_vector_cost` (float `_vc`)
- `size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (FILE \* `_f`) const

## Protected Attributes

- `__stack_t __M_stack`

### 5.236.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 140 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.237 `__gnu_profile::__warning_data` Struct Reference

Representation of a warning.

### Public Member Functions

- `__warning_data` (float `__m`, `__stack_t` `__c`, const char \*`__id`, const char \*`__msg`)
- `bool operator>` (const struct `__warning_data` &`__other`) const

### Public Attributes

- `__stack_t __context`
- float `__magnitude`
- const char \* `__warning_id`
- const char \* `__warning_message`

### 5.237.1 Detailed Description

Representation of a warning.

Definition at line 82 of file `profiler_trace.h`.

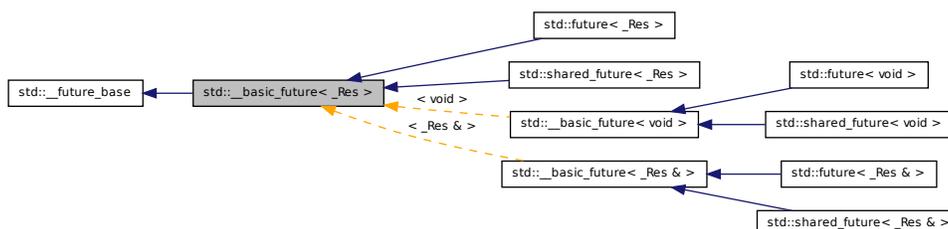
The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

## 5.238 std::\_\_basic\_future< \_Res > Class Template Reference

Common implementation for future and [shared\\_future](#).

Inheritance diagram for std::\_\_basic\_future< \_Res >:



### Public Member Functions

- `__basic_future` (const `__basic_future` &)
- `__basic_future` & `operator=` (const `__basic_future` &)
- bool `valid` () const
- void `wait` () const
- template<typename \_Rep, typename \_Period >  
bool `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration >  
bool `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs)  
const

### Protected Types

- typedef `__future_base::Result`< \_Res > & `__result_type`
- typedef `shared_ptr`< \_State > `__state_type`

### Protected Member Functions

- `__basic_future` (const `__state_type` &\_\_state)
- `__basic_future` (const `shared_future`< \_Res > &)
- `__basic_future` (`shared_future`< \_Res > &&)
- `__basic_future` (`future`< \_Res > &&)

- [\\_\\_result\\_type \\_M\\_get\\_result \(\)](#)
- [void \\_M\\_swap \(\\_\\_basic\\_future &\\_\\_that\)](#)

### 5.238.1 Detailed Description

`template<typename _Res> class std::__basic_future< _Res >`

Common implementation for future and [shared\\_future](#).

Definition at line 480 of file future.

### 5.238.2 Member Function Documentation

**5.238.2.1** `template<typename _Res> __result_type std::__basic_future< _Res >::__M_get_result ( ) [inline, protected]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

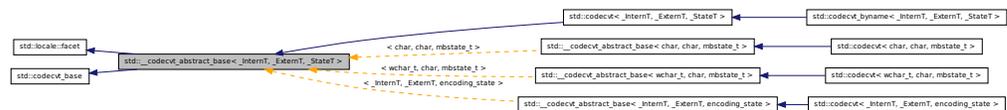
The documentation for this class was generated from the following file:

- [future](#)

## 5.239 `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >` Class Template Reference

Common base for codecvt functions.

Inheritance diagram for `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const

## Protected Member Functions

- `__codecvt_abstract_base` (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const =0

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_type_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.239.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`

Common base for codecvt functions. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 67 of file `codecvt.h`.

### 5.239.2 Member Function Documentation

**5.239.2.1** `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [protected, pure virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implemented in `std::codecvt< _InternT, _ExternT, _StateT >`.

5.239.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::_codecvt_abstract_base<_InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const [inline]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::underflow()`.

```

5.239.2.3 template<typename _InternT, typename _ExternT, typename
_StateT> result std::_codecvt_abstract_base< _InternT, _ExternT,
_StateT >::out ( state_type & __state, const intern_type * __from,
const intern_type * __from_end, const intern_type *& __from_next,
extern_type * __to, extern_type * __to_end, extern_type *&
__to_next ) const [inline]

```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.

*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```
5.239.2.4 template<typename _InternT, typename _ExternT, typename
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
_StateT >::unshift ( state_type & __state, extern_type * __to,
extern_type * __to_end, extern_type *& __to_next ) const
[inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

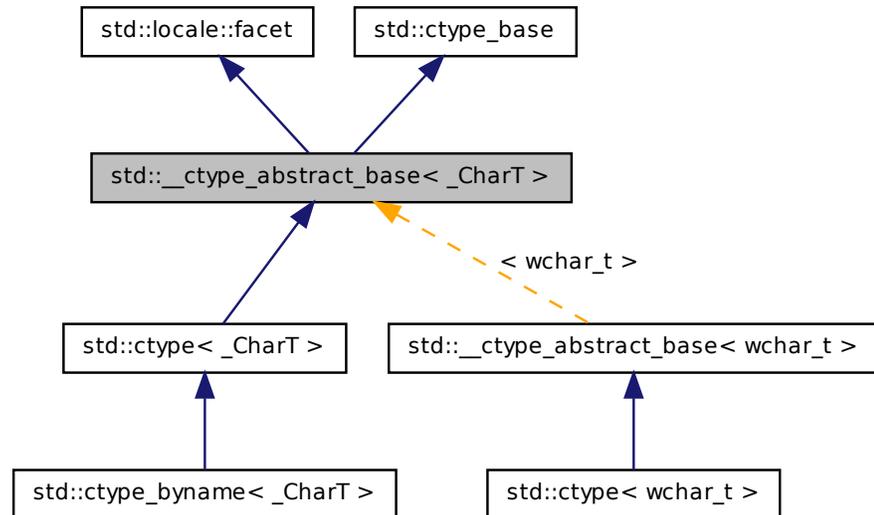
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.240 `std::__ctype_abstract_base< _CharT >` Class Template Reference

Common base for ctype facet.

Inheritance diagram for `std::_ctype_abstract_base<_CharT>`:



## Public Types

- typedef `const int * __to_type`
- typedef `_CharT char_type`
- typedef `unsigned short mask`

## Public Member Functions

- `bool is (mask __m, char_type __c) const`
- `const char_type * is (const char_type * __lo, const char_type * __hi, mask * __vec) const`
- `char narrow (char_type __c, char __dfault) const`
- `const char_type * narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const`
- `const char_type * scan_is (mask __m, const char_type * __lo, const char_type * __hi) const`

- `const char_type * scan_not` (mask `__m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `const char_type * tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char * widen` (`const char * __lo`, `const char * __hi`, `char_type * __to`) `const`
- `char_type widen` (`char __c`) `const`

### Static Public Attributes

- static `const mask` **alnum**
- static `const mask` **alpha**
- static `const mask` **cntrl**
- static `const mask` **digit**
- static `const mask` **graph**
- static `const mask` **lower**
- static `const mask` **print**
- static `const mask` **punct**
- static `const mask` **space**
- static `const mask` **upper**
- static `const mask` **xdigit**

### Protected Member Functions

- `__ctype_abstract_base` (`size_t __refs=0`)
- virtual `const char_type * do_is` (`const char_type * __lo`, `const char_type * __hi`, `mask * __vec`) `const =0`
- virtual `bool do_is` (`mask __m`, `char_type __c`) `const =0`
- virtual `char do_narrow` (`char_type`, `char __dfault`) `const =0`
- virtual `const char_type * do_narrow` (`const char_type * __lo`, `const char_type * __hi`, `char __dfault`, `char * __dest`) `const =0`
- virtual `const char_type * do_scan_is` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const =0`
- virtual `const char_type * do_scan_not` (`mask __m`, `const char_type * __lo`, `const char_type * __hi`) `const =0`
- virtual `char_type do_tolower` (`char_type`) `const =0`
- virtual `const char_type * do_tolower` (`char_type * __lo`, `const char_type * __hi`) `const =0`
- virtual `char_type do_toupper` (`char_type`) `const =0`
- virtual `const char_type * do_toupper` (`char_type * __lo`, `const char_type * __hi`) `const =0`

- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const =0
- virtual [char\\_type do\\_widen](#) (char) const =0

## Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static [\\_GLIBCXX\\_CONST](#) const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Friends

- class [locale::\\_Impl](#)

### 5.240.1 Detailed Description

**template<typename [\\_CharT](#)> class [std::\\_\\_ctype\\_abstract\\_base<\\_CharT](#) >**

Common base for ctype facet. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file [locale\\_facets.h](#).

### 5.240.2 Member Typedef Documentation

**5.240.2.1 [template<typename \[\\\_CharT\]\(#\)> typedef \[\\\_CharT\]\(#\) \[std::\\\_\\\_ctype\\\_abstract\\\_base<\\\_CharT\]\(#\) >::char\\_type](#)**

Typedef for the template parameter.

Reimplemented in [std::ctype<\\_CharT>](#), and [std::ctype<wchar\\_t>](#).

Definition at line 148 of file [locale\\_facets.h](#).

### 5.240.3 Member Function Documentation

**5.240.3.1** `template<typename _CharT> virtual bool std::__ctype_abstract_base<_CharT>::do_is ( mask __m, char_type __c ) const`  
`[protected, pure virtual]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

- c* The `char_type` to find the mask of.
- m* The mask to compare against.

#### Returns

$(M \& m) \neq 0$ .

Implemented in `std::ctype<_CharT>`.

**5.240.3.2** `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const`  
`[protected, pure virtual]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implemented in `std::ctype<_CharT>`.

**5.240.3.3** `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT >::do_narrow ( char_type, char __dfault ) const [protected, pure virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted `char`.

Implemented in `std::ctype<_CharT >`.

**5.240.3.4** `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT >::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __dest ) const [protected, pure virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

**Returns***hi*.Implemented in `std::ctype<_CharT>`.

```
5.240.3.5 template<typename _CharT> virtual const char_type*
std::_ctype_abstract_base<_CharT>::do_scan_is ( mask
__m, const char_type * __lo, const char_type * __hi ) const
[protected, pure virtual]
```

Find `char_type` matching mask.This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.**Parameters***m* The mask to compare against.*lo* Pointer to start of range.*hi* Pointer to end of range.**Returns**Pointer to a matching `char_type` if found, else *hi*.Implemented in `std::ctype<_CharT>`.

```
5.240.3.6 template<typename _CharT> virtual const char_type*
std::_ctype_abstract_base<_CharT>::do_scan_not ( mask
__m, const char_type * __lo, const char_type * __hi ) const
[protected, pure virtual]
```

Find `char_type` not matching mask.This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.**Parameters***m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implemented in `std::ctype<_CharT>`.

**5.240.3.7** `template<typename _CharT> virtual char_type  
std::__ctype_abstract_base<_CharT >::do_tolower ( char_type )  
const [protected, pure virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

### Parameters

*c* The `char_type` to convert.

### Returns

The lowercase `char_type` if convertible, else *c*.

Implemented in `std::ctype<_CharT>`.

**5.240.3.8** `template<typename _CharT> virtual const char_type*  
std::__ctype_abstract_base<_CharT >::do_tolower ( char_type  
* __lo, const char_type * __hi ) const [protected, pure  
virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

*hi*.

Implemented in [std::ctype<\\_CharT>](#).

```
5.240.3.9 template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_toupper ( char_type )
const [protected, pure virtual]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

### Parameters

*c* The `char_type` to convert.

### Returns

The uppercase `char_type` if convertible, else *c*.

Implemented in [std::ctype<\\_CharT>](#).

```
5.240.3.10 template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_toupper ( char_type
* __lo, const char_type * __hi ) const [protected, pure
virtual]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Implemented in [std::ctype<\\_CharT>](#).

**5.240.3.11** `template<typename _CharT> virtual char_type  
std::__ctype_abstract_base<_CharT>::do_widen ( char ) const  
[protected, pure virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type

Implemented in [std::ctype<\\_CharT>](#), and [std::ctype<wchar\\_t>](#).

**5.240.3.12** `template<typename _CharT> virtual const char*  
std::__ctype_abstract_base<_CharT>::do_widen ( const  
char * __lo, const char * __hi, char_type * __dest ) const  
[protected, pure virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

**Parameters**

*lo* Pointer to start range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

### Returns

*hi*.

Implemented in `std::ctype<_CharT>`.

**5.240.3.13** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::is ( const char_type *  
__lo, const char_type * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

### Parameters

*lo* Pointer to start of range.  
*hi* Pointer to end of range.  
*vec* Pointer to an array of mask storage.

### Returns

*hi*.

Definition at line 178 of file `locale_facets.h`.

**5.240.3.14** `template<typename _CharT> bool std::__ctype_abstract_base<  
_CharT>::is ( mask __m, char_type __c ) const [inline]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`. It does so by returning the value of `ctype<char_type>::do_is()`.

### Parameters

*c* The `char_type` to compare the mask of.  
*m* The mask to compare against.

### Returns

$(M \& m) \neq 0$ .

Definition at line 161 of file locale\_facets.h.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

```
5.240.3.15 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::narrow ( const char_type
* __lo, const char_type * __hi, char __dfault, char * __to ) const
[inline]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 345 of file locale\_facets.h.

```
5.240.3.16 template<typename _CharT> char std::__ctype_abstract_base<
_CharT>::narrow ( char_type __c, char __dfault ) const
[inline]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 323 of file locale\_facets.h.

Referenced by std::time\_put<\_CharT, \_OutIter>::put().

```
5.240.3.17 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,
const char_type * __lo, const char_type * __hi ) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

Pointer to matching char\_type if found, else *hi*.

Definition at line 194 of file locale\_facets.h.

```
5.240.3.18 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::scan_not ( mask __m,
const char_type * __lo, const char_type * __hi ) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

*m* The mask to compare against.

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale\_facets.h.

**5.240.3.19** `template<typename _CharT> char_type std::_ctype_-  
abstract_base< _CharT >::tolower ( char_type __c ) const  
[inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_-tolower(c)`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

**5.240.3.20** `template<typename _CharT> const char_type*  
std::_ctype_abstract_base< _CharT >::tolower ( char_type *  
__lo, const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_-tolower(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 268 of file `locale_facets.h`.

**5.240.3.21** `template<typename _CharT> char_type std::_ctype_-  
abstract_base< _CharT >::toupper ( char_type __c ) const  
[inline]`

Convert to uppercase.

## 5.240 std::\_\_ctype\_abstract\_base<\_CharT> Class Template Reference 1265

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

### Parameters

*c* The `char_type` to convert.

### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

**5.240.3.22** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::toupper ( char_type *  
__lo, const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

*hi*.

Definition at line 239 of file `locale_facets.h`.

**5.240.3.23** `template<typename _CharT> char_type std::__ctype_ -  
abstract_base<_CharT>::widen ( char __c ) const  
[inline]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type.

Definition at line 285 of file locale\_facets.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::money\_put< \_CharT, \_OutIter >::do\_put(), and std::operator<<().

**5.240.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 304 of file locale\_facets.h.

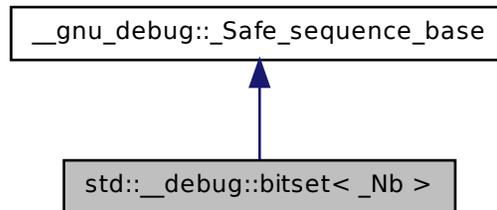
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.241 std::\_\_debug::bitset< \_Nb > Class Template Reference

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

Inheritance diagram for std::\_\_debug::bitset<\_Nb>:



## Public Member Functions

- **bitset** (unsigned long long \_\_val)
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const std::basic\_string<\_CharT, \_Traits, \_Alloc > &\_\_str, typename std::basic\_string<\_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename std::basic\_string<\_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- **bitset** (const \_Base &\_\_x)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const std::basic\_string<\_CharT, \_Traits, \_Alloc > &\_\_str, typename std::basic\_string<\_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename std::basic\_string<\_CharT, \_Traits, \_Alloc >::size\_type \_\_n=(std::basic\_string<\_CharT, \_Traits, \_Alloc >::npos))
- **bitset** (const char \*\_\_str)
- **\_Base & \_M\_base** ()
- const **\_Base & \_M\_base** () const
- void **\_M\_invalidate\_all** () const
- **bitset<\_Nb> & flip** ()
- **bitset<\_Nb> & flip** (size\_t \_\_pos)
- bool **operator!=** (const **bitset<\_Nb>** &\_\_rhs) const
- **bitset<\_Nb> & operator&=** (const **bitset<\_Nb>** &\_\_rhs)
- **bitset<\_Nb> operator<<** (size\_t \_\_pos) const
- **bitset<\_Nb> & operator<<=** (size\_t \_\_pos)
- bool **operator==** (const **bitset<\_Nb>** &\_\_rhs) const
- **bitset<\_Nb> operator>>** (size\_t \_\_pos) const

- [bitset](#)< \_Nb > & [operator](#)>>= (size\_t \_\_pos)
- reference [operator](#)[] (size\_t \_\_pos)
- bool [operator](#)[] (size\_t \_\_pos) const
- [bitset](#)< \_Nb > & [operator](#)^= (const [bitset](#)< \_Nb > & \_\_rhs)
- [bitset](#)< \_Nb > & [operator](#)|= (const [bitset](#)< \_Nb > & \_\_rhs)
- [bitset](#)< \_Nb > [operator](#)~ () const
- [bitset](#)< \_Nb > & [reset](#) ()
- [bitset](#)< \_Nb > & [reset](#) (size\_t \_\_pos)
- [bitset](#)< \_Nb > & [set](#) ()
- [bitset](#)< \_Nb > & [set](#) (size\_t \_\_pos, bool \_\_val=true)
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) (char \_\_zero, char \_\_one= '1') const
- template<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<class \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) () const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()

- void `_M_revalidate_singular()`
- void `_M_swap(_Safe_sequence_base &__x)`

### 5.241.1 Detailed Description

`template<size_t _Nb> class std::_debug::bitset<_Nb>`

Class `std::bitset` with additional safety/checking/debug instrumentation.

Definition at line 43 of file `debug/bitset`.

### 5.241.2 Member Function Documentation

**5.241.2.1** void `__gnu_debug::_Safe_sequence_base::_M_detach_all()`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.241.2.2** void `__gnu_debug::_Safe_sequence_base::_M_detach_singular()`  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.241.2.3** `__gnu_cxx::_mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex()` throw () [protected, inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

**5.241.2.4** void `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`  
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.241.2.5** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.241.2.6** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.241.3 Member Data Documentation

**5.241.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.241.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.241.3.3** `unsigned int __gnu_debug::Safe_sequence_base::M_version [mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file safe\_base.h.

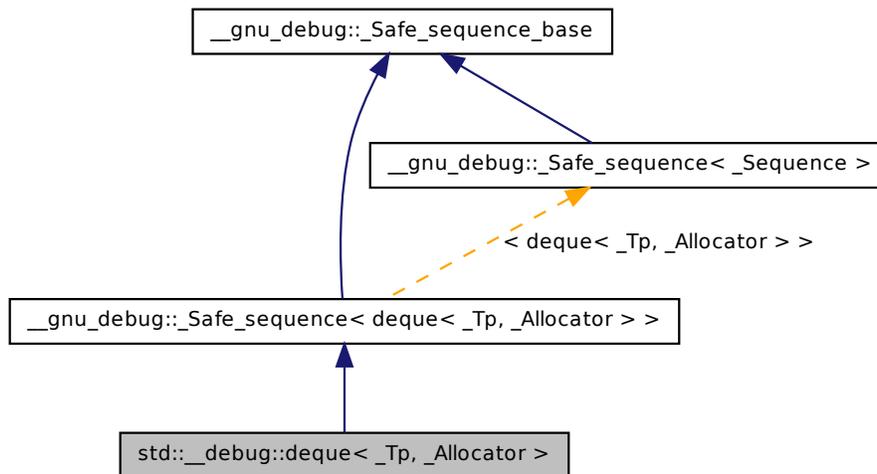
The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 5.242 std::\_\_debug::deque< \_Tp, \_Allocator > Class Template Reference

Class [std::deque](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::deque< _Tp, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, deque >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, deque >` **iterator**

- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **deque** (const `_Allocator` &\_\_a=\_Allocator())
- **deque** (size\_type \_\_n, const `_Tp` &\_\_value=\_Tp(), const `_Allocator` &\_\_a=\_Allocator())
- **deque** (const `deque` &\_\_x)
- **deque** (const `_Base` &\_\_x)
- template<class `_InputIterator` >  
**deque** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **deque** (`deque` &&\_\_x)
- **deque** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_transfer\_iter** (const `_Safe_iterator`< `_Iterator`, `deque`< `_Tp`, `_Allocator` > > &\_\_x)
- template<class `_InputIterator` >  
void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_t)
- void **assign** (`initializer_list`< `value_type` > \_\_l)
- reference **back** ()
- const\_reference **back** () const
- `const_iterator` **begin** () const
- `iterator` **begin** ()
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- template<typename... `_Args`>  
`iterator` **emplace** (`iterator` \_\_position, `_Args` &&...\_\_args)
- template<typename... `_Args`>  
void **emplace\_back** (`_Args` &&...\_\_args)

- `template<typename... _Args>`  
`void emplace_front (_Args &&... __args)`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `const_reference front () const`
- `reference front ()`
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `iterator insert (iterator __position, _Tp &&__x)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `deque & operator= (deque &&__x)`
- `deque & operator= (const deque &__x)`
- `deque & operator= (initializer_list< value_type > __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n) const`
- `void pop_back ()`
- `void pop_front ()`
- `void push_back (_Tp &&__x)`
- `void push_back (const _Tp &__x)`
- `void push_front (const _Tp &__x)`
- `void push_front (_Tp &&__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void resize (size_type __sz, _Tp __c=_Tp())`
- `void swap (deque &__x)`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

### 5.242.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::deque< _Tp, _Allocator >
```

Class [std::deque](#) with safety/checking/debug instrumentation.

Definition at line 43 of file [debug/deque](#).

### 5.242.2 Member Function Documentation

**5.242.2.1** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::\\_M\\_detach\\_all](#) ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.242.2.2** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::\\_M\\_detach\\_singular](#) ( )  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->[\\_M\\_version](#) == [\\_M\\_version](#).

**5.242.2.3** [\\_\\_gnu\\_cxx::\\_\\_mutex](#)& [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::\\_M\\_get\\_mutex](#) ( ) throw () [protected, inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence< \\_Sequence >::\\_M\\_invalidate\\_if](#)(), and [\\_\\_gnu\\_debug::Safe\\_sequence< \\_Sequence >::\\_M\\_transfer\\_iter](#)().

**5.242.2.4** void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( )  
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

**5.242.2.5** void \_\_gnu\_debug::Safe\_sequence< deque<\_Tp, \_Allocator >  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred(x)* returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

**5.242.2.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.242.2.7** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.242.2.8** void \_\_gnu\_debug::Safe\_sequence< deque<\_Tp, \_Allocator >  
>::M\_transfer\_iter ( const \_Safe\_iterator< \_Iterator, deque<\_Tp, \_Allocator > > & \_\_x ) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.242.3 Member Data Documentation

**5.242.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

### 5.242.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

### 5.242.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

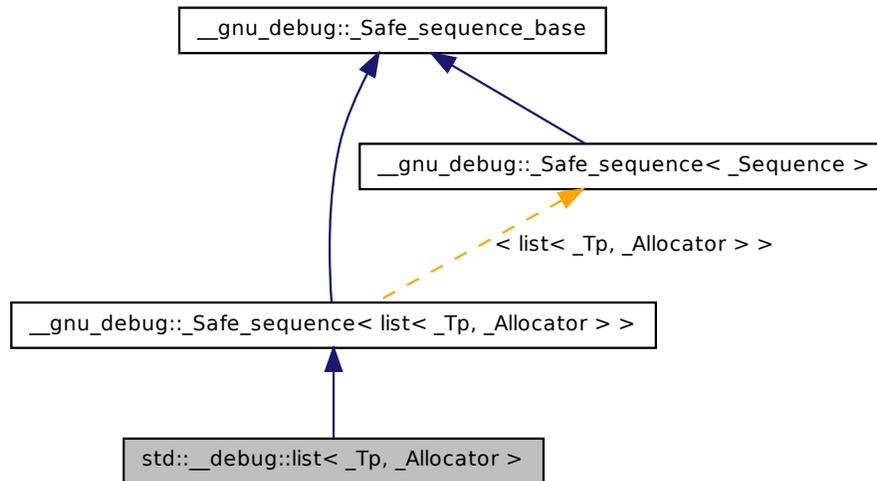
The documentation for this class was generated from the following file:

- [debug/deque](#)

## 5.243 `std::__debug::list<_Tp, _Allocator >` Class Template Reference

Class `std::list` with safety/checking/debug instrumentation.

Inheritance diagram for std::\_\_debug::list< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, list >` **iterator**
  
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **list** (const `_Allocator` &`_a`=`_Allocator`())
- **list** (size\_type `_n`, const `_Tp` &`_value`=`_Tp`(), const `_Allocator` &`_a`=`_Allocator`())
- **list** (const `list` &`_x`)
- **list** (const `_Base` &`_x`)
- template<class `_InputIterator` >  
**list** (`_InputIterator` `__first`, `_InputIterator` `__last`, const `_Allocator` &`_a`=`_Allocator`())
- **list** (`list` &&`_x`)
- **list** (`initializer_list`< `value_type` > `__l`, const `allocator_type` &`_a`=`allocator_type`()
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` `__pred`)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `list`< `_Tp`, `_Allocator` > &`_x`)
- void **assign** (`initializer_list`< `value_type` > `__l`)
- template<class `_InputIterator` >  
void **assign** (`_InputIterator` `__first`, `_InputIterator` `__last`)
- void **assign** (size\_type `_n`, const `_Tp` &`_t`)
- const\_reference **back** () const
- reference **back** ()
- `const_iterator` **begin** () const
- `iterator` **begin** ()
- `const_iterator` **cbegin** () const
- `const_iterator` **end** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- template<typename... `_Args` >  
`iterator` **emplace** (`iterator` `__position`, `_Args` &&...`__args`)
- `iterator` **end** ()
- `const_iterator` **end** () const
- `iterator` **erase** (`iterator` `__position`, `iterator` `__last`)
- `iterator` **erase** (`iterator` `__position`)
- reference **front** ()
- const\_reference **front** () const
- template<class `_InputIterator` >  
void **insert** (`iterator` `__position`, `_InputIterator` `__first`, `_InputIterator` `__last`)
- `iterator` **insert** (`iterator` `__position`, const `_Tp` &`_x`)

- `iterator insert (iterator __position, _Tp &&__x)`
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `void merge (list &__x)`
- `void merge (list &&__x)`
- `template<class _Compare >`  
`void merge (list &&__x, _Compare __comp)`
- `template<typename _Compare >`  
`void merge (list &__x, _Compare __comp)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `void remove (const _Tp &__value)`
- `template<class _Predicate >`  
`void remove_if (_Predicate __pred)`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void resize (size_type __sz, _Tp __c=_Tp())`
- `void sort ()`
- `template<typename _StrictWeakOrdering >`  
`void sort (_StrictWeakOrdering __pred)`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void swap (list &__x)`
- `template<class _BinaryPredicate >`  
`void unique (_BinaryPredicate __binary_pred)`
- `void unique ()`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x)

### 5.243.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__debug::list< _Tp, _Allocator >`

Class [std::list](#) with safety/checking/debug instrumentation.

Definition at line 43 of file `debug/list`.

### 5.243.2 Member Function Documentation

**5.243.2.1** void `__gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.243.2.2** void `__gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.243.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` [protected, inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.243.2.4** void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( )  
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

**5.243.2.5** void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator >  
>::M\_invalidate\_if( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this sequence, are not singular, and for which  $\text{pred}(x)$  returns true. The user of this routine should be careful not to make copies of the iterators passed to  $\text{pred}$ , as the copies may interfere with the invalidation.

**5.243.2.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.243.2.7** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap (   
\_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.243.2.8** void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator >  
>::M\_transfer\_iter( const \_Safe\_iterator< \_Iterator, list< \_Tp,  
\_Allocator > > & \_\_x ) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.243.3 Member Data Documentation

**5.243.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

### 5.243.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

### 5.243.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

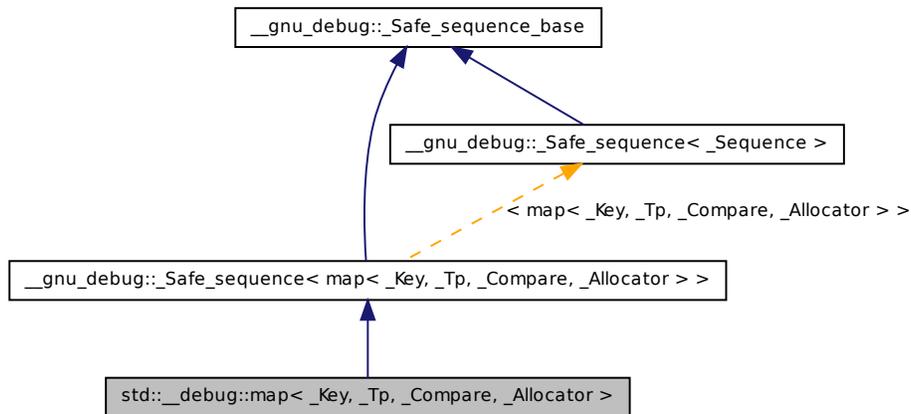
- [debug/list](#)

## 5.244 `std::__debug::map<_Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::map` with safety/checking/debug instrumentation.

## 5.244 `std::__debug::map<_Key, _Tp, _Compare, _Allocator >` Class Template Reference 1283

Inheritance diagram for `std::__debug::map<_Key, _Tp, _Compare, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, map >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, map >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **map** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **map** (const `_Base` &\_\_x)
- **map** (`map` &&\_\_x)
- **map** (const `map` &\_\_x)
- **map** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_c=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `map`< `_Key`, `_Tp`, `_Compare`, `_Allocator` > > &\_\_x)
- `iterator begin` ()
- `const_iterator begin` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- void `clear` ()
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- `iterator end` ()
- `const_iterator end` () const
- `std::pair`< `iterator`, `iterator` > `equal_range` (const `key_type` &\_\_x)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (const `key_type` &\_\_x) const
- `iterator erase` (`iterator` \_\_first, `iterator` \_\_last)
- `iterator erase` (`iterator` \_\_position)
- `size_type erase` (const `key_type` &\_\_x)
- `iterator find` (const `key_type` &\_\_x)
- `const_iterator find` (const `key_type` &\_\_x) const
- void `insert` (`std::initializer_list`< `value_type` > \_\_list)
- `std::pair`< `iterator`, `bool` > `insert` (const `value_type` &\_\_x)
- `iterator insert` (`iterator` \_\_position, const `value_type` &\_\_x)
- template<typename `_InputIterator` >  
void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator lower_bound` (const `key_type` &\_\_x)
- `const_iterator lower_bound` (const `key_type` &\_\_x) const
- `map` & `operator=` (const `map` &\_\_x)

## 5.244 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1285

---

- `map` & `operator=` (`map` &&\_\_x)
- `map` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- void `swap` (`map` &\_\_x)
- `const_iterator` `upper_bound` (const `key_type` &\_\_x) const
- `iterator` `upper_bound` (const `key_type` &\_\_x)

### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &\_\_x)

#### 5.244.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class `std::map` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/map.h`.

#### 5.244.2 Member Function Documentation

5.244.2.1 void `__gnu_debug::_Safe_sequence_base::_M_detach_all` ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.244.2.2** `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
**[protected, inherited]**

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**5.244.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` **[protected, inherited]**

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.244.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( )`  
**const [inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.244.2.5** `void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if ( _Predicate __pred )`  
**[inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.244.2.6** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( )`  
**[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.244 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference 1287

---

**5.244.2.7** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.244.2.8** void \_\_gnu\_debug::Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::M\_transfer\_iter ( const Safe\_iterator< \_Iterator, map< \_Key, \_Tp, \_Compare, \_Allocator > > & \_\_x ) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.244.3 Member Data Documentation

**5.244.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(), \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >::M\_invalidate\_single(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_iter().

**5.244.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(), \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >::M\_invalidate\_single(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_iter().

**5.244.3.3** unsigned int \_\_gnu\_debug::Safe\_sequence\_base::M\_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe\_base.h.

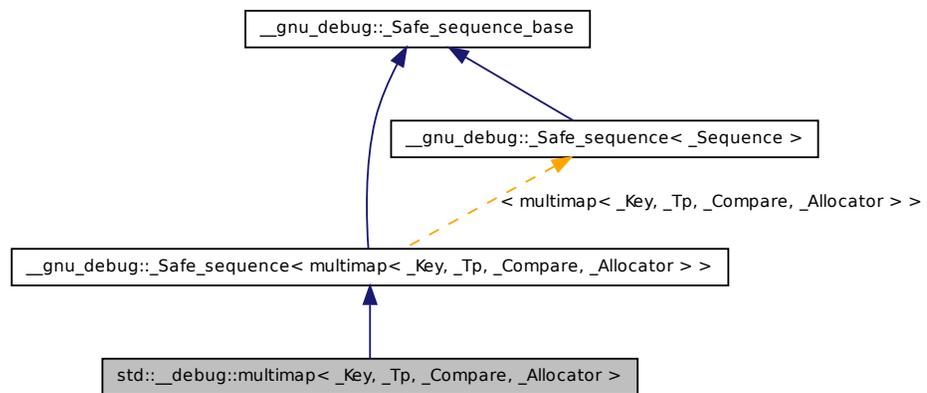
The documentation for this class was generated from the following file:

- [debug/map.h](#)

## 5.245 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, multimap >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, multimap >` **iterator**

- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **multimap** (const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- template<typename `_InputIterator` >  
**multimap** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- **multimap** (const `_Base` &\_\_x)
- **multimap** (`multimap` &&\_\_x)
- **multimap** (const `multimap` &\_\_x)
- **multimap** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_c=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_transfer\_iter** (const `_Safe_iterator`< `_Iterator`, `multimap`< `_Key`, `_Tp`, `_Compare`, `_Allocator` > > &\_\_x)
- `iterator` **begin** ()
- const `iterator` **begin** () const
- const `iterator` **cbegin** () const
- const `iterator` **cend** () const
- void **clear** ()
- const `reverse_iterator` **crbegin** () const
- const `reverse_iterator` **crend** () const
- `iterator` **end** ()
- const `iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const `iterator`, const `iterator` > **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (`iterator` \_\_first, `iterator` \_\_last)
- `iterator` **erase** (`iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **find** (const `key_type` &\_\_x)

- [const\\_iterator](#) **find** (const key\_type &\_\_x) const
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- [iterator](#) **insert** (const value\_type &\_\_x)
- [iterator](#) **insert** (iterator \_\_position, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)
- [const\\_iterator](#) **lower\_bound** (const key\_type &\_\_x) const
- [multimap](#) & **operator=** (const [multimap](#) &\_\_x)
- [multimap](#) & **operator=** ([multimap](#) &&\_\_x)
- [multimap](#) & **operator=** (initializer\_list< value\_type > \_\_l)
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- [reverse\\_iterator](#) **rend** ()
- void **swap** ([multimap](#) &\_\_x)
- [const\\_iterator](#) **upper\_bound** (const key\_type &\_\_x) const
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

### 5.245.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class [std::multimap](#) with safety/checking/debug instrumentation.

Definition at line 44 of file [debug/multimap.h](#).

## 5.245.2 Member Function Documentation

5.245.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

5.245.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.245.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` [protected, inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.245.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( )`  
`const` [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.245.2.5 `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if ( _Predicate __pred )`  
[inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.245.2.6** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.245.2.7** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.245.2.8** `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::M_transfer_iter ( const _Safe_iterator< _Iterator, multimap< _Key, _Tp, _Compare, _Allocator > > & __x ) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.245.3 Member Data Documentation

**5.245.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

**5.245.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

## 5.246 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference 1293

### 5.245.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

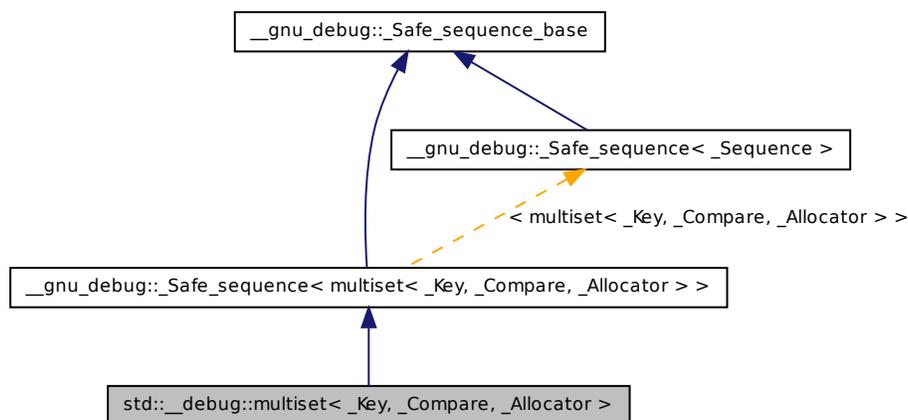
The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

## 5.246 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Class `std::multiset` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**

- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, multiset >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- template<typename `_InputIterator` >  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- **multiset** (const `_Base` &\_\_x)
- **multiset** (`multiset` &&\_\_x)
- **multiset** (const `multiset` &\_\_x)
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare()`, const `allocator_type` &\_\_a=`allocator_type()`)
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `multiset`< `_Key`, `_Compare`, `_Allocator` > > &\_\_x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)

## 5.246 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference 1295

---

- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_x) const
- `iterator erase` (iterator \_\_first, iterator \_\_last)
- `iterator erase` (iterator \_\_position)
- `size_type erase` (const key\_type &\_\_x)
- `iterator find` (const key\_type &\_\_x)
- `const_iterator find` (const key\_type &\_\_x) const
- `iterator insert` (iterator \_\_position, const value\_type &\_\_x)
- `iterator insert` (const value\_type &\_\_x)
- `template<typename _InputIterator > void insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `void insert` (initializer\_list< value\_type > \_\_l)
- `iterator lower_bound` (const key\_type &\_\_x)
- `const_iterator lower_bound` (const key\_type &\_\_x) const
- `multiset & operator=` (const multiset &\_\_x)
- `multiset & operator=` (multiset &&\_\_x)
- `multiset & operator=` (initializer\_list< value\_type > \_\_l)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- `void swap` (multiset &\_\_x)
- `const_iterator upper_bound` (const key\_type &\_\_x) const
- `iterator upper_bound` (const key\_type &\_\_x)

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_revalidate_singular` ()
- `void _M_swap` (\_Safe\_sequence\_base &\_\_x)

### 5.246.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename
_Allocator = std::allocator<_Key>> class std::__debug::multiset< _Key, _-
_Compare, _Allocator >
```

Class `std::multiset` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multiset.h`.

### 5.246.2 Member Function Documentation

**5.246.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.246.2.2** `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
**[protected, inherited]**

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.246.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-`  
`base::_M_get_mutex ( ) throw ()` **[protected,**  
**inherited]**

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`,  
and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.246.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( )`  
**const [inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

## 5.246 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator > Class Template Reference 1297

---

**5.246.2.5** void \_\_gnu\_debug::Safe\_sequence< multiset< \_Key, \_Compare, \_Allocator > >::M\_invalidate\_if ( \_Predicate *pred* )  
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred(x)* returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

**5.246.2.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.246.2.7** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & *x* ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.246.2.8** void \_\_gnu\_debug::Safe\_sequence< multiset< \_Key, \_Compare, \_Allocator > >::M\_transfer\_iter ( const \_Safe\_iterator< \_Iterator, multiset< \_Key, \_Compare, \_Allocator > > & *x* )  
[inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.246.3 Member Data Documentation

**5.246.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

### 5.246.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_` `iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

### 5.246.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` `[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

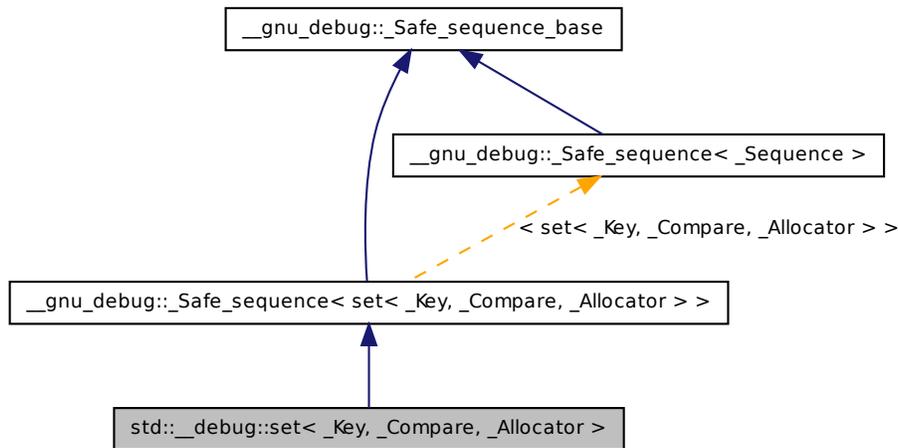
The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

## 5.247 `std::_debug::set<_Key, _Compare, _Allocator` `>` **Class Template Reference**

Class `std::set` with `safety/checking/debug` instrumentation.

Inheritance diagram for `std::__debug::set< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, set >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, set >` **iterator**
  
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **set** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- `template<typename _InputIterator >`  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **set** (const `_Base` &\_\_x)
- **set** (`set` &&\_\_x)
- **set** (const `set` &\_\_x)
- **set** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, `set`< `_Key`, `_Compare`, `_Allocator` > > &\_\_x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (`iterator` \_\_first, `iterator` \_\_last)
- `iterator` **erase** (`iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **find** (const `key_type` &\_\_x)
- `const_iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **insert** (`iterator` \_\_position, const `value_type` &\_\_x)
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &\_\_x)
- `template<typename _InputIterator >`  
void **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **insert** (`initializer_list`< `value_type` > \_\_l)
- `iterator` **lower\_bound** (const `key_type` &\_\_x)
- `const_iterator` **lower\_bound** (const `key_type` &\_\_x) const
- `set` & **operator=** (const `set` &\_\_x)

- `set & operator= (set &&__x)`
- `set & operator= (initializer_list<value_type> __l)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void swap (set &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `iterator upper_bound (const key_type &__x)`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

### 5.247.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__debug::set<_Key, _Compare, _Allocator>`

Class `std::set` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/set.h`.

### 5.247.2 Member Function Documentation

5.247.2.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.247.2.2** `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
**[protected, inherited]**

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**5.247.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` **[protected, inherited]**

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.247.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( ) const` **[inline, inherited]**

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.247.2.5** `void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_invalidate_if ( _Predicate __pred )`  
**[inherited]**

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.247.2.6** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( )`  
**[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.247 `std::__debug::set<_Key, _Compare, _Allocator >` Class Template Reference

1303

**5.247.2.7** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.247.2.8** `void __gnu_debug::Safe_sequence<set<_Key, _Compare, _Allocator >>::_M_transfer_iter ( const _Safe_iterator<_Iterator, set<_Key, _Compare, _Allocator >> & __x ) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.247.3 Member Data Documentation

**5.247.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.247.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.247.3.3** `unsigned int __gnu_debug::Safe_sequence_base::_M_version [mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

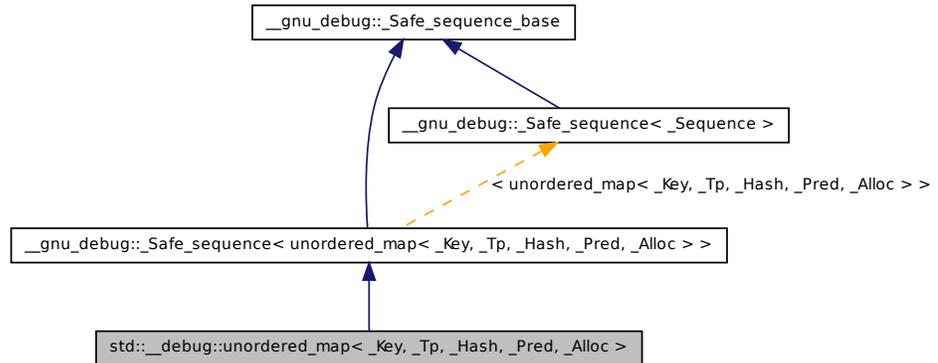
The documentation for this class was generated from the following file:

- [debug/set.h](#)

## 5.248 `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_map >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_map >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- **unordered\_map** ([unordered\\_map](#) &&\_\_x)
- **unordered\_map** (const [unordered\\_map](#) &\_\_x)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const [\\_Safe\\_iterator](#)< [\\_Iterator](#), [unordered\\_map](#)< [\\_Key](#), [\\_Tp](#), [\\_Hash](#), [\\_Pred](#), [\\_Alloc](#) > > &\_\_x)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- [const\\_iterator](#) **cbegin** () const
- [const\\_iterator](#) **cend** () const
- void **clear** ()
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- `std::pair< iterator, iterator >` **equal\_range** (const key\_type &\_\_key)
- `std::pair< const\_iterator, const\_iterator >` **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_it)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator](#) **find** (const key\_type &\_\_key)
- [const\\_iterator](#) **find** (const key\_type &\_\_key) const
- [iterator](#) **insert** ([const\\_iterator](#), const value\_type &\_\_obj)
- `std::pair< iterator, bool >` **insert** (const value\_type &\_\_obj)
- void **insert** (`std::initializer_list`< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [unordered\\_map](#) & **operator=** ([unordered\\_map](#) &&\_\_x)
- [unordered\\_map](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_map](#) & **operator=** (const [unordered\\_map](#) &\_\_x)
- void **swap** ([unordered\\_map](#) &\_\_x)

## Public Attributes

- [\\_Safe\\_iterator\\_base \\* \\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base \\* \\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex & \\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

### 5.248.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>
```

Class [std::unordered\\_map](#) with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_map`.

### 5.248.2 Member Function Documentation

**5.248.2.1** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::\\_M\\_detach\\_all](#) ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.248.2.2** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::\\_M\\_detach\\_singular](#) ( )  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**5.248 std::\_\_debug::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class  
Template Reference 1307**

---

**5.248.2.3** `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw ()` [**protected**, **inherited**]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence >::M_transfer_iter()`.

**5.248.2.4** `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const` [**inline**, **inherited**]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.248.2.5** `void __gnu_debug::Safe_sequence<unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > >::M_invalidate_if ( _Predicate __pred )` [**inherited**]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.248.2.6** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( )` [**protected**, **inherited**]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.248.2.7** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x )` [**protected**, **inherited**]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.248.2.8** `void __gnu_debug::_Safe_sequence< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_transfer_iter ( const _Safe_iterator< _Iterator, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & _x ) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.248.3 Member Data Documentation

**5.248.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.248.3.2** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.248.3.3** `unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file safe\_base.h.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.249 `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, unordered_multimap >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_multimap** (`size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_multimap** (`_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- **unordered\_multimap** (`const _Base &__x`)
- **unordered\_multimap** (`unordered_multimap &&__x`)
- **unordered\_multimap** (`const unordered_multimap &__x`)
- **unordered\_multimap** (`initializer_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`

- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_iter](#) (const \_Safe\_iterator< \_Iterator, [unordered\\_multimap](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > > &\_\_x)
- [iterator begin](#) ()
- [const\\_iterator begin](#) () const
- [const\\_iterator cbegin](#) () const
- [const\\_iterator cend](#) () const
- void [clear](#) ()
- [const\\_iterator end](#) () const
- [iterator end](#) ()
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const key\_type &\_\_key)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > [equal\\_range](#) (const key\_type &\_\_key) const
- size\_type [erase](#) (const key\_type &\_\_key)
- [iterator erase](#) ([const\\_iterator](#) \_\_it)
- [iterator erase](#) ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator find](#) (const key\_type &\_\_key)
- [const\\_iterator find](#) (const key\_type &\_\_key) const
- [iterator insert](#) ([const\\_iterator](#), const value\_type &\_\_obj)
- [iterator insert](#) (const value\_type &\_\_obj)
- void [insert](#) ([std::initializer\\_list](#)< value\_type > \_\_l)
- [template](#)<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [unordered\\_multimap](#) & [operator=](#) ([unordered\\_multimap](#) &&\_\_x)
- [unordered\\_multimap](#) & [operator=](#) ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_multimap](#) & [operator=](#) (const [unordered\\_multimap](#) &\_\_x)
- void [swap](#) ([unordered\\_multimap](#) &\_\_x)

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

### 5.249.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>> class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Definition at line 308 of file `debug/unordered_map`.

### 5.249.2 Member Function Documentation

5.249.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

5.249.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.249.2.3 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw ()` [protected, inherited]

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence >::_M_transfer_iter()`.

5.249.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( )`  
`const` [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.249.2.5** `void __gnu_debug::_Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_if ( _Predicate _pred )` [**inherited**]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.249.2.6** `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )` [**protected, inherited**]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.249.2.7** `void __gnu_debug::_Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x )` [**protected, inherited**]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.249.2.8** `void __gnu_debug::_Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_transfer_iter ( const _Safe_iterator< _Iterator, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > > & __x )` [**inherited**]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.249.3 Member Data Documentation

**5.249.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [**inherited**]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

## 5.250 `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference 1313

### 5.249.3.2 `_Safe_iterator_base*` `__gnu_debug::Safe_sequence_base::M_` `iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

### 5.249.3.3 `unsigned int` `__gnu_debug::Safe_sequence_base::M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.250 `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` `allocator_type`
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_multiset >` `const_iterator`
- typedef `_Base::hasher` `hasher`
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_multiset >` `iterator`
- typedef `_Base::key_equal` `key_equal`
- typedef `_Base::key_type` `key_type`

- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_multiset** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`const _Base &__x`)
- **unordered\_multiset** (`unordered_multiset &&__x`)
- **unordered\_multiset** (`const unordered_multiset &__x`)
- **unordered\_multiset** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eql=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multiset< _Value, _Hash, _Pred, _Alloc > > &__x)`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_iterator end () const`
- `iterator end ()`
- `std::pair< iterator, iterator > equal_range (const key_type &__key)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `iterator erase (const_iterator __it)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__key)`
- `const_iterator find (const key_type &__key) const`
- `iterator insert (const_iterator, const value_type &__obj)`
- `iterator insert (const value_type &__obj)`
- `void insert (std::initializer_list< value_type > __l)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`

## 5.250 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference 1315

---

- `unordered_multiset` & `operator=` (`unordered_multiset` &&\_\_x)
- `unordered_multiset` & `operator=` (`initializer_list`< value\_type > \_\_l)
- `unordered_multiset` & `operator=` (const `unordered_multiset` &\_\_x)
- void `swap` (`unordered_multiset` &\_\_x)

### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &\_\_x)

### 5.250.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 305 of file `debug/unordered_set`.

### 5.250.2 Member Function Documentation

**5.250.2.1** void `__gnu_debug::Safe_sequence_base::_M_detach_all` ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

**5.250.2.2** void `__gnu_debug::Safe_sequence_base::_M_detach_singular` ( )  
[protected, inherited]

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**5.250.2.3** `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex ( ) throw () [protected, inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence >::_M_transfer_iter()`.

**5.250.2.4** `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ( ) const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.250.2.5** `void __gnu_debug::Safe_sequence<unordered_multiset<_Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if ( _Predicate __pred ) [inherited]`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.250.2.6** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.250.2.7** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.250 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference 1317

---

**5.250.2.8** `void __gnu_debug::Safe_sequence< unordered_multiset<_Value, _Hash, _Pred, _Alloc>>::_M_transfer_iter ( const_Safe_iterator<_Iterator, unordered_multiset<_Value, _Hash, _Pred, _Alloc>> &_x )` **[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.250.3 Member Data Documentation

**5.250.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_constant_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

**5.250.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators` **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

**5.250.3.3** `unsigned int __gnu_debug::Safe_sequence_base::_M_version` **[mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

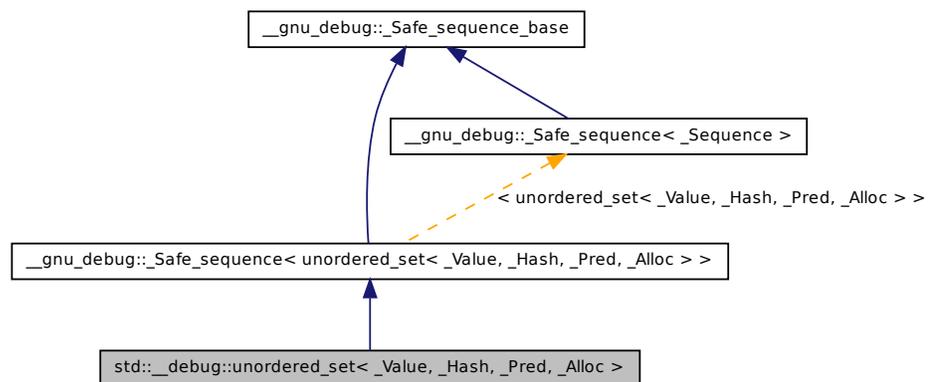
The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.251 `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_set` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, unordered_set >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_set** (`size_type __n=10`, `const hasher &_hf=hasher()`, `const key_equal &_eq=key_equal()`, `const allocator_type &_a=allocator_type()`)

- `template<typename _InputIterator >`  
`unordered_set` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_set` (`const _Base &__x`)
- `unordered_set` (`unordered_set &&__x`)
- `unordered_set` (`const unordered_set &__x`)
- `unordered_set` (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `_Base & _M_base` ()
- `const _Base & _M_base` () `const`
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_transfer_iter` (`const _Safe_iterator< _Iterator, unordered_set<_Value, _Hash, _Pred, _Alloc> > &__x`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `const_iterator end` () `const`
- `iterator end` ()
- `std::pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `iterator erase` (`const_iterator __it`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const_iterator`, `const value_type &__obj`)
- `std::pair< iterator, bool > insert` (`const value_type &__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `unordered_set & operator=` (`unordered_set &&__x`)
- `unordered_set & operator=` (`initializer_list< value_type > __l`)
- `unordered_set & operator=` (`const unordered_set &__x`)
- `void swap` (`unordered_set &__x`)

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

### 5.251.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_set`.

### 5.251.2 Member Function Documentation

**5.251.2.1** `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
**[protected, inherited]**

Detach all iterators, leaving them singular.

**5.251.2.2** `void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )`  
**[protected, inherited]**

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.251 std::\_\_debug::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc > Class  
Template Reference 1321**

---

**5.251.2.3** `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex ( ) throw () [protected, inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence >::M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence >::M_transfer_iter()`.

**5.251.2.4** `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( ) const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

**5.251.2.5** `void __gnu_debug::Safe_sequence< unordered_set<_Value, _Hash, _Pred, _Alloc > >::M_invalidate_if ( _Predicate __pred ) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.251.2.6** `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.251.2.7** `void __gnu_debug::Safe_sequence_base::M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.251.2.8** `void __gnu_debug::_Safe_sequence< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_iter ( const _Safe_iterator< _Iterator, unordered_set< _Value, _Hash, _Pred, _Alloc > > & __x )`  
**[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

### 5.251.3 Member Data Documentation

**5.251.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.251.3.2** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.251.3.3** `unsigned int __gnu_debug::_Safe_sequence_base::_M_version`  
**[mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

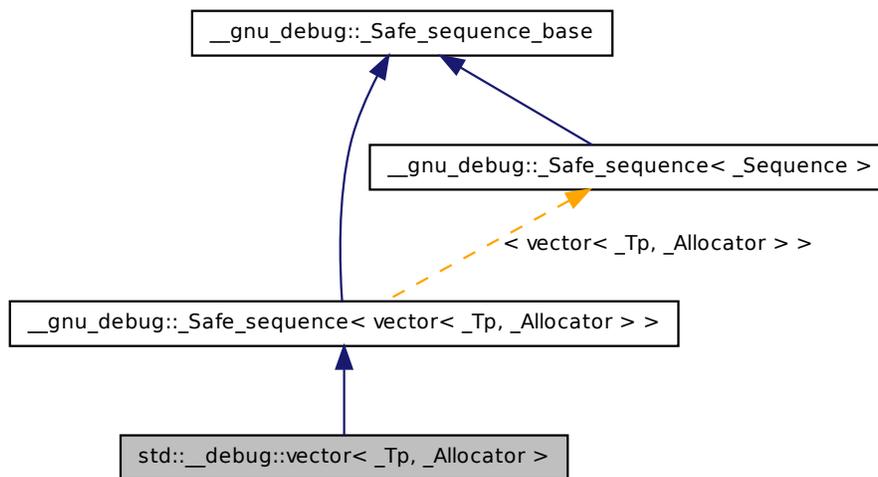
The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.252 std::\_\_debug::vector< \_Tp, \_Allocator > Class Template Reference

Class `std::vector` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::vector< _Tp, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, vector >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**

- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **vector** (const `_Allocator` &\_\_a=\_Allocator())
- **vector** (size\_type \_\_n, const `_Tp` &\_\_value=\_Tp(), const `_Allocator` &\_\_a=\_Allocator())
- **vector** (const **vector** &\_\_x)
- **vector** (const `_Base` &\_\_x)
- template<class `_InputIterator` >  
**vector** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **vector** (**vector** &&\_\_x)
- **vector** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_iter` (const `_Safe_iterator`< `_Iterator`, **vector**< `_Tp`, `_Allocator` > > &\_\_x)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_u)
- template<typename `_InputIterator` >  
void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (`initializer_list`< `value_type` > \_\_l)
- reference **back** ()
- const\_reference **back** () const
- `const_iterator` **begin** () const
- `iterator` **begin** ()
- size\_type **capacity** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- template<typename... `_Args` >  
`iterator` **emplace** (`iterator` \_\_position, `_Args` &&...\_\_args)
- template<typename... `_Args` >  
void **emplace\_back** (`_Args` &&...\_\_args)
- `iterator` **end** ()
- `const_iterator` **end** () const
- `iterator` **erase** (`iterator` \_\_position)

- `iterator erase (iterator __first, iterator __last)`
- reference `front ()`
- `const_reference front () const`
- `iterator insert (iterator __position, const _Tp &__x)`
- `template<typename _Up = _Tp> __gnu_cxx::__enable_if<!std::__are_same<_Up, bool >::__value, iterator >::__type insert (iterator __position, _Tp &&__x)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `void insert (iterator __position, initializer_list<value_type > __l)`
- `template<class _InputIterator > void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `vector & operator= (const vector &__x)`
- `vector & operator= (initializer_list<value_type > __l)`
- `vector & operator= (vector &&__x)`
- `const_reference operator[] (size_type __n) const`
- `reference operator[] (size_type __n)`
- `void pop_back ()`
- `template<typename _Up = _Tp> __gnu_cxx::__enable_if<!std::__are_same<_Up, bool >::__value, void >::__type push_back (_Tp &&__x)`
- `void push_back (const _Tp &__x)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void reserve (size_type __n)`
- `void resize (size_type __sz, _Tp __c=_Tp())`
- `void swap (vector &__x)`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

### 5.252.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::vector< _Tp, _Allocator >
```

Class [std::vector](#) with safety/checking/debug instrumentation.

Definition at line 45 of file `debug/vector`.

### 5.252.2 Constructor & Destructor Documentation

```
5.252.2.1 template<typename _Tp, typename _Allocator =
std::allocator<_Tp>> std::__debug::vector< _Tp, _Allocator
>::vector ( const_Base & __x ) [inline]
```

Construction from a release-mode vector.

Definition at line 94 of file `debug/vector`.

### 5.252.3 Member Function Documentation

```
5.252.3.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ( )
[protected, inherited]
```

Detach all iterators, leaving them singular.

```
5.252.3.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )
[protected, inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

```
5.252.3.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-
base::_M_get_mutex ( ) throw () [protected,
inherited]
```

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

**5.252.3.4** void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( )  
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe\_base.h.

**5.252.3.5** void \_\_gnu\_debug::Safe\_sequence< vector<\_Tp, \_Allocator >  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators  $x$  that reference this sequence, are not singular, and for which  $\text{pred}(x)$  returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

**5.252.3.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.252.3.7** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap (   
\_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.252.3.8** void \_\_gnu\_debug::Safe\_sequence< vector<\_Tp, \_Allocator >  
>::M\_transfer\_iter ( const \_Safe\_iterator< \_Iterator, vector<\_Tp, \_Allocator > > & \_\_x ) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

## 5.252.4 Member Data Documentation

**5.252.4.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

#### 5.252.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

#### 5.252.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

## 5.253 `std::__declval_protector<_Tp >` Struct Template Reference

`declval`

### Static Public Member Functions

- static [add\\_rvalue\\_reference<\\_Tp >::type \\_\\_delegate\(\)](#)

### Static Public Attributes

- static const bool `__stop`

### 5.253.1 Detailed Description

```
template<typename _Tp> struct std::__declval_protector< _Tp >
```

declval

Definition at line 616 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.254 `std::__exception_ptr::exception_ptr` Class Reference

An opaque pointer to an arbitrary exception.

### Public Types

- typedef `void(exception_ptr::* __safe_bool )()`

### Public Member Functions

- `exception_ptr` (`__safe_bool`) throw ()
- `exception_ptr` (`exception_ptr` &&\_\_o) throw ()
- `exception_ptr` (const `exception_ptr` &) throw ()
- const `type_info` \* `__cxa_exception_type` () const `__attribute__((__pure__))` throw ()
- `exception_ptr` & `operator=` (const `exception_ptr` &) throw ()
- `exception_ptr` & `operator=` (`exception_ptr` &&\_\_o) throw ()
- void `swap` (`exception_ptr` &) throw ()

### Friends

- bool `operator==` (const `exception_ptr` &, const `exception_ptr` &) `__attribute__((__pure__))` throw ()
- `exception_ptr` `std::current_exception` () throw ()
- void `std::rethrow_exception` (`exception_ptr`)

### 5.254.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file exception\_ptr.h.

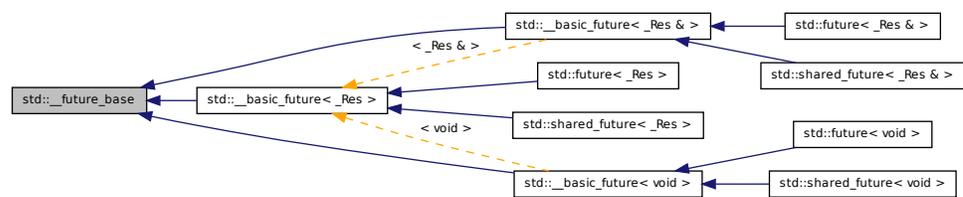
The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

### 5.255 std::\_\_future\_base Struct Reference

Base class and enclosing scope.

Inheritance diagram for std::\_\_future\_base:



### Classes

- [struct \\_Ptr](#)  
A *unique\_ptr* based on the instantiating type.
- [struct \\_Result](#)  
*Result.*
- [struct \\_Result<\\_Res & >](#)  
*Partial specialization for reference types.*
- [struct \\_Result<void >](#)  
*Explicit specialization for void.*
- [struct \\_Result\\_base](#)  
*Base class for results.*

- class `_State`

*Shared state between a promise and one or more associated futures.*

### 5.255.1 Detailed Description

Base class and enclosing scope.

Definition at line 136 of file `future`.

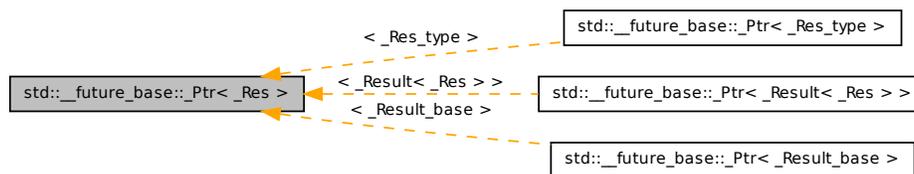
The documentation for this struct was generated from the following file:

- `future`

## 5.256 `std::__future_base::_Ptr<_Res>` Struct Template Reference

A `unique_ptr` based on the instantiating type.

Inheritance diagram for `std::__future_base::_Ptr<_Res>`:



### Public Types

- typedef `unique_ptr<_Res, _Result_base::_Deleter>` **type**

### 5.256.1 Detailed Description

```
template<typename _Res> struct std::__future_base::_Ptr<_Res>
```

A `unique_ptr` based on the instantiating type.

Definition at line 211 of file future.

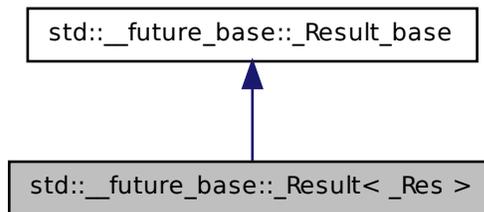
The documentation for this struct was generated from the following file:

- [future](#)

## 5.257 `std::__future_base::_Result< _Res >` Struct Template Reference

Result.

Inheritance diagram for `std::__future_base::_Result< _Res >`:



### Public Member Functions

- `void _M_set (const _Res &__res)`
- `void _M_set (_Res &&__res)`
- `_Res & _M_value ()`

### Public Attributes

- `exception_ptr _M_error`

### 5.257.1 Detailed Description

```
template<typename _Res> struct std::__future_base::_Result<_Res >
```

Result.

Definition at line 161 of file future.

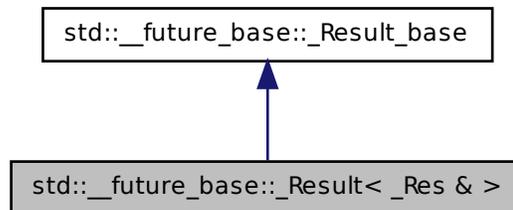
The documentation for this struct was generated from the following file:

- [future](#)

## 5.258 `std::__future_base::_Result<_Res &>` Struct Template Reference

Partial specialization for reference types.

Inheritance diagram for `std::__future_base::_Result<_Res &>`:



### Public Member Functions

- `_Res & _M_get ()`
- `void _M_set (_Res &__res)`

### Public Attributes

- `exception_ptr _M_error`

### 5.258.1 Detailed Description

```
template<typename _Res> struct std::_future_base::_Result< _Res & >
```

Partial specialization for reference types.

Definition at line 455 of file future.

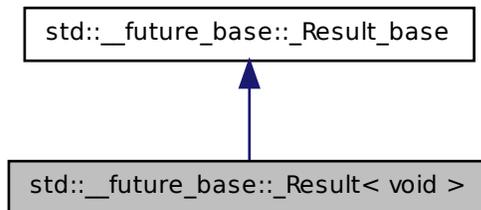
The documentation for this struct was generated from the following file:

- [future](#)

### 5.259 std::\_future\_base::\_Result< void > Struct Template Reference

Explicit specialization for void.

Inheritance diagram for std::\_future\_base::\_Result< void >:



#### Public Attributes

- `exception_ptr _M_error`

### 5.259.1 Detailed Description

```
template<> struct std::_future_base::_Result< void >
```

Explicit specialization for void.

Definition at line 471 of file future.

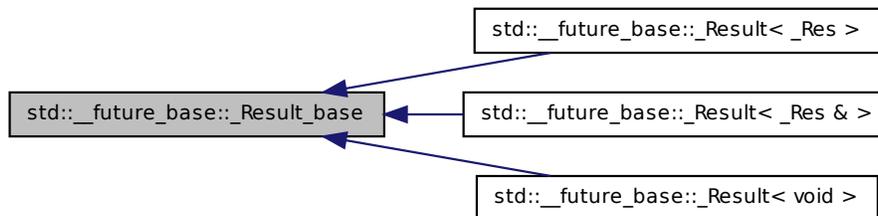
The documentation for this struct was generated from the following file:

- [future](#)

## 5.260 `std::__future_base::_Result_base` Struct Reference

Base class for results.

Inheritance diagram for `std::__future_base::_Result_base`:



### Public Member Functions

- `_Result_base` (const `_Result_base` &)
- virtual void `_M_destroy` ()=0
- `_Result_base` & `operator=` (const `_Result_base` &)

### Public Attributes

- `exception_ptr` `_M_error`

#### 5.260.1 Detailed Description

Base class for results.

Definition at line 139 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.261 `std::__future_base::_State` Class Reference

Shared state between a promise and one or more associated futures.

Inherited by `std::__future_base::Async_state< _Res >`, `std::__future_base::_Deferred_state< _Res >`, and `std::__future_base::_Task_state< _Res(_Args...)>`.

### Public Member Functions

- `_State` (const `_State` &)
- `void _M_break_promise` (`_Ptr_type __res`)
- `void _M_set_result` (`function< _Ptr_type()> __res`, `bool __ignore_failure=false`)
- `void _M_set_retrieved_flag` ()
- `_State` & `operator=` (const `_State` &)
- `_Result_base` & `wait` ()
- `template<typename _Rep, typename _Period >`  
`bool wait_for` (const `chrono::duration< _Rep, _Period >` & `__rel`)
- `template<typename _Clock, typename _Duration >`  
`bool wait_until` (const `chrono::time_point< _Clock, _Duration >` & `__abs`)

### Static Public Member Functions

- `template<typename _Res, typename _Arg >`  
`static _Setter< _Res, _Arg && >` `__setter` (`promise< _Res > * __prom`, `_Arg && __arg`)
- `template<typename _Res >`  
`static _Setter< _Res, __exception_ptr_tag >` `__setter` (`exception_ptr & __ex`, `promise< _Res > * __prom`)
- `static _Setter< void, void >` `__setter` (`promise< void > * __prom`)
- `template<typename _Tp >`  
`static bool _S_check` (const `shared_ptr< _Tp >` & `__p`)

#### 5.261.1 Detailed Description

Shared state between a promise and one or more associated futures.

Definition at line 262 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## **5.262 `std::__is_location_invariant< _Tp >` Struct Template Reference**

Inherits `integral_constant< bool,(is_pointer< _Tp >::value||is_member_pointer< _Tp >::value)>`.

Inherited by `std::__is_location_invariant< _Simple_type_wrapper< _Tp > >`.

### **5.262.1 Detailed Description**

**`template<typename _Tp> struct std::__is_location_invariant< _Tp >`**

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

Definition at line 1432 of file `functional`.

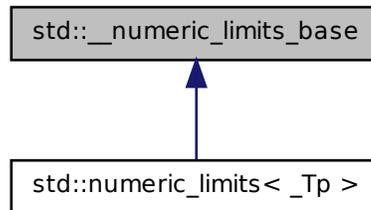
The documentation for this struct was generated from the following file:

- [functional](#)

## **5.263 `std::__numeric_limits_base` Struct Reference**

Part of [std::numeric\\_limits](#).

Inheritance diagram for `std::_numeric_limits_base`:



### Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const [float\\_denorm\\_style](#) [has\\_denorm](#)
- static const bool [has\\_denorm\\_loss](#)
- static const bool [has\\_infinity](#)
- static const bool [has\\_quiet\\_NaN](#)
- static const bool [has\\_signaling\\_NaN](#)
- static const bool [is\\_bounded](#)
- static const bool [is\\_exact](#)
- static const bool [is\\_iec559](#)
- static const bool [is\\_integer](#)
- static const bool [is\\_modulo](#)
- static const bool [is\\_signed](#)
- static const bool [is\\_specialized](#)
- static const int [max\\_digits10](#)
- static const int [max\\_exponent](#)
- static const int [max\\_exponent10](#)
- static const int [min\\_exponent](#)
- static const int [min\\_exponent10](#)
- static const int [radix](#)
- static const [float\\_round\\_style](#) [round\\_style](#)
- static const bool [tinyness\\_before](#)
- static const bool [traps](#)

### 5.263.1 Detailed Description

Part of `std::numeric_limits`. The `static const` members are usable as integral constant expressions.

#### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

Definition at line 190 of file `limits`.

### 5.263.2 Member Data Documentation

#### 5.263.2.1 `const int std::__numeric_limits_base::digits` [`static`]

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file `limits`.

#### 5.263.2.2 `const int std::__numeric_limits_base::digits10` [`static`]

The number of base 10 digits that can be represented without change.

Definition at line 201 of file `limits`.

#### 5.263.2.3 `const float_denorm_style std::__numeric_limits_base::has_denorm` [`static`]

See `std::float_denorm_style` for more information.

Definition at line 244 of file `limits`.

#### 5.263.2.4 `const bool std::__numeric_limits_base::has_denorm_loss` [`static`]

*True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42*

Definition at line 247 of file `limits`.

**5.263.2.5 const bool std::\_\_numeric\_limits\_base::has\_infinity [static]**

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

**5.263.2.6 const bool std::\_\_numeric\_limits\_base::has\_quiet\_NaN [static]**

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

**5.263.2.7 const bool std::\_\_numeric\_limits\_base::has\_signaling\_NaN [static]**

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

**5.263.2.8 const bool std::\_\_numeric\_limits\_base::is\_bounded [static]**

*True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54*

Definition at line 255 of file limits.

**5.263.2.9 const bool std::\_\_numeric\_limits\_base::is\_exact [static]**

True if the type uses an exact representation. *All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15*

Definition at line 216 of file limits.

**5.263.2.10 const bool std::\_\_numeric\_limits\_base::is\_iec559 [static]**

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

**5.263.2.11 const bool std::\_\_numeric\_limits\_base::is\_integer [static]**

True if the type is integer. Is this supposed to be *if the type is integral?*

Definition at line 211 of file limits.

#### 5.263.2.12 const bool std::\_\_numeric\_limits\_base::is\_modulo [static]

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

#### 5.263.2.13 const bool std::\_\_numeric\_limits\_base::is\_signed [static]

True if the type is signed.

Definition at line 208 of file limits.

#### 5.263.2.14 const bool std::\_\_numeric\_limits\_base::is\_specialized [static]

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

#### 5.263.2.15 const int std::\_\_numeric\_limits\_base::max\_digits10 [static]

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

#### 5.263.2.16 const int std::\_\_numeric\_limits\_base::max\_exponent [static]

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file limits.

#### 5.263.2.17 const int std::\_\_numeric\_limits\_base::max\_exponent10 [static]

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file limits.

**5.263.2.18** `const int std::__numeric_limits_base::min_exponent` `[static]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file `limits`.

**5.263.2.19** `const int std::__numeric_limits_base::min_exponent10` `[static]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file `limits`.

**5.263.2.20** `const int std::__numeric_limits_base::radix` `[static]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file `limits`.

**5.263.2.21** `const float_round_style std::__numeric_limits_base::round_style`  
`[static]`

See [std::float\\_round\\_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file `limits`.

**5.263.2.22** `const bool std::__numeric_limits_base::tinyness_before` `[static]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file `limits`.

**5.263.2.23** `const bool std::__numeric_limits_base::traps` `[static]`

True if trapping is implemented for this type.

Definition at line 263 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.264 `std::__parallel::CRandNumber<_MustBeInt>` Struct Template Reference

Functor wrapper for `std::rand()`.

### Public Member Functions

- `int operator() (int __limit)`

#### 5.264.1 Detailed Description

```
template<typename _MustBeInt = int> struct std::__parallel::_CRandNumber<_MustBeInt>
```

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

## 5.265 `std::__profile::bitset<_Nb>` Class Template Reference

Class `std::bitset` wrapper with performance instrumentation.

Inherits `bitset<_Nb>`.

### Public Member Functions

- `bitset (unsigned long long __val)`
- `template<class _CharT, class _Traits, class _Alloc> bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))`
- `bitset (const _Base &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos=0, type-`

name `std::basic_string<_CharT, _Traits, _Alloc >::size_type __n=(std::basic_string<_CharT, _Traits, _Alloc >::npos)`

- `bitset` (const char \*\_\_str)
- `_Base & _M_base` ()
- const `_Base & _M_base` () const
- `bitset<_Nb > & flip` ()
- `bitset<_Nb > & flip` (size\_t \_\_pos)
- bool `operator!=` (const `bitset<_Nb > &__rhs`) const
- `bitset<_Nb > & operator&=` (const `bitset<_Nb > &__rhs`)
- `bitset<_Nb > operator<<` (size\_t \_\_pos) const
- `bitset<_Nb > & operator<<=` (size\_t \_\_pos)
- bool `operator==` (const `bitset<_Nb > &__rhs`) const
- `bitset<_Nb > operator>>` (size\_t \_\_pos) const
- `bitset<_Nb > & operator>>=` (size\_t \_\_pos)
- reference `operator[]` (size\_t \_\_pos)
- bool `operator[]` (size\_t \_\_pos) const
- `bitset<_Nb > & operator^=` (const `bitset<_Nb > &__rhs`)
- `bitset<_Nb > & operator|=` (const `bitset<_Nb > &__rhs`)
- `bitset<_Nb > operator~` () const
- `bitset<_Nb > & reset` ()
- `bitset<_Nb > & reset` (size\_t \_\_pos)
- `bitset<_Nb > & set` (size\_t \_\_pos, bool \_\_val=true)
- `bitset<_Nb > & set` ()
- `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_string` (char \_\_zero, char \_\_one= '1') const
- `template<typename _CharT, typename _Traits > std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string` () const
- `template<typename _CharT, typename _Traits, typename _Alloc > std::basic_string< _CharT, _Traits, _Alloc > to_string` () const
- `template<class _CharT, class _Traits > std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_string` () const
- `template<typename _CharT > std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > to_string` () const
- `template<class _CharT > std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- `template<class _CharT, class _Traits, class _Alloc > std::basic_string< _CharT, _Traits, _Alloc > to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const

### 5.265.1 Detailed Description

`template<size_t _Nb> class std::__profile::bitset< _Nb >`

Class `std::bitset` wrapper with performance instrumentation.

Definition at line 40 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

## 5.266 `std::__profile::deque< _Tp, _Allocator >` Class Template Reference

Class `std::deque` wrapper with performance instrumentation.

Inherits `deque< _Tp, _Allocator >`.

### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **deque** (`const _Allocator &__a=_Allocator()`)
- **deque** (`size_type __n, const _Tp &__value=_Tp(), const _Allocator &__a=_Allocator()`)
- **deque** (`const deque &__x`)
- **deque** (`const _Base &__x`)

- `template<class _InputIterator >`  
**deque** (`_InputIterator __first`, `_InputIterator __last`, `const _Allocator &__a=_`  
`Allocator()`)
- **deque** (`deque &&__x`)
- **deque** (`initializer_list< value_type > __l`, `const allocator_type &__`  
`a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void assign (size_type __n, const _Tp &__t)`
- `template<class _InputIterator >`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- reference **back** ()
- `const_reference back () const`
- `const_iterator begin () const`
- `iterator begin ()`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `template<typename... _Args>`  
`iterator emplace (iterator __position, _Args &&...__args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&...__args)`
- `template<typename... _Args>`  
`void emplace_front (_Args &&...__args)`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __position)`
- `iterator erase (iterator __first, iterator __last)`
- `const_reference front () const`
- reference **front** ()
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `iterator insert (iterator __position, _Tp &&__x)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `deque & operator= (deque &&__x)`
- `deque & operator= (const deque &__x)`
- `deque & operator= (initializer_list< value_type > __l)`
- `const_reference operator[] (size_type __n) const`

- reference `operator[]` (size\_type \_\_n)
- void `pop_back` ()
- void `pop_front` ()
- void `push_back` (const \_Tp &\_\_x)
- void `push_back` (\_Tp &&\_\_x)
- void `push_front` (const \_Tp &\_\_x)
- void `push_front` (\_Tp &&\_\_x)
- reverse\_iterator `rbegin` ()
- const\_reverse\_iterator `rbegin` () const
- const\_reverse\_iterator `rend` () const
- reverse\_iterator `rend` ()
- void `resize` (size\_type \_\_sz, \_Tp \_\_c=\_Tp())
- void `swap` ([deque](#) &\_\_x)

### 5.266.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::deque< _Tp, _Allocator >`

Class [std::deque](#) wrapper with performance instrumentation.

Definition at line 40 of file `profile/deque`.

The documentation for this class was generated from the following file:

- [profile/deque](#)

## 5.267 `std::__profile::list< _Tp, _Allocator >` Class Template Reference

List wrapper with performance instrumentation.

Inherits `list< _Tp, _Allocator >`.

### Public Types

- typedef `_Allocator` `allocator_type`
- typedef `__iterator_tracker< typename _Base::const_iterator, list >` `const_iterator`
- typedef `_Base::const_pointer` `const_pointer`
- typedef `_Base::const_reference` `const_reference`
- typedef `std::reverse_iterator< const_iterator >` `const_reverse_iterator`

- typedef `_Base::difference_type` **difference\_type**
- typedef `__iterator_tracker< typename _Base::iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **list** (const `_Allocator` &`__a`=`_Allocator`())
- **list** (size\_type `__n`, const `_Tp` &`__value`=`_Tp`(), const `_Allocator` &`__a`=`_Allocator`())
- **list** (const `list` &`__x`)
- **list** (const `_Base` &`__x`)
- template<class `_InputIterator` >  
**list** (`_InputIterator` `__first`, `_InputIterator` `__last`, const `_Allocator` &`__a`=`_Allocator`())
- **list** (`list` &&`__x`)
- **list** (`initializer_list`< `value_type` > `__l`, const `allocator_type` &`__a`=`allocator_type`())
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_profile_find` () const
- void `_M_profile_iterate` (int `__rewind`=0) const
- void **assign** (`initializer_list`< `value_type` > `__l`)
- template<class `_InputIterator` >  
void **assign** (`_InputIterator` `__first`, `_InputIterator` `__last`)
- void **assign** (size\_type `__n`, const `_Tp` &`__t`)
- const\_reference **back** () const
- reference **back** ()
- const\_iterator **begin** () const
- iterator **begin** ()
- const\_iterator **cbegin** () const
- const\_iterator **kend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- template<typename... `_Args`>  
iterator **emplace** (iterator `__position`, `_Args` &&...`__args`)
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator `__position`)

- iterator **erase** (iterator \_\_position, iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- iterator **insert** (iterator \_\_position, const \_Tp &\_\_x)
- iterator **insert** (iterator \_\_position, \_Tp &&\_\_x)
- void **insert** (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- void **insert** (iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Compare >  
void **merge** (list &\_\_x, \_Compare \_\_comp)
- void **merge** (list &\_\_x)
- void **merge** (list &&\_\_x)
- template<class \_Compare >  
void **merge** (list &&\_\_x, \_Compare \_\_comp)
- list & **operator=** (const list &\_\_x)
- list & **operator=** (list &&\_\_x)
- list & **operator=** (initializer\_list< value\_type > \_\_l)
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_front** (const value\_type &\_\_x)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- void **remove** (const \_Tp &\_\_value)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- void **resize** (size\_type \_\_sz, \_Tp \_\_c=\_Tp())
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **sort** ()
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &&\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_first, iterator \_\_last)
- void **swap** (list &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_binary\_pred)

### 5.267.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__profile::list< _Tp, _Allocator >
```

List wrapper with performance instrumentation.

Definition at line 42 of file profile/list.

The documentation for this class was generated from the following file:

- [profile/list](#)

### 5.268 `std::__profile::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::map` wrapper with performance instrumentation.

Inherits `map< _Key, _Tp, _Compare, _Allocator >`.

#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

#### Public Member Functions

- **map** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())

## 5.268 `std::__profile::map<_Key, _Tp, _Compare, _Allocator>` Class Template Reference 1351

---

- `template<typename _InputIterator >`  
`map` (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator()`)
- `map` (`const _Base &__x`)
- `map` (`map &&__x`)
- `map` (`const map &__x`)
- `map` (`initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k) const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `size_type count (const key_type &__x) const`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `iterator end ()`
- `const_iterator end () const`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `map & operator= (const map &__x)`
- `map & operator= (initializer_list< value_type > __l)`
- `map & operator= (map &&__x)`
- `mapped_type & operator[] (const key_type &__k)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`

- [reverse\\_iterator](#) **rend** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- void **swap** ([map](#) &\_\_x)
- [const\\_iterator](#) **upper\_bound** (const key\_type &\_\_x) const
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)

### 5.268.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class [std::map](#) wrapper with performance instrumentation.

Definition at line 47 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

### 5.269 [std::\\_\\_profile::multimap](#)< [\\_Key](#), [\\_Tp](#), [\\_Compare](#), [\\_Allocator](#) > Class Template Reference

Class [std::multimap](#) wrapper with performance instrumentation.

Inherits [multimap](#)< [\\_Key](#), [\\_Tp](#), [\\_Compare](#), [\\_Allocator](#) >.

#### Public Types

- typedef [\\_Allocator](#) **allocator\_type**
- typedef [\\_Base::const\\_iterator](#) **const\_iterator**
- typedef [\\_Base::const\\_pointer](#) **const\_pointer**
- typedef [\\_Base::const\\_reference](#) **const\_reference**
- typedef [\\_Base::const\\_reverse\\_iterator](#) **const\_reverse\_iterator**
- typedef [\\_Base::difference\\_type](#) **difference\_type**
- typedef [\\_Base::iterator](#) **iterator**
- typedef [\\_Compare](#) **key\_compare**
- typedef [\\_Key](#) **key\_type**
- typedef [\\_Tp](#) **mapped\_type**
- typedef [\\_Base::pointer](#) **pointer**
- typedef [\\_Base::reference](#) **reference**

- `typedef _Base::reverse_iterator reverse_iterator`
- `typedef _Base::size_type size_type`
- `typedef std::pair< const _Key, _Tp > value_type`

## Public Member Functions

- **multimap** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- `template<typename _InputIterator >`  
**multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multimap** (const [\\_Base](#) &\_\_x)
- **multimap** ([multimap](#) &&\_\_x)
- **multimap** (const [multimap](#) &\_\_x)
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_c=\_Compare(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- iterator **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- iterator **end** ()
- const\_iterator **end** () const
- [std::pair](#)< iterator, iterator > **equal\_range** (const [key\\_type](#) &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const [key\\_type](#) &\_\_x) const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
- iterator **find** (const [key\\_type](#) &\_\_x)
- const\_iterator **find** (const [key\\_type](#) &\_\_x) const
- iterator **insert** (const [value\\_type](#) &\_\_x)
- void **insert** ([std::initializer\\_list](#)< [value\\_type](#) > \_\_list)
- iterator **insert** (iterator \_\_position, const [value\\_type](#) &\_\_x)
- `template<typename _InputIterator >`  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **lower\_bound** (const [key\\_type](#) &\_\_x)
- const\_iterator **lower\_bound** (const [key\\_type](#) &\_\_x) const
- [multimap](#) & **operator=** (const [multimap](#) &\_\_x)

- `multimap` & `operator=` (`initializer_list`< `value_type` > `__l`)
- `multimap` & `operator=` (`multimap` &&`__x`)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- `void` `swap` (`multimap` &`__x`)
- `iterator` `upper_bound` (const `key_type` &`__x`)
- `const_iterator` `upper_bound` (const `key_type` &`__x`) const

### 5.269.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` wrapper with performance instrumentation.

Definition at line 41 of file `profile/multimap.h`.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

### 5.270 `std::__profile::multiset`< `_Key`, `_Compare`, `_Allocator` > Class Template Reference

Class `std::multiset` wrapper with performance instrumentation.

Inherits `multiset`< `_Key`, `_Compare`, `_Allocator` >.

#### Public Types

- `typedef` `_Allocator` `allocator_type`
- `typedef` `_Base::const_iterator` `const_iterator`
- `typedef` `_Base::const_pointer` `const_pointer`
- `typedef` `_Base::const_reference` `const_reference`
- `typedef` `_Base::const_reverse_iterator` `const_reverse_iterator`
- `typedef` `_Base::difference_type` `difference_type`
- `typedef` `_Base::iterator` `iterator`
- `typedef` `_Compare` `key_compare`
- `typedef` `_Key` `key_type`

- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- template<typename `_InputIterator` >  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- **multiset** (const `_Base` &\_\_x)
- **multiset** (`multiset` &&\_\_x)
- **multiset** (const `multiset` &\_\_x)
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare()`, const `allocator_type` &\_\_a=`allocator_type()`)
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **kend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- iterator **end** ()
- const\_iterator **end** () const
- `std::pair`< iterator, iterator > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > **equal\_range** (const `key_type` &\_\_x) const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- size\_type **erase** (const `key_type` &\_\_x)
- iterator **find** (const `key_type` &\_\_x)
- const\_iterator **find** (const `key_type` &\_\_x) const
- iterator **insert** (const `value_type` &\_\_x)
- iterator **insert** (iterator \_\_position, const `value_type` &\_\_x)
- template<typename `_InputIterator` >  
void **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **insert** (`initializer_list`< `value_type` > \_\_l)

- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- **multiset** & **operator=** (const **multiset** &\_\_x)
- **multiset** & **operator=** (**initializer\_list**< value\_type > \_\_l)
- **multiset** & **operator=** (**multiset** &&\_\_x)
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rbegin** ()
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- void **swap** (**multiset** &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### 5.270.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename
_Allocator = std::allocator<_Key>> class std::__profile::multiset< _Key, _-
_Compare, _Allocator >
```

Class [std::multiset](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

### 5.271 [std::\\_\\_profile::set< \\_Key, \\_Compare, \\_Allocator >](#) Class Template Reference

Class [std::set](#) wrapper with performance instrumentation.

Inherits [set< \\_Key, \\_Compare, \\_Allocator >](#).

#### Public Types

- typedef [\\_Allocator](#) **allocator\_type**
- typedef [\\_Base::const\\_iterator](#) **const\_iterator**
- typedef [\\_Base::const\\_pointer](#) **const\_pointer**
- typedef [\\_Base::const\\_reference](#) **const\_reference**
- typedef [\\_Base::const\\_reverse\\_iterator](#) **const\_reverse\_iterator**
- typedef [\\_Base::difference\\_type](#) **difference\_type**

- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **set** (const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- `template<typename _InputIterator >`  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- **set** (const `_Base` &\_\_x)
- **set** (`set` &&\_\_x)
- **set** (const `set` &\_\_x)
- **set** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **rend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- iterator **end** ()
- const\_iterator **end** () const
- `std::pair`< iterator, iterator > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > **equal\_range** (const `key_type` &\_\_x) const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- iterator **find** (const `key_type` &\_\_x)
- const\_iterator **find** (const `key_type` &\_\_x) const
- `std::pair`< iterator, bool > **insert** (const `value_type` &\_\_x)

- iterator **insert** (iterator \_\_position, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- **set** & **operator=** (const **set** &\_\_x)
- **set** & **operator=** (initializer\_list< value\_type > \_\_l)
- **set** & **operator=** (**set** &&\_\_x)
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rbegin** ()
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- void **swap** (**set** &\_\_x)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### 5.271.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>> class std::\_\_profile::set< \_Key, \_Compare, \_Allocator >

Class [std::set](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

### 5.272 std::\_\_profile::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Class [std::unordered\\_map](#) wrapper with performance instrumentation.

Inherits [\\_GLIBCXX\\_STD\\_PR::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#).

#### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**

- typedef **\_Base::const\_reference** **const\_reference**
- typedef **\_Base::difference\_type** **difference\_type**
- typedef **\_Base::hasher** **hasher**
- typedef **\_Base::iterator** **iterator**
- typedef **\_Base::key\_equal** **key\_equal**
- typedef **\_Base::key\_type** **key\_type**
- typedef **\_Base::mapped\_type** **mapped\_type**
- typedef **\_Base::reference** **reference**
- typedef **\_Base::size\_type** **size\_type**
- typedef **\_Base::value\_type** **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename **\_InputIterator** >  
**unordered\_map** (**\_InputIterator** \_\_f, **\_InputIterator** \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** ([unordered\\_map](#) &&\_\_x)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eql=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const **\_Base** &\_\_x)
- **\_Base** & **\_M\_base** ()
- const **\_Base** & **\_M\_base** () const
- void **clear** ()
- template<typename **\_InputIter** >  
void **insert** (**\_InputIter** \_\_first, **\_InputIter** \_\_last)
- void **insert** (const value\_type \* \_\_first, const value\_type \* \_\_last)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- [unordered\\_map](#) & **operator=** ([unordered\\_map](#) &&\_\_x)
- [unordered\\_map](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_map](#) & **operator=** (const [unordered\\_map](#) &\_\_x)
- mapped\_type & **operator[]** (const **\_Key** &\_\_k)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_map](#) &\_\_x)

### 5.272.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_map` wrapper with performance instrumentation.

Definition at line 56 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

### 5.273 `std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

#### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

#### Public Member Functions

- `unordered_multimap (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`

## 5.274 `std::__profile::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference 1361

- `template<typename _InputIterator >`  
`unordered_multimap` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_multimap` (`unordered_multimap &&__x`)
- `unordered_multimap` (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_multimap` (`const _Base &__x`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void clear ()`
- `void insert (const value_type * __first, const value_type * __last)`
- `template<typename _InputIter >`  
`void insert (_InputIter __first, _InputIter __last)`
- `iterator insert (const_iterator __iter, const value_type & __v)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const value_type & __obj)`
- `unordered_multimap & operator= (unordered_multimap && __x)`
- `unordered_multimap & operator= (initializer_list< value_type > __l)`
- `unordered_multimap & operator= (const unordered_multimap & __x)`
- `void rehash (size_type __n)`
- `void swap (unordered_multimap & __x)`

### 5.273.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multimap` wrapper with performance instrumentation.

Definition at line 296 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

## 5.274 `std::__profile::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

`Unordered_multiset` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_multiset** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`unordered_multiset &&__x`)
- **unordered\_multiset** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`const _Base &__x`)
- `void clear ()`
- `template<typename _InputIter >`  
`void insert (_InputIter __first, _InputIter __last)`
- `void insert (const value_type *__first, const value_type *__last)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const value_type &__obj)`
- `iterator insert (const_iterator __iter, const value_type &__v)`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `unordered_multiset & operator= (unordered_multiset &&__x)`
- `unordered_multiset & operator= (const unordered_multiset &__x)`
- `void rehash (size_type __n)`
- `void swap (unordered_multiset &__x)`

### 5.274.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Unordered\_multiset wrapper with performance instrumentation.

Definition at line 284 of file `profile/unordered_set`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 5.275 `std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >` Class Template Reference

Unordered\_set wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_set** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_set** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)

- **unordered\_set** ([unordered\\_set](#) &&\_\_x)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const \_Base &\_\_x)
- void **clear** ()
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \* \_\_first, const value\_type \* \_\_last)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- [unordered\\_set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_set](#) & **operator=** ([unordered\\_set](#) &&\_\_x)
- [unordered\\_set](#) & **operator=** (const [unordered\\_set](#) &\_\_x)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_set](#) &\_\_x)

### 5.275.1 Detailed Description

```
template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >
```

Unordered\_set wrapper with performance instrumentation.

Definition at line 56 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

### 5.276 std::\_Base\_bitset< \_Nw > Struct Template Reference

Inheritance diagram for std::\_Base\_bitset< \_Nw >:



## Public Types

- typedef unsigned long **\_WordT**

## Public Member Functions

- **\_Base\_bitset** (unsigned long long \_\_val)
- size\_t **\_M\_are\_all\_aux** () const
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)<\_Nw> &\_\_x)
- size\_t **\_M\_do\_count** () const
- size\_t **\_M\_do\_find\_first** (size\_t \_\_not\_found) const
- size\_t **\_M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const
- void **\_M\_do\_flip** ()
- void **\_M\_do\_left\_shift** (size\_t \_\_shift)
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)<\_Nw> &\_\_x)
- void **\_M\_do\_reset** ()
- void **\_M\_do\_right\_shift** (size\_t \_\_shift)
- void **\_M\_do\_set** ()
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)<\_Nw> &\_\_x)
- const \_WordT \* **\_M\_getdata** () const
- \_WordT **\_M\_getword** (size\_t \_\_pos) const
- \_WordT & **\_M\_getword** (size\_t \_\_pos)
- \_WordT **\_M\_hiword** () const
- \_WordT & **\_M\_hiword** ()
- bool **\_M\_is\_any** () const
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)<\_Nw> &\_\_x) const

## Static Public Member Functions

- static \_WordT **\_S\_maskbit** (size\_t \_\_pos)
- static size\_t **\_S\_whichbit** (size\_t \_\_pos)
- static size\_t **\_S\_whichbyte** (size\_t \_\_pos)
- static size\_t **\_S\_whichword** (size\_t \_\_pos)

## Public Attributes

- \_WordT **\_M\_w** [\_Nw]

### 5.276.1 Detailed Description

```
template<size_t _Nw> struct std::_Base_bitset< _Nw >
```

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 69 of file `bitset`.

### 5.276.2 Member Data Documentation

**5.276.2.1** `template<size_t _Nw> _WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

Definition at line 74 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.277 `std::_Base_bitset< 0 >` Struct Template Reference

### Public Types

- typedef unsigned long `_WordT`

### Public Member Functions

- `_Base_bitset` (unsigned long long)
- `size_t _M_are_all_aux` () const
- `void _M_do_and` (const `_Base_bitset< 0 >` &)
- `size_t _M_do_count` () const
- `size_t _M_do_find_first` (size\_t) const
- `size_t _M_do_find_next` (size\_t, size\_t) const
- `void _M_do_flip` ()
- `void _M_do_left_shift` (size\_t)
- `void _M_do_or` (const `_Base_bitset< 0 >` &)
- `void _M_do_reset` ()
- `void _M_do_right_shift` (size\_t)

- `void _M_do_set ()`
- `unsigned long long _M_do_to_ullong () const`
- `unsigned long _M_do_to_ulong () const`
- `void _M_do_xor (const _Base_bitset< 0 > &)`
- `_WordT & _M_getword (size_t) const`
- `_WordT _M_hiword () const`
- `bool _M_is_any () const`
- `bool _M_is_equal (const _Base_bitset< 0 > &) const`

### Static Public Member Functions

- `static _WordT _S_maskbit (size_t __pos)`
- `static size_t _S_whichbit (size_t __pos)`
- `static size_t _S_whichbyte (size_t __pos)`
- `static size_t _S_whichword (size_t __pos)`

#### 5.277.1 Detailed Description

`template<> struct std::_Base_bitset< 0 >`

Base class, specialization for no storage (zero-length bitset).

See documentation for `bitset`.

Definition at line 511 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.278 `std::_Base_bitset< 1 >` Struct Template Reference

### Public Types

- `typedef unsigned long _WordT`

### Public Member Functions

- `_Base_bitset (unsigned long long __val)`
- `size_t _M_are_all_aux () const`
- `void _M_do_and (const _Base_bitset< 1 > &__x)`

- `size_t _M_do_count () const`
- `size_t _M_do_find_first (size_t __not_found) const`
- `size_t _M_do_find_next (size_t __prev, size_t __not_found) const`
- `void _M_do_flip ()`
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const \_Base\_bitset< 1 > &__x)`
- `void _M_do_reset ()`
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()`
- `unsigned long long _M_do_to_ullong () const`
- `unsigned long _M_do_to_ulong () const`
- `void _M_do_xor (const \_Base\_bitset< 1 > &__x)`
- `const _WordT * _M_getdata () const`
- `_WordT _M_getword (size_t)`
- `_WordT & _M_getword (size_t)`
- `_WordT _M_hiword () const`
- `_WordT & _M_hiword ()`
- `bool _M_is_any () const`
- `bool _M_is_equal (const \_Base\_bitset< 1 > &__x) const`

### Static Public Member Functions

- `static _WordT _S_maskbit (size_t __pos)`
- `static size_t _S_whichbit (size_t __pos)`
- `static size_t _S_whichbyte (size_t __pos)`
- `static size_t _S_whichword (size_t __pos)`

### Public Attributes

- `_WordT _M_w`

#### 5.278.1 Detailed Description

`template<> struct std::_Base_bitset< 1 >`

Base class, specialization for a single word.

See documentation for `bitset`.

Definition at line 367 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.279 `std::_Build_index_tuple< _Num >` Struct Template Reference

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

### Public Types

- typedef `_Build_index_tuple< _Num-1 >::__type::__next __type`

### 5.279.1 Detailed Description

```
template<std::size_t _Num> struct std::_Build_index_tuple< _Num >
```

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

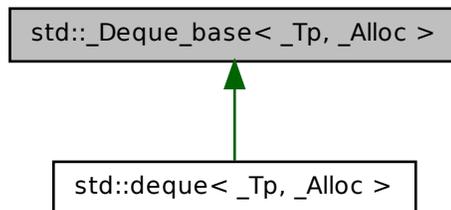
Definition at line 872 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

## 5.280 `std::_Deque_base< _Tp, _Alloc >` Class Template Reference

Inheritance diagram for `std::_Deque_base< _Tp, _Alloc >`:



## Public Types

- typedef `_Alloc` `allocator_type`
- typedef `_Deque_iterator`< `_Tp`, `const _Tp &`, `const _Tp * >` `const_iterator`
- typedef `_Deque_iterator`< `_Tp`, `_Tp &`, `_Tp * >` `iterator`

## Public Member Functions

- `_Deque_base` (`const allocator_type &__a`)
- `_Deque_base` (`_Deque_base &&__x`)
- `_Deque_base` (`const allocator_type &__a`, `size_t __num_elements`)
- `allocator_type` `get_allocator` () const

## Protected Types

- enum { `_S_initial_map_size` }
- typedef `_Alloc::template rebind`< `_Tp * >::other` `_Map_alloc_type`
- typedef `_Alloc::template rebind`< `_Tp >::other` `_Tp_alloc_type`

## Protected Member Functions

- `_Tp **` `_M_allocate_map` (`size_t __n`)
- `_Tp *` `_M_allocate_node` ()
- void `_M_create_nodes` (`_Tp ** __nstart`, `_Tp ** __nfinish`)
- void `_M_deallocate_map` (`_Tp ** __p`, `size_t __n`)
- void `_M_deallocate_node` (`_Tp * __p`)
- void `_M_destroy_nodes` (`_Tp ** __nstart`, `_Tp ** __nfinish`)
- `_Map_alloc_type` `_M_get_map_allocator` () const
- `const _Tp_alloc_type &` `_M_get_Tp_allocator` () const
- `_Tp_alloc_type &` `_M_get_Tp_allocator` ()
- void `_M_initialize_map` (`size_t`)

## Protected Attributes

- `_Deque_impl` `_M_impl`

## 5.281 std::Deque\_iterator< \_Tp, \_Ref, \_Ptr > Struct Template Reference 1371

### 5.280.1 Detailed Description

**template<typename \_Tp, typename \_Alloc> class std::Deque\_base< \_Tp, \_Alloc >**

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 436 of file stl\_deque.h.

### 5.280.2 Member Function Documentation

**5.280.2.1** **template<typename \_Tp , typename \_Alloc > void**  
**std::Deque\_base< \_Tp, \_Alloc >::M\_initialize\_map ( size\_t**  
***num\_elements* ) [protected]**

Layout storage.

#### Parameters

*num\_elements* The count of T's for which to allocate space at first.

#### Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 568 of file stl\_deque.h.

References std::max().

Referenced by std::deque< \_Tp, \_Alloc >::M\_range\_initialize().

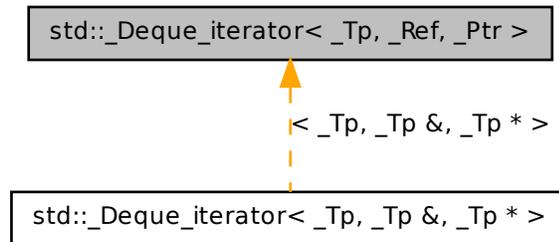
The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

## 5.281 std::Deque\_iterator< \_Tp, \_Ref, \_Ptr > Struct Template Reference

A deque::iterator.

Inheritance diagram for `std::_Deque_iterator<_Tp, _Ref, _Ptr >`:



## Public Types

- typedef `_Tp **` **Map\_pointer**
- typedef `_Deque_iterator` **Self**
- typedef `_Deque_iterator<_Tp, const _Tp &, const _Tp * >` **const\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Deque_iterator<_Tp, _Tp &, _Tp * >` **iterator**
- typedef `std::random_access_iterator_tag` **iterator\_category**
- typedef `_Ptr` **pointer**
- typedef `_Ref` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`)
- `_Deque_iterator` (`const iterator &__x`)
- void `_M_set_node` (`_Map_pointer __new_node`)
- reference `operator*` () const
- `_Self operator+` (`difference_type __n`) const
- `_Self operator++` (int)
- `_Self & operator++` ()
- `_Self & operator+=` (`difference_type __n`)
- `_Self operator-` (`difference_type __n`) const

## 5.281 `std::Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference 1373

- `_Self & operator-- ()`
- `_Self operator-- (int)`
- `_Self & operator-= (difference_type __n)`
- pointer `operator-> () const`
- reference `operator[] (difference_type __n) const`

### Static Public Member Functions

- static `size_t _S_buffer_size ()`

### Public Attributes

- `_Tp * _M_cur`
- `_Tp * _M_first`
- `_Tp * _M_last`
- `_Map_pointer _M_node`

### 5.281.1 Detailed Description

`template<typename _Tp, typename _Ref, typename _Ptr> struct std::Deque_iterator<_Tp, _Ref, _Ptr>`

A `deque::iterator`. Quite a bit of intelligence here. Much of the functionality of `deque` is actually passed off to this class. A `deque` holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 103 of file `stl_deque.h`.

### 5.281.2 Member Function Documentation

**5.281.2.1** `template<typename _Tp, typename _Ref, typename _Ptr> void std::Deque_iterator<_Tp, _Ref, _Ptr>::_M_set_node ( _Map_pointer __new_node ) [inline]`

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 231 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)

## 5.282 `std::_Derives_from_binary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [binary\\_function](#).

Inherits `__sfinae_types`.

### Static Public Attributes

- static const bool **value**

#### 5.282.1 Detailed Description

```
template<typename _Tp> struct std::_Derives_from_binary_function< _Tp >
```

Determines if the type `_Tp` derives from [binary\\_function](#).

Definition at line 201 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.283 `std::_Derives_from_unary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [unary\\_function](#).

Inherits `__sfinae_types`.

### Static Public Attributes

- static const bool **value**

#### 5.283.1 Detailed Description

```
template<typename _Tp> struct std::_Derives_from_unary_function< _Tp >
```

Determines if the type `_Tp` derives from [unary\\_function](#).

Definition at line 185 of file `functional`.

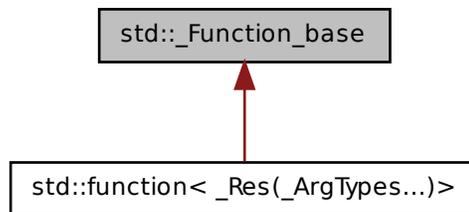
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.284 std::\_Function\_base Class Reference

Base class of all polymorphic function object wrappers.

Inheritance diagram for std::\_Function\_base:



### Public Types

- `typedef bool(* _Manager_type )(_Any_data &, const _Any_data &, _Manager_operation)`

### Public Member Functions

- `bool _M_empty () const`

### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

### 5.284.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 1510 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.285 `std::_Function_to_function_pointer< _Tp, _IsFunctionType >` Struct Template Reference

Turns a function type into a function pointer type.

### Public Types

- `typedef _Tp type`

### 5.285.1 Detailed Description

```
template<typename _Tp, bool _IsFunctionType = is_function<_Tp>::value>
struct std::_Function_to_function_pointer< _Tp, _IsFunctionType >
```

Turns a function type into a function pointer type.

Definition at line 217 of file functional.

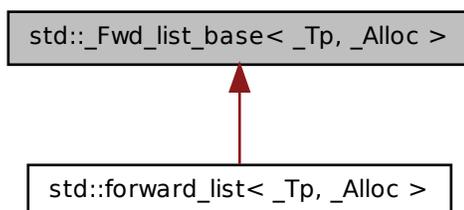
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.286 `std::_Fwd_list_base< _Tp, _Alloc >` Struct Template Reference

Base class for `forward_list`.

Inheritance diagram for std::\_Fwd\_list\_base< \_Tp, \_Alloc >:



## Public Types

- typedef `_Fwd_list_node< _Tp >` **\_Node**
- typedef `_Fwd_list_const_iterator< _Tp >` **const\_iterator**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**

## Public Member Functions

- `_Fwd_list_base` (const \_Alloc &\_\_a)
- `_Fwd_list_base` (\_Fwd\_list\_base &&\_\_lst)
- `_Fwd_list_base` (const \_Fwd\_list\_base &\_\_lst, const \_Alloc &\_\_a)
- `_Fwd_list_base` (\_Fwd\_list\_base &&\_\_lst, const \_Alloc &\_\_a)
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const
- `_Node_alloc_type` & `_M_get_Node_allocator` ()

## Protected Types

- typedef `_Alloc::template rebind< _Fwd_list_node< _Tp > >::other` **\_Node\_alloc\_type**
- typedef `_Alloc::template rebind< _Tp >::other` **\_Tp\_alloc\_type**

## Protected Member Functions

- template<typename... \_Args>  
`_Node` \* `_M_create_node` (\_Args &&...\_\_args)

- `void _M_erase_after (_Fwd_list_node_base * __pos)`
- `void _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `template<typename... _Args> _Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... _args)`
- `void _M_put_node (_Node * __p)`

### Protected Attributes

- `_Fwd_list_impl _M_impl`

### 5.286.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Fwd_list_base< _Tp, _Alloc >`

Base class for `forward_list`.

Definition at line 272 of file `forward_list.h`.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

### 5.287 `std::_Fwd_list_const_iterator< _Tp >` Struct Template Reference

A `forward_list::const_iterator`.

### Public Types

- `typedef const _Fwd_list_node< _Tp > _Node`
- `typedef _Fwd_list_const_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _Fwd_list_iterator< _Tp > iterator`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

## Public Member Functions

- `_Fwd_list_const_iterator` (const `_Fwd_list_node_base * __n`)
- `_Fwd_list_const_iterator` (const `iterator & __iter`)
- `_Self _M_next` () const
- bool `operator!=` (const `_Self & __x`) const
- reference `operator*` () const
- `_Self operator++` (int)
- `_Self & operator++` ()
- pointer `operator->` () const
- bool `operator==` (const `_Self & __x`) const

## Public Attributes

- const `_Fwd_list_node_base * _M_node`

### 5.287.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_const_iterator<_Tp>`

A `forward_list::const_iterator`. All the functions are op overloads.

Definition at line 185 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.288 `std::_Fwd_list_iterator<_Tp>` Struct Template Reference

A `forward_list::iterator`.

## Public Types

- typedef `_Fwd_list_node<_Tp> _Node`
- typedef `_Fwd_list_iterator<_Tp> _Self`
- typedef `ptrdiff_t difference_type`
- typedef `std::forward_iterator_tag iterator_category`
- typedef `_Tp * pointer`
- typedef `_Tp & reference`
- typedef `_Tp value_type`

## Public Member Functions

- [\\_Fwd\\_list\\_iterator](#) ([\\_Fwd\\_list\\_node\\_base](#) \*\_\_n)
- [\\_Self](#) [\\_M\\_next](#) () const
- bool [operator!=](#) (const [\\_Self](#) &\_\_x) const
- reference [operator\\*](#) () const
- [\\_Self](#) & [operator++](#) ()
- [\\_Self](#) [operator++](#) (int)
- pointer [operator->](#) () const
- bool [operator==](#) (const [\\_Self](#) &\_\_x) const

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_node](#)

### 5.288.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_iterator< _Tp >`

A `forward_list::iterator`. All the functions are op overloads.

Definition at line 118 of file `forward_list.h`.

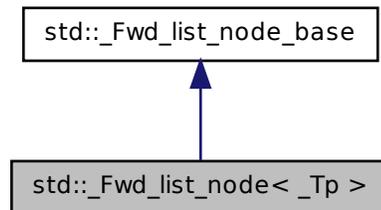
The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 5.289 `std::_Fwd_list_node< _Tp >` Struct Template Reference

A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



## Public Member Functions

- `template<typename... _Args> _Fwd_list_node (_Args &&... __args)`
- `void _M_reverse_after ()`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end)`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin)`

## Static Public Member Functions

- `static void swap (_Fwd_list_node_base &__x, _Fwd_list_node_base &__y)`

## Public Attributes

- `_Fwd_list_node_base * _M_next`
- `_Tp _M_value`

### 5.289.1 Detailed Description

```
template<typename _Tp> struct std::_Fwd_list_node< _Tp >
```

A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.

Definition at line 101 of file forward\_list.h.

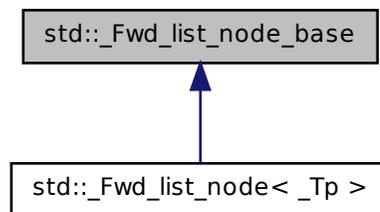
The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.290 std::\_Fwd\_list\_node\_base Struct Reference

A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Inheritance diagram for std::\_Fwd\_list\_node\_base:



### Public Member Functions

- void `_M_reverse_after` ()
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`, `_Fwd_list_node_base * __end`)
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`)

### Static Public Member Functions

- static void `swap` (`_Fwd_list_node_base & __x`, `_Fwd_list_node_base & __y`)

### Public Attributes

- `_Fwd_list_node_base * _M_next`

### 5.290.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Definition at line 44 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.291 `std::_Has_result_type_helper<_Tp>` Class Template Reference

Inherits `__sfnac_types`.

### Static Public Attributes

- static const bool `value`

### 5.291.1 Detailed Description

`template<typename _Tp> class std::_Has_result_type_helper<_Tp>`

Actual implementation of `_Has_result_type`, which uses SFINAE to determine if the type `_Tp` has a publicly-accessible member type `result_type`.

Definition at line 72 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.292 `std::_Index_tuple<_Indexes>` Struct Template Reference

### Public Types

- typedef `_Index_tuple<_Indexes..., sizeof...(_Indexes)>` `__next`

### 5.292.1 Detailed Description

```
template<int... _Indexes> struct std::_Index_tuple< _Indexes >
```

Stores a tuple of indices. Used by [bind\(\)](#) to extract the elements in a tuple.

Definition at line 865 of file functional.

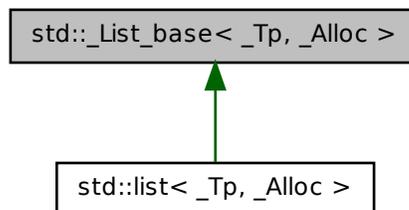
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.293 `std::_List_base< _Tp, _Alloc >` Class Template Reference

See [bits/stl\\_deque.h](#)'s [\\_Deque\\_base](#) for an explanation.

Inheritance diagram for `std::_List_base< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` `allocator_type`

### Public Member Functions

- `_List_base` (`const allocator_type &__a`)
- `_List_base` (`_List_base &&__x`)
- `void _M_clear ()`

- `const _Node_alloc_type & _M_get_Node_allocator () const`
- `_Node_alloc_type & _M_get_Node_allocator ()`
- `_Tp_alloc_type _M_get_Tp_allocator () const`
- `void _M_init ()`
- `allocator_type get_allocator () const`

### Protected Types

- `typedef _Alloc::template rebind< _List_node<_Tp> >::other _Node_alloc_type`
- `typedef _Alloc::template rebind<_Tp>::other _Tp_alloc_type`

### Protected Member Functions

- `_List_node<_Tp> * _M_get_node ()`
- `void _M_put_node (_List_node<_Tp> *__p)`

### Protected Attributes

- `_List_impl _M_impl`

#### 5.293.1 Detailed Description

`template<typename _Tp, typename _Alloc> class std::List_base<_Tp, _Alloc>`

See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.

Definition at line 277 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.294 `std::List_const_iterator<_Tp>` Struct Template Reference

A `list::const_iterator`.

## Public Types

- typedef const [\\_List\\_node](#)<\_Tp > **\_Node**
- typedef [\\_List\\_const\\_iterator](#)<\_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_List\\_iterator](#)<\_Tp > **iterator**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- [\\_List\\_const\\_iterator](#) (const [\\_List\\_node\\_base](#) \* \_\_x)
- [\\_List\\_const\\_iterator](#) (const [iterator](#) & \_\_x)
- bool **operator!=** (const [\\_Self](#) & \_\_x) const
- reference **operator\*** () const
- [\\_Self](#) **operator++** (int)
- [\\_Self](#) & **operator++** ()
- [\\_Self](#) & **operator--** ()
- [\\_Self](#) **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const [\\_Self](#) & \_\_x) const

## Public Attributes

- const [\\_List\\_node\\_base](#) \* **\_M\_node**

### 5.294.1 Detailed Description

**template<typename \_Tp> struct [std::\\_List\\_const\\_iterator](#)<\_Tp >**

A [list::const\\_iterator](#). All the functions are op overloads.

Definition at line 188 of file [stl\\_list.h](#).

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.295 `std::_List_iterator<_Tp>` Struct Template Reference

A list::iterator.

### Public Types

- typedef `_List_node<_Tp>` `_Node`
- typedef `_List_iterator<_Tp>` `_Self`
- typedef `ptrdiff_t` `difference_type`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef `_Tp *` `pointer`
- typedef `_Tp &` `reference`
- typedef `_Tp` `value_type`

### Public Member Functions

- `_List_iterator` (`_List_node_base *__x`)
- bool `operator!=` (const `_Self` & `__x`) const
- reference `operator*` () const
- `_Self` & `operator++` ()
- `_Self` `operator++` (int)
- `_Self` & `operator--` ()
- `_Self` `operator--` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self` & `__x`) const

### Public Attributes

- `_List_node_base * _M_node`

#### 5.295.1 Detailed Description

```
template<typename _Tp> struct std::_List_iterator<_Tp>
```

A list::iterator. All the functions are op overloads.

Definition at line 113 of file `stl_list.h`.

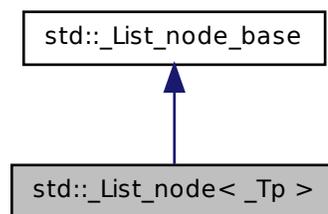
The documentation for this struct was generated from the following file:

- `stl_list.h`

## 5.296 `std::_List_node< _Tp >` Struct Template Reference

An actual node in the list.

Inheritance diagram for `std::_List_node< _Tp >`:



### Public Member Functions

- `template<typename... _Args> _List_node (_Args &&... args)`
- `void _M_hook (_List_node_base *const __position) throw ()`
- `void _M_reverse () throw ()`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) throw ()`
- `void _M_unhook () throw ()`

### Static Public Member Functions

- `static void swap (_List_node_base &__x, _List_node_base &__y) throw ()`

### Public Attributes

- `_Tp _M_data`
- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

### 5.296.1 Detailed Description

```
template<typename _Tp> struct std::_List_node< _Tp >
```

An actual node in the list.

Definition at line 95 of file `stl_list.h`.

### 5.296.2 Member Data Documentation

**5.296.2.1** `template<typename _Tp> _Tp std::_List_node< _Tp >::_M_data`

< User's data.

Definition at line 98 of file `stl_list.h`.

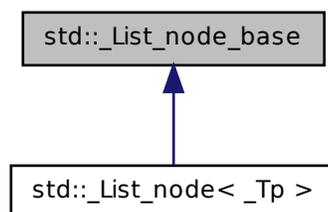
The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.297 `std::_List_node_base` Struct Reference

Common part of a node in the list.

Inheritance diagram for `std::_List_node_base`:



### Public Member Functions

- `void _M_hook (_List_node_base *const __position) throw ()`

- void `_M_reverse ()` throw ()
- void `_M_transfer (_List_node_base *const __first, _List_node_base *const __last)` throw ()
- void `_M_unhook ()` throw ()

### Static Public Member Functions

- static void `swap (_List_node_base &__x, _List_node_base &__y)` throw ()

### Public Attributes

- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

#### 5.297.1 Detailed Description

Common part of a node in the list.

Definition at line 71 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.298 `std::_Maybe_get_result_type< _Has_result_type, _Functor >` Struct Template Reference

If we have found a `result_type`, extract it.

Inheritance diagram for `std::_Maybe_get_result_type< _Has_result_type, _Functor >`:



#### 5.298.1 Detailed Description

`template<bool _Has_result_type, typename _Functor> struct std::_Maybe_get_result_type< _Has_result_type, _Functor >`

If we have found a `result_type`, extract it.

## 5.299 `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes > Struct Template Reference` 1391

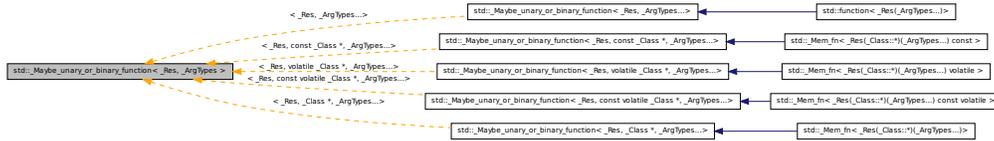
Definition at line 96 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.299 `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes > Struct Template Reference`

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes >`:



### 5.299.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Maybe_unary_or_binary_function<_Res, _ArgTypes >`

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

Definition at line 480 of file `functional`.

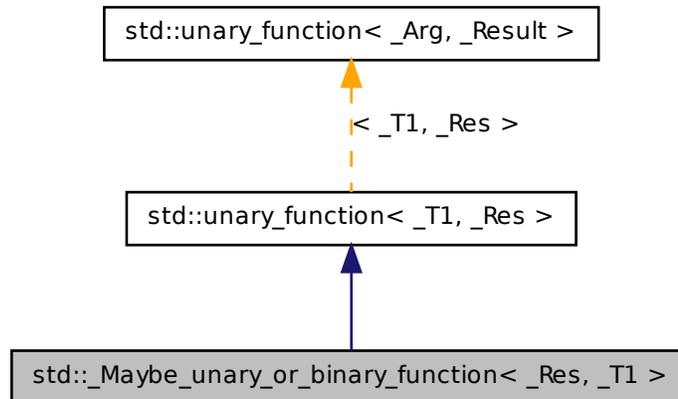
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.300 `std::_Maybe_unary_or_binary_function<_Res, _T1 > Struct Template Reference`

Derives from `unary_function`, as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _T1 >`:



## Public Types

- typedef `_T1` `argument_type`
- typedef `_Res` `result_type`

### 5.300.1 Detailed Description

```
template<typename _Res, typename _T1> struct std::_Maybe_unary_or_
binary_function<_Res, _T1 >
```

Derives from `unary_function`, as appropriate.

Definition at line 484 of file `functional`.

### 5.300.2 Member Typedef Documentation

**5.300.2.1** typedef `_T1` `std::unary_function<_T1, _Res >::argument_type`  
[`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

**5.301 std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 > Struct Template Reference** 1393

---

Definition at line 102 of file stl\_function.h.

**5.300.2.2 typedef \_Res std::unary\_function< \_T1 , \_Res >::result\_type [inherited]**

result\_type is the return type

Definition at line 105 of file stl\_function.h.

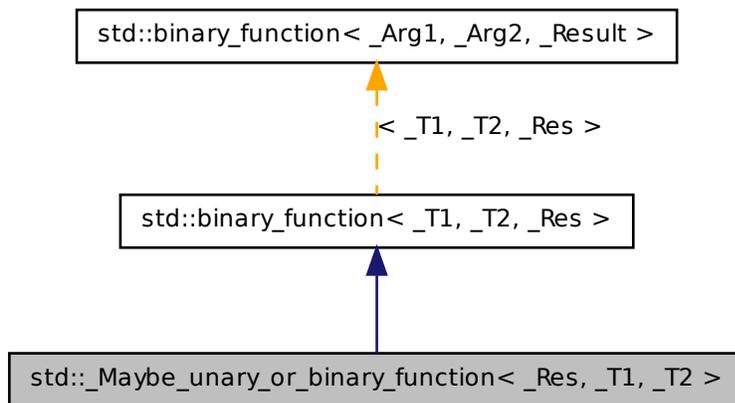
The documentation for this struct was generated from the following file:

- [functional](#)

**5.301 std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 > Struct Template Reference**

Derives from [binary\\_function](#), as appropriate.

Inheritance diagram for std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >:



**Public Types**

- typedef \_T1 [first\\_argument\\_type](#)

- typedef `_Res` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### 5.301.1 Detailed Description

`template<typename _Res, typename _T1, typename _T2> struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`

Derives from [binary\\_function](#), as appropriate.

Definition at line 489 of file `functional`.

### 5.301.2 Member Typedef Documentation

**5.301.2.1** `typedef _T1 std::binary_function< _T1 , _T2 , _Res >::first_argument_type` [`inherited`]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.301.2.2** `typedef _Res std::binary_function< _T1 , _T2 , _Res >::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.301.2.3** `typedef _T2 std::binary_function< _T1 , _T2 , _Res >::second_argument_type` [`inherited`]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.302 `std::_Maybe_wrap_member_pointer<_Tp>` > Struct Template Reference

## **5.302 `std::_Maybe_wrap_member_pointer<_Tp>` > Struct Template Reference**

### **Public Types**

- `typedef _Tp type`

### **Static Public Member Functions**

- `static const _Tp & __do_wrap (const _Tp &__x)`

#### **5.302.1 Detailed Description**

`template<typename _Tp> struct std::_Maybe_wrap_member_pointer<_Tp>`

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. The primary template handles the non--member-pointer case.

Definition at line 1060 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## **5.303 `std::_Maybe_wrap_member_pointer<_Tp _Class::*>` > Struct Template Reference**

### **Public Types**

- `typedef _Mem_fn<_Tp _Class::*> type`

### **Static Public Member Functions**

- `static type __do_wrap (_Tp _Class::* __pm)`

### 5.303.1 Detailed Description

```
template<typename _Tp, typename _Class> struct std::_Maybe_wrap_
member_pointer< _Tp _Class::* >
```

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. This partial specialization handles the member pointer case.

Definition at line 1075 of file `functional`.

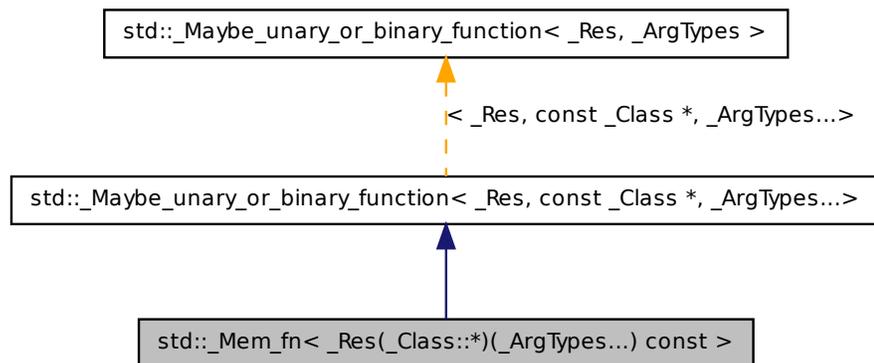
The documentation for this struct was generated from the following file:

- [functional](#)

### 5.304 `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >` Class Template Reference

Implementation of `mem_fn` for const member function pointers.

Inheritance diagram for `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >`:



#### Public Types

- `typedef _Res result_type`

## Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class *__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class &__object, _ArgTypes... __args) const`

### 5.304.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class`  
`std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const >`

Implementation of `mem_fn` for const member function pointers.

Definition at line 540 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.305 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile > Class` Template Reference

Implementation of `mem_fn` for const volatile member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`:



## Public Types

- `typedef _Res result_type`

## Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`

- `_Res operator()` (const volatile `_Class *``__object`, `_ArgTypes...``__args`) const
- `_Res operator()` (const volatile `_Class &``__object`, `_ArgTypes...``__args`) const

### 5.305.1 Detailed Description

template<typename `_Res`, typename `_Class`, typename... `_ArgTypes`> class `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`

Implementation of `mem_fn` for const volatile member function pointers.

Definition at line 633 of file `functional`.

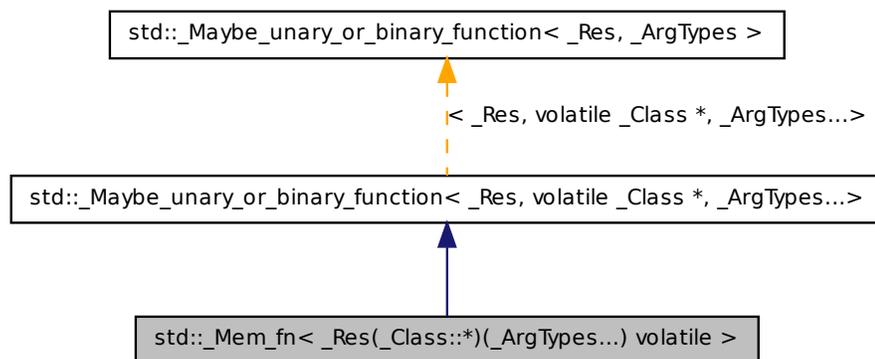
The documentation for this class was generated from the following file:

- [functional](#)

### 5.306 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference

Implementation of `mem_fn` for volatile member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`:



## 5.307 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

### Public Types

- typedef `_Res` `result_type`

### Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (volatile _Class * __object, _ArgTypes... __args) const`
- `_Res operator() (volatile _Class & __object, _ArgTypes... __args) const`

#### 5.306.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >
```

Implementation of `mem_fn` for volatile member function pointers.

Definition at line 586 of file `functional`.

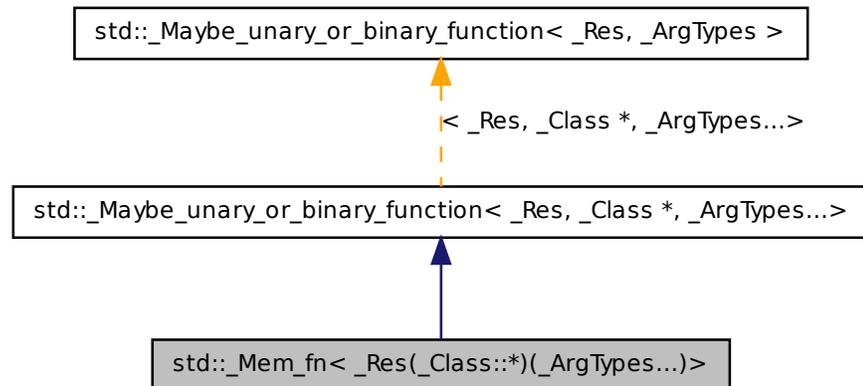
The documentation for this class was generated from the following file:

- [functional](#)

## 5.307 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

Implementation of `mem_fn` for member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- **\_Mem\_fn** (`_Functor __pmf`)
- template<typename `_Tp` >  
`_Res operator()` (`_Tp &__object, _ArgTypes...__args`) const
- `_Res operator()` (`_Class *__object, _ArgTypes...__args`) const
- `_Res operator()` (`_Class &__object, _ArgTypes...__args`) const

### 5.307.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>
```

Implementation of `mem_fn` for member function pointers.

Definition at line 494 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.308 `std::_Mu<_Arg, false, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVArg, typename _Tuple > _CVArg && operator() (_CVArg &&__arg, _Tuple &&) const volatile`

#### 5.308.1 Detailed Description

`template<typename _Arg> class std::_Mu<_Arg, false, false >`

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the `_Mu` object. [TR1 3.6.3/5 bullet 4]

Definition at line 1036 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.309 `std::_Mu<_Arg, false, true >` Class Template Reference

### Public Member Functions

- `template<typename _Tuple > result<_Mu(_Arg, _Tuple)>::type operator() (const volatile _Arg &, _Tuple &&__tuple) const volatile`

#### 5.309.1 Detailed Description

`template<typename _Arg> class std::_Mu<_Arg, false, true >`

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 1002 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.310 `std::_Mu< _Arg, true, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVArg, typename... _Args> result_of< _CVArg(_Args...)>::type operator() (_CVArg &__arg, tuple< _Args...> &&__tuple) const volatile`

### 5.310.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, true, false >`

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). [TR1 3.6.3/5 bullet 2]

Definition at line 961 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.311 `std::_Mu< reference_wrapper< _Tp >, false, false >` Class Template Reference

### Public Types

- `typedef _Tp & result_type`

### Public Member Functions

- `template<typename _CVRef, typename _Tuple > result_type operator() (_CVRef &__arg, _Tuple &&) const volatile`

### 5.311.1 Detailed Description

```
template<typename _Tp> class std::_Mu< reference_wrapper< _Tp >, false,  
false >
```

If the argument is `reference_wrapper<_Tp>`, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 940 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.312 `std::_Placeholder<_Num >` Struct Template Reference

The type of placeholder objects defined by `libstdc++`.

### 5.312.1 Detailed Description

```
template<int _Num> struct std::_Placeholder<_Num >
```

The type of placeholder objects defined by `libstdc++`.

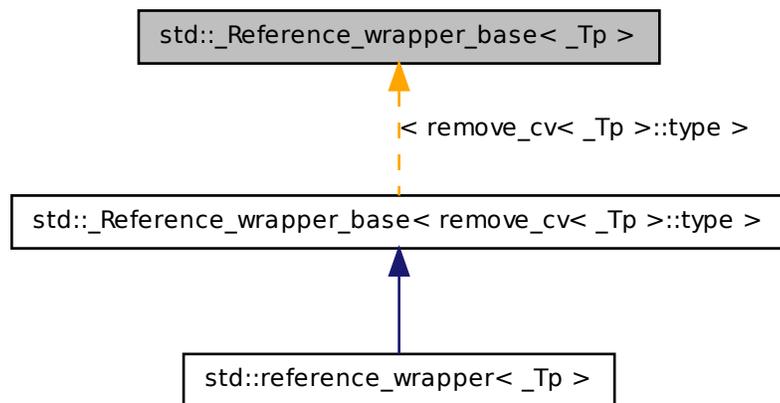
Definition at line 804 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.313 `std::_Reference_wrapper_base< _Tp >` Struct Template Reference

Inheritance diagram for `std::_Reference_wrapper_base< _Tp >`:



#### 5.313.1 Detailed Description

`template<typename _Tp> struct std::_Reference_wrapper_base< _Tp >`

Derives from [unary\\_function](#) or [binary\\_function](#) when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both [unary\\_function](#) and [binary\\_function](#).

Definition at line 307 of file `functional`.

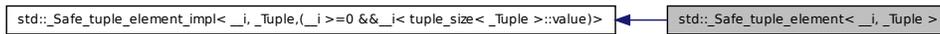
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.314 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference 1405

### 5.314 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element< __i, _Tuple >`:



#### 5.314.1 Detailed Description

`template<int __i, typename _Tuple> struct std::_Safe_tuple_element< __i, _Tuple >`

Like `tuple_element`, but returns `_No_tuple_element` when `tuple_element` would return an error.

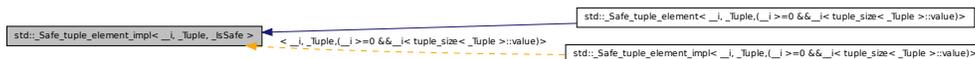
Definition at line 914 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.315 `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`:



### 5.315.1 Detailed Description

```
template<int __i, typename _Tuple, bool _IsSafe> struct std::_Safe_tuple_
element_impl< __i, _Tuple, _IsSafe >
```

Implementation helper for [\\_Safe\\_tuple\\_element](#). This primary template handles the case where it is safe to use `tuple_element`.

Definition at line 895 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.316 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference

#### Public Types

- typedef `_No_tuple_element` **type**

#### 5.316.1 Detailed Description

```
template<int __i, typename _Tuple> struct std::_Safe_tuple_element_impl< __i,
_Tuple, false >
```

Implementation helper for [\\_Safe\\_tuple\\_element](#). This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

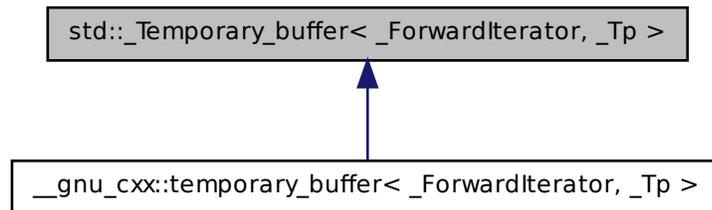
Definition at line 904 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.317 `std::_Temporary_buffer<_ForwardIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::_Temporary_buffer<_ForwardIterator, _Tp >`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `_Temporary_buffer` (`_ForwardIterator __first`, `_ForwardIterator __last`)
- iterator `begin` ()
- iterator `end` ()
- size\_type `requested_size` () const
- size\_type `size` () const

### Protected Attributes

- pointer `_M_buffer`
- size\_type `_M_len`
- size\_type `_M_original_len`

### 5.317.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp> class std::_Temporary_  
buffer< _ForwardIterator, _Tp >
```

This class is used in two places: [stl\\_algo.h](#) and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 122 of file `stl_tempbuf.h`.

### 5.317.2 Constructor & Destructor Documentation

```
5.317.2.1 template<typename _ForwardIterator , typename _Tp  
> std::_Temporary_buffer< _ForwardIterator, _Tp  
>::_Temporary_buffer ( _ForwardIterator __first,  
_ForwardIterator __last )
```

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 181 of file `stl_tempbuf.h`.

References `std::pair<_T1, _T2 >::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair<_T1, _T2 >::second`.

### 5.317.3 Member Function Documentation

```
5.317.3.1 template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( )  
[inline]
```

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

```
5.317.3.2 template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( )  
[inline]
```

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.

**5.317.3.3** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size(  
) const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

**5.317.3.4** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp>::size ( ) const  
[inline]`

As per Table mumble.

Definition at line 141 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 5.318 `std::_Tuple_impl<_Idx>` Struct Template Reference

### Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &)`

### 5.318.1 Detailed Description

`template<std::size_t _Idx> struct std::_Tuple_impl<_Idx>`

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

Definition at line 125 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.319 `std::_Tuple_impl<_Idx, _Head, _Tail...>` Struct Template Reference

Inherits `_Tuple_impl<_Idx+1, _Tail...>`, and `_Head_base<_Idx, _Head, std::is_empty<_Head>::value >`.

### Public Types

- `typedef _Head_base<_Idx, _Head, std::is_empty<_Head>::value > _Base`
- `typedef _Tuple_impl<_Idx+1, _Tail...> _Inherited`

### Public Member Functions

- `_Tuple_impl (const _Tuple_impl &__in)`
- `_Tuple_impl (_Tuple_impl &&__in)`
- `template<typename... _UElements> _Tuple_impl (const _Tuple_impl<_Idx, _UElements...> &__in)`
- `template<typename... _UElements> _Tuple_impl (_Tuple_impl<_Idx, _UElements...> &&__in)`
- `_Tuple_impl (const _Head &__head, const _Tail &...__tail)`
- `template<typename _UHead, typename... _UTail> _Tuple_impl (_UHead &&__head, _UTail &&...__tail)`
- `_Head & _M_head ()`
- `const _Head & _M_head () const`
- `const _Inherited & _M_tail () const`
- `_Inherited & _M_tail ()`
- `template<typename... _UElements> _Tuple_impl & operator= (_Tuple_impl<_Idx, _UElements...> &&__in)`
- `_Tuple_impl & operator= (const _Tuple_impl &__in)`
- `_Tuple_impl & operator= (_Tuple_impl &&__in)`
- `template<typename... _UElements> _Tuple_impl & operator= (const _Tuple_impl<_Idx, _UElements...> &__in)`

### Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &__in)`

### 5.319.1 Detailed Description

`template<std::size_t _Idx, typename _Head, typename... _Tail> struct std::_Tuple_impl<_Idx, _Head, _Tail...>`

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 137 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.320 `std::_Vector_base<_Tp, _Alloc>` Struct Template Reference

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

Inheritance diagram for `std::_Vector_base<_Tp, _Alloc>`:



### Public Types

- `typedef _Alloc::template rebind<_Tp>::other _Tp_alloc_type`
- `typedef _Alloc allocator_type`

### Public Member Functions

- `_Vector_base` (const allocator\_type &\_\_a)
- `_Vector_base` (size\_t \_\_n, const allocator\_type &\_\_a)
- `_Vector_base` (`_Vector_base` &&\_\_x)
- `_Tp_alloc_type::pointer _M_allocate` (size\_t \_\_n)
- `void _M_deallocate` (typename `_Tp_alloc_type::pointer` \_\_p, size\_t \_\_n)

- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `allocator_type get_allocator () const`

## Public Attributes

- `_Vector_impl _M_impl`

### 5.320.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Vector_base< _Tp, _Alloc >`

See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.

Definition at line 69 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

## 5.321 `std::_Weak_result_type< _Functor > Struct` Template Reference

Inheritance diagram for `std::_Weak_result_type< _Functor >`:



### 5.321.1 Detailed Description

`template<typename _Functor> struct std::_Weak_result_type< _Functor >`

Strip top-level cv-qualifiers from the function object and let `_Weak_result_type_impl` perform the real work.

Definition at line 168 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.322 `std::_Weak_result_type_impl<_Functor>` > Struct Template Reference 413

## 5.322 `std::_Weak_result_type_impl<_Functor>` > Struct Template Reference

Inheritance diagram for `std::_Weak_result_type_impl<_Functor>`:



### 5.322.1 Detailed Description

```
template<typename _Functor> struct std::_Weak_result_type_impl<_Functor>
```

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 110 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.323 `std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function reference.

### Public Types

- typedef `_Res` `result_type`

### 5.323.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>
```

Retrieve the result type for a function reference.

Definition at line 123 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.324 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function pointer.

#### Public Types

- `typedef _Res result_type`

#### 5.324.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>
```

Retrieve the result type for a function pointer.

Definition at line 130 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.325 `std::_Weak_result_type_impl< _Res(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function type.

#### Public Types

- `typedef _Res result_type`

#### 5.325.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(_ArgTypes...)>
```

Retrieve the result type for a function type.

**5.326** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`  
**Struct Template Reference** **1415**

---

Definition at line 116 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.326** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >` **Struct Template Reference**

Retrieve result type for a const member function pointer.

### Public Types

- `typedef _Res result_type`

#### 5.326.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> struct std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`

Retrieve result type for a const member function pointer.

Definition at line 144 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.327** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >` **Struct Template Reference**

Retrieve result type for a const volatile member function pointer.

### Public Types

- `typedef _Res result_type`

### 5.327.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >
```

Retrieve result type for a const volatile member function pointer.

Definition at line 158 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.328 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >` Struct Template Reference

Retrieve result type for a volatile member function pointer.

#### Public Types

- typedef `_Res result_type`

### 5.328.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >
```

Retrieve result type for a volatile member function pointer.

Definition at line 151 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.329 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) >` Struct Template Reference

Retrieve result type for a member function pointer.

## Public Types

- typedef `_Res` `result_type`

### 5.329.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...)>
```

Retrieve result type for a member function pointer.

Definition at line 137 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.330 `std::add_lvalue_reference< _Tp >` Struct Template Reference

[add\\_lvalue\\_reference](#)

Inherits `std::_add_lvalue_reference_helper< _Tp >`.

## Public Types

- typedef `_Tp` `type`

### 5.330.1 Detailed Description

```
template<typename _Tp> struct std::add_lvalue_reference< _Tp >
```

[add\\_lvalue\\_reference](#)

Definition at line 125 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.331 `std::add_rvalue_reference< _Tp >` Struct Template Reference

[add\\_rvalue\\_reference](#)

Inherits `std::_add_rvalue_reference_helper< _Tp >`.

### Public Types

- `typedef _Tp type`

#### 5.331.1 Detailed Description

`template<typename _Tp> struct std::add_rvalue_reference< _Tp >`

[add\\_rvalue\\_reference](#)

Definition at line 140 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.332 `std::adopt_lock_t` Struct Reference

Assume the calling thread has already obtained mutex ownership /// and manage it.

#### 5.332.1 Detailed Description

Assume the calling thread has already obtained mutex ownership /// and manage it.

Definition at line 383 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.333 `std::aligned_storage< _Len, _Align >` Struct Template Reference

Alignment type.

### 5.333.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __-
aligned_storage_msa<_Len>::__type)> struct std::aligned_storage< _Len, _-
Align >
```

Alignment type. The value of `_Align` is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than `_Len` (3.9). The member typedef `type` shall be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

Definition at line 342 of file `type_traits`.

The documentation for this struct was generated from the following file:

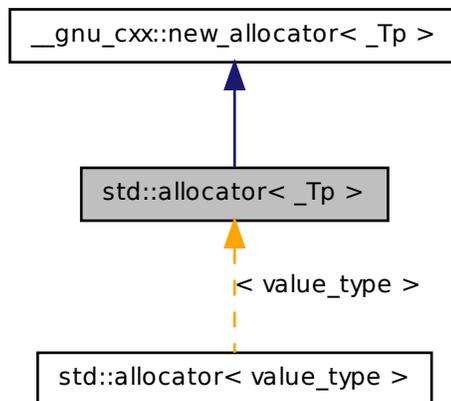
- [type\\_traits](#)

## 5.334 `std::allocator<_Tp>` Class Template Reference

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Inheritance diagram for `std::allocator<_Tp>`:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **allocator** (const [allocator](#) &\_\_a) throw ()
- template<typename \_Tp1 >  
**allocator** (const [allocator](#)<\_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- pointer **allocate** (size\_type \_\_n, const void \*=0)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- template<typename... \_Args >  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.334.1 Detailed Description

template<typename \_Tp> class **std::allocator**<\_Tp >

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Definition at line 86 of file allocator.h.

The documentation for this class was generated from the following file:

- [allocator.h](#)

### 5.335 **std::allocator**< void > Class Template Reference

[allocator](#)<void> specialization.

## Public Types

- typedef `const void *` **const\_pointer**
- typedef `ptrdiff_t` **difference\_type**
- typedef `void *` **pointer**
- typedef `size_t` **size\_type**
- typedef `void` **value\_type**

### 5.335.1 Detailed Description

`template<> class std::allocator< void >`

[allocator<void>](#) specialization.

Definition at line 64 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.336 `std::atomic< _Tp >` Struct Template Reference

`atomic` /// 29.4.3, Generic atomic type, primary class template.

### Public Member Functions

- `atomic` (`_Tp __i`)
- `atomic` (`const atomic &`)
- `bool compare_exchange_strong` (`_Tp &`, `_Tp`, `memory_order`, `memory_order`) `volatile`
- `bool compare_exchange_strong` (`_Tp &`, `_Tp`, `memory_order=memory_order_order_seq_cst`) `volatile`
- `bool compare_exchange_weak` (`_Tp &`, `_Tp`, `memory_order=memory_order_order_seq_cst`) `volatile`
- `bool compare_exchange_weak` (`_Tp &`, `_Tp`, `memory_order`, `memory_order`) `volatile`
- `_Tp exchange` (`_Tp __i`, `memory_order=memory_order_order_seq_cst`) `volatile`
- `bool is_lock_free` () `const volatile`
- `_Tp load` (`memory_order=memory_order_order_seq_cst`) `const volatile`
- `operator _Tp` () `const`
- `atomic & operator=` (`const atomic &`) `volatile`
- `_Tp operator=` (`_Tp __i`)
- `void store` (`_Tp`, `memory_order=memory_order_order_seq_cst`) `volatile`

### 5.336.1 Detailed Description

`template<typename _Tp> struct std::atomic< _Tp >`

`atomic` /// 29.4.3, Generic atomic type, primary class template.

Definition at line 86 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.337 `std::atomic< _Tp * >` Struct Template Reference

Partial specialization for pointer types.

Inherits `atomic_address`.

#### Public Member Functions

- `atomic` (`_Tp *__v`)
- `atomic` (const `atomic` &)
- bool `compare_exchange_strong` (`_Tp *&`, `_Tp *`, `memory_order`, `memory_order`)
- bool `compare_exchange_strong` (`_Tp *&`, `_Tp *`, `memory_order=memory_order_seq_cst`)
- bool `compare_exchange_weak` (`_Tp *&`, `_Tp *`, `memory_order`, `memory_order`)
- bool `compare_exchange_weak` (`_Tp *&`, `_Tp *`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `exchange` (`_Tp *__v`, `memory_order __m=memory_order_seq_cst`)
- `_Tp *` `fetch_add` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `fetch_sub` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp *` `load` (`memory_order __m=memory_order_seq_cst`) const
- `operator _Tp *` () const
- `_Tp *` `operator++` (int)
- `_Tp *` `operator++` ()
- `_Tp *` `operator+=` (`ptrdiff_t __d`)
- `_Tp *` `operator--` ()
- `_Tp *` `operator--` (int)
- `_Tp *` `operator-=` (`ptrdiff_t __d`)
- `_Tp *` `operator=` (`_Tp *__v`)
- `atomic` & `operator=` (const `atomic` &) volatile
- void `store` (`_Tp *__v`, `memory_order __m=memory_order_seq_cst`)

### 5.337.1 Detailed Description

```
template<typename _Tp> struct std::atomic< _Tp * >
```

Partial specialization for pointer types.

Definition at line 134 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.338 `std::atomic< bool >` Struct Template Reference

Explicit specialization for `bool`.

Inherits `__atomic_bool_base`.

### Public Types

- typedef `atomic_bool` `__base_type`
- typedef `bool` `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

### Public Attributes

- `bool _M_i`

### 5.338.1 Detailed Description

```
template<> struct std::atomic< bool >
```

Explicit specialization for `bool`.

Definition at line 225 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.339 `std::atomic< char >` Struct Template Reference

Explicit specialization for char.

Inherits `__atomic_char_base`.

### Public Types

- typedef `atomic_char_base_type`
- typedef `char __integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

### Public Attributes

- `char _M_i`

#### 5.339.1 Detailed Description

`template<> struct std::atomic< char >`

Explicit specialization for char.

Definition at line 243 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.340 `std::atomic< char16_t >` Struct Template Reference

Explicit specialization for `char16_t`.

Inherits `__atomic_short_base`.

## Public Types

- typedef `atomic_char16_t __base_type`
- typedef `char16_t __integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

## Public Attributes

- `short _M_i`

### 5.340.1 Detailed Description

`template<> struct std::atomic< char16_t >`

Explicit specialization for `char16_t`.

Definition at line 459 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.341 `std::atomic< char32_t >` Struct Template Reference

Explicit specialization for `char32_t`.

Inherits `__atomic_int_base`.

## Public Types

- typedef `atomic_char32_t __base_type`
- typedef `char32_t __integral_type`

## Public Member Functions

- [atomic](#) (`__integral_type __i`)
- [atomic](#) (`const atomic &`)
- [atomic](#) & [operator=](#) (`const atomic &`) volatile

## Public Attributes

- `int _M_i`

### 5.341.1 Detailed Description

`template<> struct std::atomic< char32_t >`

Explicit specialization for `char32_t`.

Definition at line 477 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.342 `std::atomic< int >` Struct Template Reference

Explicit specialization for `int`.

Inherits `__atomic_int_base`.

## Public Types

- typedef [atomic\\_int\\_base\\_type](#)
- typedef `int` [\\_\\_integral\\_type](#)

## Public Member Functions

- [atomic](#) (`__integral_type __i`)
- [atomic](#) (`const atomic &`)
- [atomic](#) & [operator=](#) (`const atomic &`) volatile

## Public Attributes

- `int _M_i`

### 5.342.1 Detailed Description

`template<> struct std::atomic< int >`

Explicit specialization for int.

Definition at line 333 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.343 `std::atomic< long >` Struct Template Reference

Explicit specialization for long.

Inherits `__atomic_long_base`.

### Public Types

- typedef `atomic_long __base_type`
- typedef long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- long `_M_i`

### 5.343.1 Detailed Description

`template<> struct std::atomic< long >`

Explicit specialization for long.

Definition at line 369 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.344 `std::atomic< long long >` Struct Template Reference

Explicit specialization for long long.

Inherits `__atomic_llong_base`.

### Public Types

- typedef `atomic_llong_base_type`
- typedef long long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- long long `_M_i`

### 5.344.1 Detailed Description

`template<> struct std::atomic< long long >`

Explicit specialization for long long.

Definition at line 405 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.345 `std::atomic< short >` Struct Template Reference

Explicit specialization for short.

Inherits `__atomic_short_base`.

## Public Types

- typedef `atomic_short` `__base_type`
- typedef `short` `__integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) `volatile`

## Public Attributes

- `short _M_i`

### 5.345.1 Detailed Description

`template<> struct std::atomic< short >`

Explicit specialization for `short`.

Definition at line 297 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.346 `std::atomic< signed char >` Struct Template Reference

Explicit specialization for `signed char`.

Inherits `__atomic_schar_base`.

## Public Types

- typedef `atomic_schar` `__base_type`
- typedef `signed char` `__integral_type`

## Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

## Public Attributes

- signed char `_M_i`

### 5.346.1 Detailed Description

**template<> struct std::atomic< signed char >**

Explicit specialization for signed char.

Definition at line 261 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.347 **std::atomic< unsigned char > Struct Template Reference**

Explicit specialization for unsigned char.

Inherits `__atomic_uchar_base`.

## Public Types

- typedef [atomic\\_uchar](#) `__base_type`
- typedef unsigned char `__integral_type`

## Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

## Public Attributes

- unsigned char `_M_i`

### 5.347.1 Detailed Description

`template<> struct std::atomic< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 279 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.348 `std::atomic< unsigned int >` Struct Template Reference

Explicit specialization for unsigned int.

Inherits `__atomic_uint_base`.

## Public Types

- typedef `atomic_uint __base_type`
- typedef unsigned int `__integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

## Public Attributes

- unsigned int `_M_i`

### 5.348.1 Detailed Description

`template<> struct std::atomic< unsigned int >`

Explicit specialization for unsigned int.

Definition at line 351 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.349 `std::atomic< unsigned long >` Struct Template Reference

Explicit specialization for unsigned long.

Inherits `__atomic_ulong_base`.

### Public Types

- typedef [atomic\\_ulong](#) `__base_type`
- typedef unsigned long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- `atomic` & `operator=` (const [atomic](#) &) volatile

### Public Attributes

- unsigned long `_M_i`

### 5.349.1 Detailed Description

`template<> struct std::atomic< unsigned long >`

Explicit specialization for unsigned long.

Definition at line 387 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.350 `std::atomic< unsigned long long >` Struct Template Reference

Explicit specialization for unsigned long long.

Inherits `__atomic_ullong_base`.

### Public Types

- typedef `atomic_ullong __base_type`
- typedef unsigned long long `__integral_type`

### Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

### Public Attributes

- unsigned long long `_M_i`

#### 5.350.1 Detailed Description

`template<> struct std::atomic< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 423 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.351 `std::atomic< unsigned short >` Struct Template Reference

Explicit specialization for unsigned short.

Inherits `__atomic_ushort_base`.

## Public Types

- typedef `atomic_ushort` `__base_type`
- typedef unsigned short `__integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

## Public Attributes

- unsigned short `_M_i`

### 5.351.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 315 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

### 5.352 `std::atomic< void * >` Struct Template Reference

Explicit specialization for `void*`.

Inherits `atomic_address`.

## Public Types

- typedef `atomic_address` `__base_type`
- typedef `void *` `__integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic` & `operator=` (`const atomic &`) volatile

### 5.352.1 Detailed Description

`template<> struct std::atomic< void * >`

Explicit specialization for `void*`.

Definition at line 207 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.353 `std::atomic< wchar_t >` Struct Template Reference

Explicit specialization for `wchar_t`.

Inherits `__atomic_wchar_t_base`.

## Public Types

- typedef `atomic_wchar_t__base_type`
- typedef `wchar_t__integral_type`

## Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic` & `operator=` (`const atomic &`) volatile

## Public Attributes

- `wchar_t_M_i`

### 5.353.1 Detailed Description

`template<> struct std::atomic< wchar_t >`

Explicit specialization for `wchar_t`.

Definition at line 441 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.354 `std::auto_ptr< _Tp >` Class Template Reference

A simple smart pointer providing strict ownership semantics.

### Public Types

- `typedef _Tp element_type`

### Public Member Functions

- `auto_ptr (element_type *__p=0) throw ()`
- `auto_ptr (auto_ptr &__a) throw ()`
- `template<typename _Tp1 > auto_ptr (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr (auto_ptr_ref< element_type > __ref) throw ()`
- `~auto_ptr ()`
- `element_type * get () const throw ()`
- `template<typename _Tp1 > operator auto_ptr< _Tp1 > () throw ()`
- `template<typename _Tp1 > operator auto_ptr_ref< _Tp1 > () throw ()`
- `element_type & operator* () const throw ()`
- `element_type * operator-> () const throw ()`
- `auto_ptr & operator= (auto_ptr &__a) throw ()`
- `template<typename _Tp1 > auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr & operator= (auto_ptr_ref< element_type > __ref) throw ()`
- `element_type * release () throw ()`
- `void reset (element_type *__p=0) throw ()`

### 5.354.1 Detailed Description

`template<typename _Tp> class std::auto_ptr<_Tp>`

A simple smart pointer providing strict ownership semantics. The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyCon requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the `libstdc++` test suite.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 85 of file `auto_ptr.h`.

### 5.354.2 Member Typedef Documentation

**5.354.2.1** `template<typename _Tp> typedef _Tp std::auto_ptr<_Tp>::element_type`

The pointed-to type.

Definition at line 92 of file `auto_ptr.h`.

### 5.354.3 Constructor & Destructor Documentation

**5.354.3.1** `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( element_type * __p = 0 ) throw () [inline, explicit]`

An `auto_ptr` is usually constructed from a raw pointer.

#### Parameters

*p* A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*.

Definition at line 101 of file auto\_ptr.h.

**5.354.3.2** `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( auto_ptr<_Tp> & __a ) throw () [inline]`

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

*a* Another auto\_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 110 of file auto\_ptr.h.

**5.354.3.3** `template<typename _Tp> template<typename _Tp1 > std::auto_ptr<_Tp>::auto_ptr ( auto_ptr<_Tp1> & __a ) throw () [inline]`

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

*a* Another auto\_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 123 of file auto\_ptr.h.

**5.354.3.4** `template<typename _Tp> std::auto_ptr<_Tp>::~~auto_ptr ( ) [inline]`

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 168 of file auto\_ptr.h.

### 5.354.3.5 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr ( auto_ptr_ref< element_type > __ref ) throw () [inline]`

Automatic conversions.

These operations convert an `auto_ptr` into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(.....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

Definition at line 258 of file `auto_ptr.h`.

## 5.354.4 Member Function Documentation

### 5.354.4.1 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::get ( void ) const throw () [inline]`

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This `auto_ptr` still owns the memory.

Definition at line 209 of file `auto_ptr.h`.

### 5.354.4.2 `template<typename _Tp> element_type& std::auto_ptr<_Tp>::operator*( ) const throw () [inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 179 of file `auto_ptr.h`.

**5.354.4.3** `template<typename _Tp> element_type* std::auto_ptr< _Tp >::operator-> ( ) const throw () [inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 192 of file auto\_ptr.h.

**5.354.4.4** `template<typename _Tp> template<typename _Tp1 > auto_ptr& std::auto_ptr< _Tp >::operator= ( auto_ptr< _Tp1 > & __a ) throw () [inline]`

auto\_ptr assignment operator.

#### Parameters

*a* Another auto\_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 152 of file auto\_ptr.h.

**5.354.4.5** `template<typename _Tp> auto_ptr& std::auto_ptr< _Tp >::operator= ( auto_ptr< _Tp > & __a ) throw () [inline]`

auto\_ptr assignment operator.

#### Parameters

*a* Another auto\_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 134 of file auto\_ptr.h.

**5.354.4.6** `template<typename _Tp> element_type* std::auto_ptr< _Tp >::release ( ) throw () [inline]`

Bypassing the smart pointer.

**Returns**

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

**Note**

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 223 of file `auto_ptr.h`.

**5.354.4.7** `template<typename _Tp> void std::auto_ptr<_Tp>::reset ( element_type * __p = 0 ) throw () [inline]`

Forcibly deletes the managed object.

**Parameters**

*p* A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*. The previous object has been deleted.

Definition at line 238 of file `auto_ptr.h`.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

## 5.355 `std::auto_ptr_ref<_Tp1>` Struct Template Reference

**Public Member Functions**

- `auto_ptr_ref(_Tp1 *__p)`

**Public Attributes**

- `_Tp1 * _M_ptr`

### 5.355.1 Detailed Description

```
template<typename _Tp1> struct std::auto_ptr_ref< _Tp1 >
```

A wrapper class to provide [auto\\_ptr](#) with reference semantics. For example, an [auto\\_ptr](#) can be assigned (or constructed from) the result of a function which returns an [auto\\_ptr](#) by value.

All the [auto\\_ptr\\_ref](#) stuff should happen behind the scenes.

Definition at line 46 of file `auto_ptr.h`.

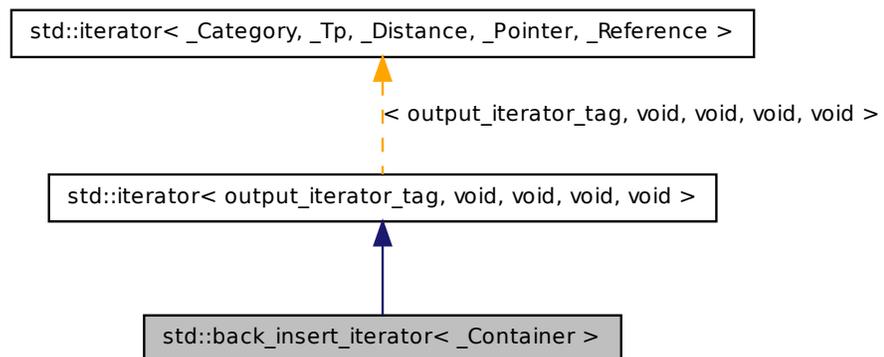
The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

## 5.356 `std::back_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::back_insert_iterator< _Container >`:



### Public Types

- typedef `_Container` [container\\_type](#)

- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- `back_insert_iterator` (`_Container &__x`)
- `back_insert_iterator` & `operator*` ()
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` `operator++` (int)
- `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container * container`

### 5.356.1 Detailed Description

`template<typename _Container> class std::back_insert_iterator<_Container>`

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 394 of file `stl_iterator.h`.

### 5.356.2 Member Typedef Documentation

**5.356.2.1** `template<typename _Container> typedef _Container std::back_insert_iterator<_Container>::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 402 of file `stl_iterator.h`.

**5.356.2.2** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.356.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.356.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file `stl_iterator_base_types.h`.

**5.356.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file `stl_iterator_base_types.h`.

**5.356.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

### 5.356.3 Constructor & Destructor Documentation

**5.356.3.1** `template<typename _Container > std::back_insert_iterator< _Container >::back_insert_iterator ( _Container & __x ) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 406 of file `stl_iterator.h`.

## 5.356.4 Member Function Documentation

**5.356.4.1** `template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator* ( )`  
**[inline]**

Simply returns \*this.

Definition at line 444 of file stl\_iterator.h.

**5.356.4.2** `template<typename _Container > back_insert_iterator std::back_insert_iterator< _Container >::operator++ ( int )`  
**[inline]**

Simply returns \*this. (This iterator does not *move*).

Definition at line 454 of file stl\_iterator.h.

**5.356.4.3** `template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator++ ( )`  
**[inline]**

Simply returns \*this. (This iterator does not *move*).

Definition at line 449 of file stl\_iterator.h.

**5.356.4.4** `template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator= ( const typename _Container::value_type & __value )` **[inline]**

### Parameters

*value* An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 428 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

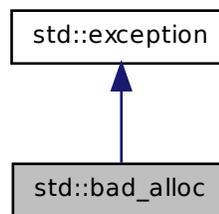
- [stl\\_iterator.h](#)

## 5.357 std::bad\_alloc Class Reference

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Inheritance diagram for `std::bad_alloc`:



### Public Member Functions

- virtual const char \* `what` () const throw ()

#### 5.357.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 56 of file `new`.

#### 5.357.2 Member Function Documentation

##### 5.357.2.1 virtual const char\* std::bad\_alloc::what ( ) const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

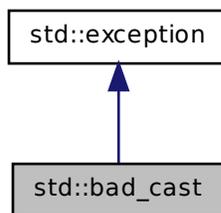
- [new](#)

## 5.358 `std::bad_cast` Class Reference

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Inheritance diagram for `std::bad_cast`:



### Public Member Functions

- virtual const char \* [what](#) () const throw ()

#### 5.358.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 174 of file `typeinfo`.

## 5.358.2 Member Function Documentation

### 5.358.2.1 `virtual const char* std::bad_cast::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

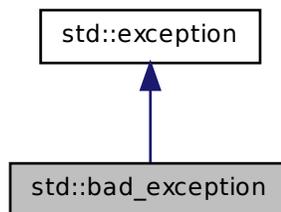
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.359 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



### Public Member Functions

- `virtual const char * what () const throw ()`

### 5.359.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 74 of file `exception`.

## 5.359.2 Member Function Documentation

### 5.359.2.1 `virtual const char* std::bad_exception::what ( ) const throw ( )` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

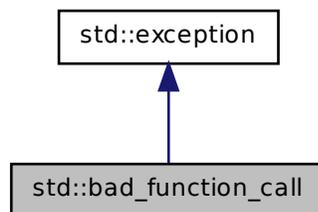
The documentation for this class was generated from the following file:

- [exception](#)

## 5.360 `std::bad_function_call` Class Reference

Exception class thrown when class template function's `operator()` is called with an empty target.

Inheritance diagram for `std::bad_function_call`:



### Public Member Functions

- `virtual const char * what ( ) const throw ( )`

### 5.360.1 Detailed Description

Exception class thrown when class template function's `operator()` is called with an empty target.

Definition at line 1417 of file functional.

## 5.360.2 Member Function Documentation

### 5.360.2.1 virtual const char\* std::exception::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_exception](#), [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_typeid](#), [std::future\\_error](#), [std::logic\\_error](#), [std::runtime\\_error](#), [std::tr1::bad\\_weak\\_ptr](#), and [std::ios\\_base::failure](#).

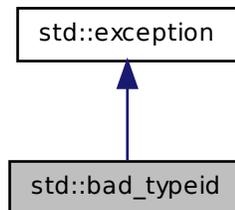
The documentation for this class was generated from the following file:

- [functional](#)

## 5.361 std::bad\_typeid Class Reference

Thrown when a NULL pointer in a `typeid` expression is used.

Inheritance diagram for `std::bad_typeid`:



### Public Member Functions

- virtual const char \* [what](#) () const throw ()

### 5.361.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 191 of file `typeinfo`.

### 5.361.2 Member Function Documentation

#### 5.361.2.1 `virtual const char* std::bad_typeid::what ( ) const throw ( )` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

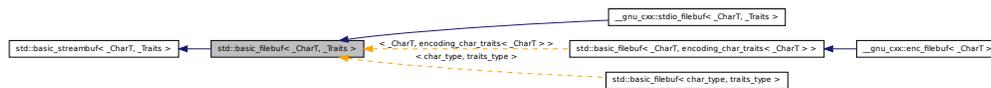
- [typeinfo](#)

## 5.362 `std::basic_filebuf< _CharT, _Traits >` Class Template Reference

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Inheritance diagram for `std::basic_filebuf< _CharT, _Traits >`:



### Public Types

- typedef `codecvt< char_type, char, __state_type >` **\_\_codecvt\_type**
- typedef `__basic_file< char >` **\_\_file\_type**
- typedef `basic_filebuf< char_type, traits_type >` **\_\_filebuf\_type**
- typedef `traits_type::state_type` **\_\_state\_type**
- typedef `basic_streambuf< char_type, traits_type >` **\_\_streambuf\_type**

- typedef `_CharT` [char\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)
- typedef `traits_type::pos_type` [pos\\_type](#)
- typedef `_Traits` [traits\\_type](#)

## Public Member Functions

- [basic\\_filebuf](#) ()
- virtual [~basic\\_filebuf](#) ()
- [\\_\\_filebuf\\_type \\* close](#) ()
- [streamsize in\\_avail](#) ()
- [bool is\\_open](#) () const throw ()
- [\\_\\_filebuf\\_type \\* open](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode)
- [\\_\\_filebuf\\_type \\* open](#) (const std::string & \_\_s, [ios\\_base::openmode](#) \_\_mode)
- [int\\_type sbumpc](#) ()
- [int\\_type sgetc](#) ()
- [streamsize sgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [int\\_type snextc](#) ()
- [int\\_type sputbackc](#) ([char\\_type](#) \_\_c)
- [int\\_type sputc](#) ([char\\_type](#) \_\_c)
- [streamsize sputn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- void [stoss](#) ()
- [int\\_type sungetc](#) ()

## Protected Member Functions

- void [\\_M\\_allocate\\_internal\\_buffer](#) ()
- [bool \\_M\\_convert\\_to\\_external](#) ([char\\_type](#) \*, [streamsize](#))
- void [\\_M\\_create\\_pback](#) ()
- void [\\_M\\_destroy\\_internal\\_buffer](#) () throw ()
- void [\\_M\\_destroy\\_pback](#) () throw ()
- [pos\\_type \\_M\\_seek](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [\\_\\_state\\_type](#) \_\_state)
- void [\\_M\\_set\\_buffer](#) ([streamsize](#) \_\_off)
- [bool \\_M\\_terminate\\_output](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) & \_\_loc)
- virtual [int\\_type overflow](#) ([int\\_type](#) \_\_c=\_Traits::eof())
- virtual [int\\_type overflow](#) ([int\\_type](#)=[traits\\_type::eof](#)())
- virtual [int\\_type pbackfail](#) ([int\\_type](#) \_\_c=\_Traits::eof())
- virtual [int\\_type pbackfail](#) ([int\\_type](#)=[traits\\_type::eof](#)())

- void `pbump` (int \_\_n)
  - virtual `pos_type seekoff` (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual `pos_type seekoff` (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual `pos_type seekpos` (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual `pos_type seekpos` (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual `__streambuf_type * setbuf` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual `basic_streambuf< char_type, _Traits > * setbuf` (char\_type \*, streamsize)
  - void `setg` (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
  - void `setp` (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
  - virtual `streamsize showmanyc` ()
  - virtual int `sync` ()
  - virtual `int_type uflow` ()
  - virtual `int_type underflow` ()
  - virtual `streamsize xsgetn` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual `streamsize xsgetn` (char\_type \*\_\_s, streamsize \_\_n)
  - virtual `streamsize xspun` (const char\_type \*\_\_s, streamsize \_\_n)
  - virtual `streamsize xspun` (const char\_type \*\_\_s, streamsize \_\_n)
- 
- `char_type * eback` () const
  - `char_type * gptr` () const
  - `char_type * egptr` () const
- 
- `char_type * pbase` () const
  - `char_type * pptr` () const
  - `char_type * epptr` () const

## Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`

- [bool \\_M\\_reading](#)
  - [\\_\\_state\\_type \\_M\\_state\\_beg](#)
  - [\\_\\_state\\_type \\_M\\_state\\_cur](#)
  - [\\_\\_state\\_type \\_M\\_state\\_last](#)
  - [bool \\_M\\_writing](#)
- 
- [char\\_type \\_M\\_pback](#)
  - [char\\_type \\* \\_M\\_pback\\_cur\\_save](#)
  - [char\\_type \\* \\_M\\_pback\\_end\\_save](#)
  - [bool \\_M\\_pback\\_init](#)

## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`  
`_CharT2 >, _CharT2 *)`
  - `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`  
`bool &)`
  - `class basic_ios< char_type, traits_type >`
  - `class basic_istream< char_type, traits_type >`
  - `class basic_ostream< char_type, traits_type >`
  - `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_-`  
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, -`  
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, -`  
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
  - `class ios_base`
  - `class istreambuf_iterator< char_type, traits_type >`
  - `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, _CharT2 *)`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
  - `class ostreambuf_iterator< char_type, traits_type >`
- 
- [char\\_type \\* \\_M\\_in\\_beg](#)
  - [char\\_type \\* \\_M\\_in\\_cur](#)
  - [char\\_type \\* \\_M\\_in\\_end](#)
  - [char\\_type \\* \\_M\\_out\\_beg](#)

- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `locale _M_buf_locale`
- `locale pubimbue (const locale & __loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type * __s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `int pubsync ()`

### 5.362.1 Detailed Description

**template<typename \_CharT, typename \_Traits> class std::basic\_filebuf< \_CharT, \_Traits >**

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Definition at line 67 of file `fstream`.

### 5.362.2 Member Typedef Documentation

**5.362.2.1 template<typename \_CharT, typename \_Traits> typedef basic\_streambuf<char\_type, traits\_type> std::basic\_filebuf< \_CharT, \_Traits >::\_\_streambuf\_type**

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, _Traits >`.

Definition at line 77 of file `fstream`.

**5.362.2.2 template<typename \_CharT, typename \_Traits> typedef \_CharT std::basic\_filebuf< \_CharT, \_Traits >::char\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits >`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 71 of file `fstream`.

**5.362.2.3 `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_filebuf<_CharT, _Traits >::int_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 73 of file `fstream`.

**5.362.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_filebuf<_CharT, _Traits >::off_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 75 of file `fstream`.

**5.362.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_filebuf<_CharT, _Traits >::pos_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT >](#), and [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits >](#).

Definition at line 74 of file `fstream`.

### 5.362.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_filebuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Reimplemented in `__gnu_cxx::enc_filebuf<_CharT>`, and `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`.

Definition at line 72 of file `fstream`.

## 5.362.3 Constructor & Destructor Documentation

### 5.362.3.1 `template<typename _CharT, typename _Traits> std::basic_filebuf< _CharT, _Traits>::basic_filebuf ( )`

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file `fstream.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::_M_buf_locale`.

### 5.362.3.2 `template<typename _CharT, typename _Traits> virtual std::basic_filebuf<_CharT, _Traits>::~~basic_filebuf ( ) [inline, virtual]`

The destructor closes the file first.

Definition at line 214 of file `fstream`.

## 5.362.4 Member Function Documentation

### 5.362.4.1 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::_M_create_pback ( ) [inline, protected]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

**5.362.4.2** `template<typename _CharT, typename _Traits> void  
std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback ( ) throw  
( ) [inline, protected]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.4.3** `template<typename _CharT, typename _Traits> void  
std::basic_filebuf< _CharT, _Traits >::_M_set_buffer ( streamsize  
__off ) [inline, protected]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `eptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 387 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.4.4** `template<typename _CharT , typename _Traits > basic_filebuf<  
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,  
_Traits >::close ( )`

Closes the currently associated file.

### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.362.4.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eback ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.362.4.8** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

#### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

**5.362.4.9** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::getloc ( ) const  
[inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

**5.362.4.10** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.4.11** `template<typename _CharT, typename _Traits > void  
std::basic_filebuf< _CharT, _Traits >::imbue ( const locale & )  
[protected, virtual]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 855 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

**5.362.4.12** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::in_avail ( )  
[inline, inherited]`

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file streambuf.

**5.362.4.13** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::is_open ( ) const throw ( )  
[inline]`

Returns true if the external file is open.

Definition at line 222 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::close()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, and `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`.

**5.362.4.14** `template<typename _CharT , typename _Traits > basic_filebuf<  
_CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT,  
_Traits >::open ( const char * __s, ios_base::openmode __mode )`

Opens an external file.

**Parameters**

*s* The name of the file.

*mode* The open mode flags.

**Returns**

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

	<code>ios_base</code> Flag combination	stdio equivalent	binary	in	out	trunc	app
+-----+	+ w	+ a	+ w	+	+	+	+
r     + + r+     + + + w+     + + + a+     + + a+   +-----+	+ + w	+ + a	+ + w	+	+	+	+
-----+   + + wb     + + + ab     + + ab     + + + wb     + + rb     + + + r+b     + + + +	+ + wb	+ + ab	+ + rb	+ + +	+ + +	+ + +	+ + +
w+b     + + + + a+b     + + + a+b   +-----+	+ + + w	+ + + a	+ + + w	+ + +	+ + +	+ + +	+ + +

Definition at line 94 of file fstream.tcc.

References std::basic\_filebuf< \_CharT, \_Traits >::\_M\_mode, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_reading, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer(), std::ios\_base::ate, std::basic\_filebuf< \_CharT, \_Traits >::close(), std::ios\_base::end, and std::basic\_filebuf< \_CharT, \_Traits >::is\_open().

**5.362.4.15** `template<typename _CharT, typename _Traits> __filebuf_type*  
std::basic_filebuf< _CharT, _Traits >::open ( const std::string &  
_s, ios_base::openmode _mode ) [inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

#### Returns

*this* on success, NULL on failure

Definition at line 275 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::open().

**5.362.4.16** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::overflow ( int_type =  
traits_type::eof() ) [inline, protected, virtual,  
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

- c* An additional character to consume.

#### Returns

eof() to indicate failure, something else (usually *c*, or not\_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns eof().

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 746 of file streambuf.

Referenced by [std::basic\\_streambuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**5.362.4.17** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::pbackfail ( int_type =  
traits_type::eof() ) [inline, protected, virtual,  
inherited]`

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

eof() on failure, *some other value* on success

#### Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

#### Note

Base class version does nothing, returns eof().

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 702 of file streambuf.

**5.362.4.18** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

**5.362.4.19** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

- `n` The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.362.4.20** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 510 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.362.4.21** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale  
& _loc ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

**5.362.4.22** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

**5.362.4.23** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 244 of file streambuf.

**5.362.4.24** `template<typename _CharT, typename _Traits>  
 __streambuf_type* std::basic_streambuf< _CharT, _Traits  
 >::pubsetbuf( char_type * __s, streamsize __n ) [inline,  
 inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file streambuf.

**5.362.4.25** `template<typename _CharT, typename _Traits> int  
 std::basic_streambuf< _CharT, _Traits >::pubsync ( )  
 [inline, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 249 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.362.4.26** `template<typename _CharT, typename _Traits> int_type  
 std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline,  
 inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits>::operator++()`.

```
5.362.4.27 template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff
( off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out ) [inline, protected,
virtual, inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

```
5.362.4.28 template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf<_CharT, _Traits>::seekpos ( pos_type,
ios_base::openmode = ios_base::in | ios_base::out ) [inline,
protected, virtual, inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

```
5.362.4.29 template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits>::__streambuf_type * std::basic_filebuf<
_CharT, _Traits>::setbuf ( char_type * __s, streamsize __n )
[protected, virtual]
```

Manipulates the buffer.

**Parameters**

- s* Pointer to a buffer area.
- n* Size of *s*.

**Returns**

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 657 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.362.4.30** `template<typename _CharT, typename _Traits> virtual basic_streambuf<char_type, _Traits>* std::basic_streambuf<_CharT, _Traits>::setbuf( char_type*, streamsize ) [inline, protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

**5.362.4.31** `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setg( char_type* __gbeg, char_type* __gnext, char_type* __gend ) [inline, protected, inherited]`

Setting the three read area pointers.

**Parameters**

- gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

#### Postcondition

*pbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

**5.362.4.32** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setp ( char_type  
* __pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

#### Parameters

*pbeg* A pointer.

*pend* A pointer.

#### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `pptr()`

Definition at line 533 of file `streambuf`.

**5.362.4.33** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.362.4.34** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for xsgetn.

#### Parameters

*s* A buffer area.

*n* A count.

Returns xsgetn(s,n). The effect is to fill s[0] through s[n-1] with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

**5.362.4.35** `template<typename _CharT , typename _Traits > streamsize  
std::basic_filebuf< _CharT, _Traits >::showmanyc ( )  
[protected, virtual]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to [underflow\(\)](#) will not return `traits::eof()` until at least that number of characters have been supplied. If [showmanyc\(\)](#) returns -1, then calls to [underflow\(\)](#) or [uflow\(\)](#) will fail.*  
[27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 178 of file fstream.tcc.

References [std::basic\\_filebuf< \\_CharT, \\_Traits >::\\_M\\_mode](#), [std::ios\\_base::binary](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gptr\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::is\\_open\(\)](#).

**5.362.4.36** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.362.4.37** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputbackc ( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

#### Parameters

*c* The character to push back.

#### Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.362.4.38** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

**Returns**

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.362.4.39** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sputn ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.362.4.40** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::stoss ( ) [inline,  
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.362.4.41** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.362.4.42** `template<typename _CharT , typename _Traits > int  
std::basic_filebuf< _CharT, _Traits >::sync ( ) [protected,  
virtual]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 838 of file fstream.tcc.

References `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

**5.362.4.43** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 678 of file `streambuf`.

**5.362.4.44** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::underflow( ) [protected, virtual]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::M_mode`, `std::basic_filebuf<_CharT, _Traits>::M_reading`, `std::basic_filebuf<_CharT, _Traits>::M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, and `std::min()`.

**5.362.4.45** `template<typename _CharT , typename _Traits > streamsize  
std::basic_streambuf< _CharT, _Traits >::xsgetn ( char_type *  
__s, streamsize __n ) [protected, virtual, inherited]`

Multiple character extraction.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

#### Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

**5.362.4.46** `template<typename _CharT , typename _Traits > streamsize  
std::basic_streambuf< _CharT, _Traits >::xsputn ( const  
char_type * __s, streamsize __n ) [protected, virtual,  
inherited]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

## 5.362.5 Member Data Documentation

### 5.362.5.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf` [protected]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

### 5.362.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

### 5.362.5.3 `template<typename _CharT, typename _Traits> size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size` [protected]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.362.5.4** `template<typename _CharT, typename _Traits> char*  
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 151 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.5.5** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size  
[protected]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.5.6** `template<typename _CharT, typename _Traits> const  
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next  
[protected]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to [egptr\(\)](#).

Definition at line 163 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_beg  
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

**5.362.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.362.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.362.5.10** `template<typename _CharT, typename _Traits>  
ios_base::openmode std::basic_filebuf< _CharT, _Traits  
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.362.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

**5.362.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

**5.362.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

**5.362.5.14** `template<typename _CharT, typename _Traits> char_type  
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

**5.362.5.15** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  
[protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

**5.362.5.16** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  
[protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

**5.362.5.17** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_pback_init  
[protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

**5.362.5.18** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_reading  
[protected]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

The documentation for this class was generated from the following files:

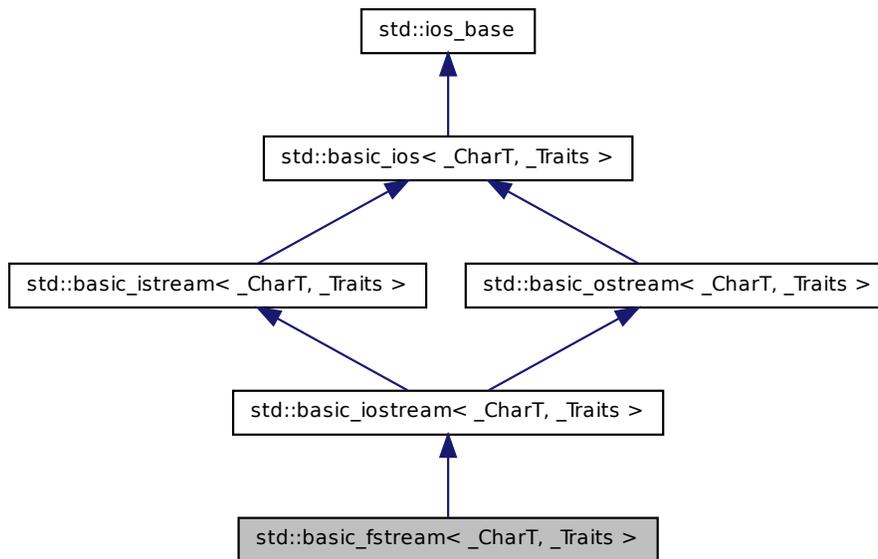
- [fstream](#)
- [fstream.tcc](#)

## 5.363 `std::basic_fstream< _CharT, _Traits >` Class Template Reference

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `s.b`.

Inheritance diagram for `std::basic_fstream<_CharT, _Traits >`:



## Public Types

- typedef `ctype<_CharT > __ctype_type`
- typedef `ctype<_CharT > __ctype_type`
- typedef `basic_filebuf<char_type, traits_type > __filebuf_type`
- typedef `basic_ios<_CharT, _Traits > __ios_type`
- typedef `basic_ios<char_type, traits_type > __ios_type`
- typedef `basic_iostream<char_type, traits_type > __iostream_type`
- typedef `basic_istream<_CharT, _Traits > __istream_type`
- typedef `basic_istream<_CharT, _Traits > __istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > __num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __num_put_type`
- typedef `basic_ostream<_CharT, _Traits > __ostream_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`

- typedef `_CharT` [char\\_type](#)
- typedef `_CharT` [char\\_type](#)
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback` )(event, ios\_base &, int)
- typedef `_Ios_Fmtflags` [fmtflags](#)
- typedef traits\_type::int\_type [int\\_type](#)
- typedef `_Traits::int_type` [int\\_type](#)
- typedef int `io_state`
- typedef `_Ios_Iostate` [iostate](#)
- typedef traits\_type::off\_type [off\\_type](#)
- typedef `_Traits::off_type` [off\\_type](#)
- typedef int `open_mode`
- typedef `_Ios_Openmode` [openmode](#)
- typedef `_Traits::pos_type` [pos\\_type](#)
- typedef traits\_type::pos\_type [pos\\_type](#)
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` [streamoff](#)
- typedef `std::streampos` [streampos](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `_Traits` [traits\\_type](#)
  
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >` [\\_\\_num\\_put\\_type](#)

## Public Member Functions

- [basic\\_fstream](#) ()
- [basic\\_fstream](#) (const char \*\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [basic\\_fstream](#) (const std::string &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [~basic\\_fstream](#) ()
- const `locale` & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) (iostate \_\_state)
- bool [bad](#) () const
- void [clear](#) (iostate \_\_state=goodbit)
- void [close](#) ()
- `basic_ios` & [copyfmt](#) (const `basic_ios` &\_\_rhs)
- bool [eof](#) () const
- `iostate` [exceptions](#) () const
- void [exceptions](#) (iostate \_\_except)

- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `bool is_open ()`
- `bool is_open () const`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `__filebuf_type * rdbuf () const`
- `iostate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`

- [fmtflags setf](#) ([fmtflags](#) \_\_fmtfl)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [pos\\_type tellp](#) ()
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tistr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type widen](#) (char \_\_c) const
  - [streamsize width](#) ([streamsize](#) \_\_wide)
  - [streamsize width](#) () const
- 
- [\\_\\_istream\\_type & operator>>](#) ([\\_\\_istream\\_type](#) &(\_\_pf)(\_\_istream\_type &))
  - [\\_\\_istream\\_type & operator>>](#) ([\\_\\_ios\\_type](#) &(\_\_pf)(\_\_ios\_type &))
  - [\\_\\_istream\\_type & operator>>](#) ([ios\\_base](#) &(\_\_pf)([ios\\_base](#) &))

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_istream\\_type & operator>>](#) (bool &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (unsigned long long &\_\_n)
- [\\_\\_istream\\_type & operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type & operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type & operator>>](#) (long double &\_\_f)
- [\\_\\_istream\\_type & operator>>](#) (void \*&\_\_p)
- [\\_\\_istream\\_type & operator>>](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
  
- `operator void * () const`
- `bool operator! () const`
  
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type * __sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`

- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename `_ValueT` >  
  `__istream_type` & `_M_extract` (`_ValueT` &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)

- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  \_\_ostream\_type & **\_M\_insert** (\_ValueT \_\_v)
- void **init** (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)

### Protected Attributes

- \_Callback\_list \* **\_M\_callbacks**
- const \_\_ctype\_type \* **\_M\_ctype**
- iostate **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- fmtflags **\_M\_flags**
- streamsize **\_M\_gcount**
- locale **\_M\_ios\_locale**
- \_Words **\_M\_local\_word** [\_S\_local\_word\_size]
- const \_\_num\_get\_type \* **\_M\_num\_get**
- const \_\_num\_put\_type \* **\_M\_num\_put**
- streamsize **\_M\_precision**
- basic\_streambuf< \_CharT, \_Traits > \* **\_M\_streambuf**
- iostate **\_M\_streambuf\_state**
- basic\_ostream< \_CharT, \_Traits > \* **\_M\_tie**
- streamsize **\_M\_width**
- \_Words \* **\_M\_word**
- int **\_M\_word\_size**
- \_Words **\_M\_word\_zero**

### Friends

- class **sentry**
- class **sentry**

### 5.363.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_fstream< _CharT, _Traits >
```

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 756 of file `fstream`.

## 5.363.2 Member Typedef Documentation

**5.363.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream<_CharT, _Traits>::_ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file `istream`.

**5.363.2.2** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream<_CharT, _Traits>::_ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file `ostream`.

**5.363.2.3** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_istream<_CharT, _Traits>::_num_get_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `istream`.

**5.363.2.4** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::_num_put_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

**5.363.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits >::_num_put_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.363.2.6** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_istream<_CharT, _Traits >::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits >](#), and [std::basic\\_istringstream<\\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 59 of file istream.

**5.363.2.7** `template<typename _CharT, typename _Traits > typedef _CharT std::basic_fstream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits >](#).

Definition at line 760 of file fstream.

**5.363.2.8** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the [ios\\_base](#) object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file [ios\\_base.h](#).

### **5.363.2.9 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags [inherited]**

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file [ios\\_base.h](#).

**5.363.2.10** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 60 of file `istream`.

**5.363.2.11** `template<typename _CharT , typename _Traits > typedef  
traits_type::int_type std::basic_fstream< _CharT, _Traits  
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 762 of file `fstream`.

**5.363.2.12** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

**5.363.2.13** `template<typename _CharT, typename _Traits> typedef`  
`_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type`  
`[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), and [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 62 of file `istream`.

**5.363.2.14** `template<typename _CharT, typename _Traits> typedef`  
`traits_type::off_type std::basic_fstream<_CharT, _Traits`  
`>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 764 of file `fstream`.

**5.363.2.15** `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

**5.363.2.16** `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_fstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 763 of file `fstream`.

**5.363.2.17** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 61 of file `istream`.

**5.363.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

**5.363.2.19** `template<typename _CharT , typename _Traits > typedef _Traits std::basic_fstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 761 of file `fstream`.

**5.363.2.20** `template<typename _CharT, typename _Traits> typedef`  
`_Traits std::basic_istream<_CharT, _Traits>::traits_type`  
`[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), and [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 63 of file `istream`.

### 5.363.3 Member Enumeration Documentation

**5.363.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

### 5.363.4 Constructor & Destructor Documentation

**5.363.4.1** `template<typename _CharT, typename _Traits>`  
`std::basic_fstream<_CharT, _Traits>::basic_fstream ( )`  
`[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 783 of file `fstream`.

**5.363.4.2** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::basic_fstream ( const char *  
__s, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, explicit]`

Create an input/output file stream.

#### Parameters

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 796 of file `fstream`.

**5.363.4.3** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::basic_fstream  
( const std::string & __s, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, explicit]`

Create an input/output file stream.

#### Parameters

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

Definition at line 811 of file `fstream`.

**5.363.4.4** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::~~basic_fstream ( )  
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 826 of file `fstream`.

## 5.363.5 Member Function Documentation

### 5.363.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline, inherited]

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::time\_put< \_CharT, \_OutIter >::put().

### 5.363.5.2 template<typename \_CharT, typename \_Traits> void std::basic\_ostream< \_CharT, \_Traits >::\_M\_write ( const char\_type \* \_\_s, streamsize \_\_n ) [inline, inherited]

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

### 5.363.5.3 template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::bad ( ) const [inline, inherited]

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

**5.363.5.4** `template<typename _CharT , typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file basic\_ios.tcc.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.363.5.5** `template<typename _CharT , typename _Traits > void  
std::basic_fstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 906 of file fstream.

**5.363.5.6** `template<typename _CharT , typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (   
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

**Parameters**

*\_\_rhs* The source values for the copies.

**Returns**

Reference to this object.

## 5.363 `std::basic_fstream<_CharT, _Traits>` Class Template Reference 1501

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

### 5.363.5.7 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof( ) const` [`inline`, `inherited`]

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.363.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions( ) const` [`inline`, `inherited`]

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.363.5.9** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

**5.363.5.10** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.363.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file basic\_ios.h.

**5.363.5.12** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt().

**5.363.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

**5.363.5.14** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 550 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.363.5.15** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf ()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ()->pubsync ()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.363.5.16** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline, inherited]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

**5.363.5.17** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get( void ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.18** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get( char_type & __c ) [inherited]`

Simple extraction.

**Parameters**

`c` The character in which to store data.

**Returns**

`*this`

Tries to extract a character and store it in `c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.19** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n, char_type __delim )`  
**[inherited]**

Simple multiple-character extraction.

#### Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, `failbit` is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.363.5.20** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::sputc()`.

**5.363.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

**5.363.5.22** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

**Returns**

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file `istream`.

**5.363.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

**Parameters**

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

**5.363.5.24** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim )`  
**[inherited]**

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::nextc()`.

**5.363.5.25** `locale std::ios_base::getloc ( ) const` **[inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(locale)` has previously been called, then this function returns `locale`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.363.5.26** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good() const [inline,  
inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.363.5.27** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore  
(void) [inherited]`

Discarding characters.

**Parameters**

*n* Number of characters to discard.

*delim* A "stop" character.

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### 5.363.5.28 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( streamsize __n ) [inherited]`

Simple extraction.

##### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

#### 5.363.5.29 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

##### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.363.5.30** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file basic\_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.363.5.31** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic\_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.363.5.32** `template<typename _CharT, typename _Traits > bool  
std::basic_fstream<_CharT, _Traits >::is_open ( ) [inline]`

Wrapper to test for an open file.

**Returns**

`rdbuf () -> is_open ()`

Definition at line 845 of file fstream.

**5.363.5.33** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.363.5.34** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits >::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

```
5.363.5.35 template<typename _CharT , typename _Traits > void
std::basic_fstream< _CharT, _Traits >::open ( const char * __s,
ios_base::openmode __mode = ios_base::in | ios_base::out )
[inline]
```

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 866 of file `fstream`.

```
5.363.5.36 template<typename _CharT , typename _Traits > void
std::basic_fstream< _CharT, _Traits >::open ( const std::string &
__s, ios_base::openmode __mode = ios_base::in | ios_base::out )
[inline]
```

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 887 of file `fstream`.

**5.363.5.37** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.363.5.38** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.363.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

**5.363.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

**5.363.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

**5.363.5.42** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

## 5.363 `std::basic_fstream<_CharT, _Traits>` Class Template Reference 1517

Definition at line 92 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.363.5.43** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ostream_type &(*)(__ostream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

**5.363.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

**5.363.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

**5.363.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

**5.363.5.47** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( bool __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

**5.363.5.48** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

**5.363.5.49** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( float __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

**5.363.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

**5.363.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

**5.363.5.52** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.363.5.53** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<<( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

**5.363.5.54** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.363.5.55** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.363.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**5.363.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.363.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

**5.363.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

**5.363.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**5.363.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

```
5.363.5.62 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( unsigned int
& __n ) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

```
5.363.5.63 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( double &
__f ) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

```
5.363.5.64 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( bool & __n
) [inline, inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

**5.363.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

**5.363.5.66** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.67** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file istream.

**5.363.5.69** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file `istream`.

**5.363.5.70** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

**5.363.5.71** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.72** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

`A` variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

**5.363.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void ) [inherited]`

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.74** streamsize std::ios\_base::precision ( ) const [inline, inherited]

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.363.5.75** streamsize std::ios\_base::precision ( streamsize \_\_prec ) [inline, inherited]

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of [precision\(\)](#).

Definition at line 629 of file ios\_base.h.

**5.363.5.76** template<typename \_CharT , typename \_Traits > basic\_ostream< \_CharT, \_Traits > & std::basic\_ostream< \_CharT, \_Traits >::put ( char\_type \_\_c ) [inherited]

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**5.363.5.77** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::putback( char_type __c ) [inherited]`

Unextracting a single character.

#### Parameters

`c` The character to push back into the input stream.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**5.363.5.78** `void*& std::ios_base::pword( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

**5.363.5.79** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

**Parameters**

*sb* The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.363.5.80** `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_fstream<_CharT, _Traits>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 837 of file `fstream`.

**5.363.5.81** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.363.5.82** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::read(  
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

**Parameters**

*s* A character array.

*n* Maximum number of characters to store.

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.83** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

**Parameters**

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the `streambuf`'s buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets `eofbit` and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.84** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

- \_\_fn* The function to add.
- \_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.363.5.85** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

**Parameters**

- off* A file offset object.
- dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.363.5.86** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

**Parameters**

- pos* A file position object.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.363.5.87** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.363.5.88** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

- off* A file offset object.
- dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.363.5.89 `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

**Parameters**

- fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

### 5.363.5.90 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline, inherited]`

Setting new format flags.

**Parameters**

- fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios\\_base::adjustfield](#).

Definition at line 594 of file ios\_base.h.

**5.363.5.91** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic\_ios.h.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::peek\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::read\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**5.363.5.92** `template<typename _CharT , typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.363.5.93** `static bool std::ios_base::sync_with_stdio( bool __sync = true )`  
**[static, inherited]**

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.363.5.94** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg( void )`  
**[inherited]**

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

#### 5.363.5.95 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp()` **[inherited]**

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

#### 5.363.5.96 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie()` **const [inline, inherited]**

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.363.5.97** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( basic_ostream<_CharT, _Traits > * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

**5.363.5.98** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::unget  
( void ) [inherited]`

Unextracting the previous character.

#### Returns

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sungetc()`.

**5.363.5.99** `void std::ios_base::unsetf ( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.363.5.100** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

### Parameters

*c* The character to widen.

### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.363.5.101** `streamsize std::ios_base::width ( streamsize __wide ) [inline,  
inherited]`

Changing flags.

### Parameters

*wide* The new width value.

### Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.363.5.102** `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.363.5.103** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

### Parameters

*s* The array to insert.

*n* Maximum number of characters to insert.

### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

### Note

This function is not overloaded on signed `char` and unsigned `char`.

**5.363.5.104** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.363.6 Member Data Documentation****5.363.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.363.6.2** `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

**5.363.6.3** `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.363.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.363.6.5 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.363.6.6 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.363.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

**5.363.6.8 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### **5.363.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

#### **5.363.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### **5.363.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### **5.363.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

#### **5.363.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter

>::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

#### 5.363.6.14 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

#### 5.363.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

#### 5.363.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of *setf*.

Definition at line 324 of file ios\_base.h.

#### 5.363.6.17 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),

std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### **5.363.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### **5.363.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

#### **5.363.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

#### **5.363.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.363.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.363.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::seekp(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.363.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file ios\_base.h.

**5.363.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.363.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios\_base.h.

**5.363.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.363.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

## 5.364 `std::basic_ifstream<_CharT, _Traits >` Class Template Reference 1549

Definition at line 305 of file `ios_base.h`.

### 5.363.6.29 `const fmtflags std::ios_base::skipws` [static, inherited]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

### 5.363.6.30 `const openmode std::ios_base::trunc` [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

### 5.363.6.31 `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

### 5.363.6.32 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outputter >::do_put()`.

The documentation for this class was generated from the following file:

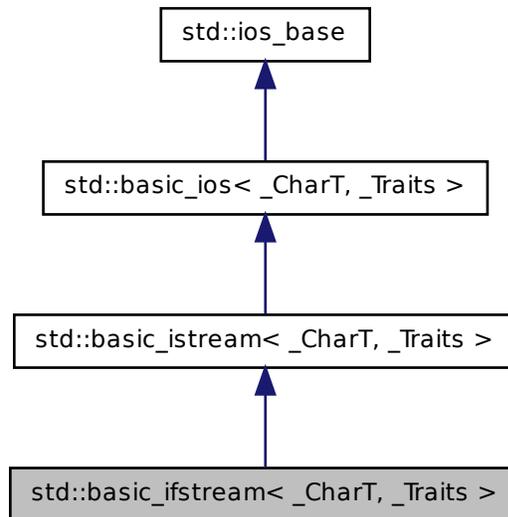
- [fstream](#)

## 5.364 `std::basic_ifstream<_CharT, _Traits >` Class Template Reference

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ifstream< _CharT, _Traits >`:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int `open_mode`
  - typedef `_Ios_Openmode` `openmode`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef int `seek_dir`
  - typedef `_Ios_Seekdir` `seekdir`
  - typedef `std::streamoff` `streamoff`
  - typedef `std::streampos` `streampos`
  - typedef `_Traits` `traits_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > >` `__-`  
`num_put_type`

## Public Member Functions

- `basic_ifstream ()`
- `basic_ifstream (const char *__s, ios_base::openmode __mode=ios_base::in)`
- `basic_ifstream (const std::string &__s, ios_base::openmode __mode=ios_base::in)`
- `~basic_ifstream ()`
- `const locale & _M_getloc () const`
- `void _M_setstate (iostate __state)`
- `bool bad () const`
- `void clear (iostate __state=goodbit)`
- `void close ()`
- `basic_ios & copyfmt (const basic_ios &__rhs)`
- `bool eof () const`
- `iostate exceptions () const`
- `void exceptions (iostate __except)`
- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `streamsize gcount () const`
- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `locale getloc () const`
- `bool good () const`

- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale & __loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `void open (const char * __s, ios_base::openmode __mode=ios_base::in)`
  - `void open (const std::string & __s, ios_base::openmode __mode=ios_base::in)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__filebuf_type * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `void setstate (iosate __state)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several

effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>>` (`bool &__n`)
- `__istream_type & operator>>` (`short &__n`)
- `__istream_type & operator>>` (`unsigned short &__n`)
- `__istream_type & operator>>` (`int &__n`)
- `__istream_type & operator>>` (`unsigned int &__n`)
- `__istream_type & operator>>` (`long &__n`)
- `__istream_type & operator>>` (`unsigned long &__n`)
- `__istream_type & operator>>` (`long long &__n`)
- `__istream_type & operator>>` (`unsigned long long &__n`)
- `__istream_type & operator>>` (`float &__f`)
- `__istream_type & operator>>` (`double &__f`)
- `__istream_type & operator>>` (`long double &__f`)
- `__istream_type & operator>>` (`void *&__p`)
- `__istream_type & operator>>` (`__streambuf_type * __sb`)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ifstream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`

- [\\_\\_istream\\_type & ignore \(streamsize \\_\\_n, int\\_type \\_\\_delim\)](#)
  - [int\\_type peek \(\)](#)
  - [\\_\\_istream\\_type & read \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [streamsize readsome \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [\\_\\_istream\\_type & putback \(char\\_type \\_\\_c\)](#)
  - [\\_\\_istream\\_type & unget \(\)](#)
  - [int sync \(\)](#)
  - [pos\\_type tellg \(\)](#)
  - [\\_\\_istream\\_type & seekg \(pos\\_type\)](#)
  - [\\_\\_istream\\_type & seekg \(off\\_type, ios\\_base::seekdir\)](#)
- 
- [operator void \\* \(\) const](#)
  - [bool operator! \(\) const](#)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)

- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename `_ValueT` >  
`__istream_type` & `_M_extract` (`_ValueT` &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iostate` `_M_streambuf_state`

- [basic\\_ostream](#)<\_CharT, \_Traits > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- [int](#) [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

## Friends

- class [sentry](#)

### 5.364.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ifstream< _CharT, _Traits >`

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 415 of file `fstream`.

### 5.364.2 Member Typedef Documentation

**5.364.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic\_istream<_CharT, _Traits >::\_ctype\_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios](#)<\_CharT, \_Traits >.

Definition at line 71 of file `istream`.

**5.364.2.2** `template<typename _CharT, typename _Traits> typedef num\_get<_CharT, istreambuf\_iterator<_CharT, _Traits> > std::basic\_istream<_CharT, _Traits >::\_num\_get\_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios](#)<\_CharT, \_Traits >.

Definition at line 70 of file `istream`.

**5.364.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits >::_num_put_type` **[inherited]**

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits >`, `std::basic_ostream<char, _Traits >`, and `std::basic_ostream<char >`.

Definition at line 84 of file `basic_ios.h`.

**5.364.2.4** `template<typename _CharT, typename _Traits > typedef _CharT std::basic_ifstream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits >`.

Definition at line 419 of file `fstream`.

**5.364.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` **[inherited]**

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

**5.364.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios\_base.h.

**5.364.2.7** `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ifstream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 421 of file fstream.

**5.364.2.8 typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]**

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 338 of file *ios\_base.h*.

**5.364.2.9 template<typename \_CharT , typename \_Traits > typedef traits\_type::off\_type std::basic\_ifstream<\_CharT, \_Traits >::off\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits >](#).

Definition at line 423 of file *fstream*.

**5.364.2.10 typedef \_Ios\_Openmode std::ios\_base::openmode [inherited]**

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 369 of file *ios\_base.h*.

**5.364.2.11** `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_istream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 422 of file `fstream`.

**5.364.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

**5.364.2.13** `template<typename _CharT , typename _Traits > typedef _Traits std::basic_istream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 420 of file `fstream`.

**5.364.3 Member Enumeration Documentation****5.364.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

## 5.364.4 Constructor & Destructor Documentation

**5.364.4.1** `template<typename _CharT, typename _Traits >  
std::basic_ifstream<_CharT, _Traits >::basic_ifstream ( )  
[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 441 of file `fstream`.

**5.364.4.2** `template<typename _CharT, typename _Traits >  
std::basic_ifstream<_CharT, _Traits >::basic_ifstream ( const char  
* __s, ios_base::openmode __mode = ios_base::in ) [inline,  
explicit]`

Create an input file stream.

### Parameters

`s` Null terminated string specifying the filename.

`mode` Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::in](#) is automatically included in `mode`.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 455 of file `fstream`.

**5.364.4.3** `template<typename _CharT, typename _Traits >  
std::basic_ifstream<_CharT, _Traits >::basic_ifstream ( const  
std::string & __s, ios_base::openmode __mode = ios_base::in )  
[inline, explicit]`

Create an input file stream.

### Parameters

`s` `std::string` specifying the filename.

`mode` Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::in](#) is automatically included in `mode`.

Definition at line 471 of file `fstream`.

**5.364.4.4** `template<typename _CharT, typename _Traits >  
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream ( )  
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 486 of file fstream.

## 5.364.5 Member Function Documentation

**5.364.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline,  
inherited]`

Locale access.

### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

**5.364.5.2** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

**5.364.5.3** `template<typename _CharT , typename _Traits > void  
std::basic_ios<_CharT, _Traits >::clear ( iosstate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits >::rdbuf()`.

**5.364.5.4** `template<typename _CharT , typename _Traits > void  
std::basic_ifstream<_CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 564 of file `fstream`.

**5.364.5.5** `template<typename _CharT , typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios<_CharT, _Traits >::copyfmt (   
const basic_ios<_CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

*\_\_rhs* The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### 5.364.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof( ) const [inline, inherited]`

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.364.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

#### 5.364.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

##### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

#### 5.364.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail( ) const` [`inline`, `inherited`]

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

#### 5.364.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill( char_type __ch )` [`inline`, `inherited`]

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.364.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.364.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

**5.364.5.13** `fmtflags std::ios_base::flags( ) const [inline, inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.364.5.14** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount( ) const [inline, inherited]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

**5.364.5.15** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get( void ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.364.5.16** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

*c* The character in which to store data.

#### Returns

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.364.5.17** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored

- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_ifstream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.364.5.18** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

**5.364.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

#### Returns

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file istream.

**5.364.5.20** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

**5.364.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

#### Returns

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.364.5.22** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case `eofbit` is set in the stream error state
2. the next character equals `delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored

3.  $n-1$  characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sputc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::getc()`, and `std::basic_streambuf<_CharT, _Traits>::nextc()`.

#### 5.364.5.23 locale `std::ios_base::getloc ( ) const` [`inline`, `inherited`]

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

#### 5.364.5.24 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good ( ) const` [`inline`, `inherited`]

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file basic\_ios.h.

**5.364.5.25** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_ifstream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.364.5.26** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.364.5.27** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.364.5.28** `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

## 5.364 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference 1575

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.364.5.29** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.364.5.30** `template<typename _CharT, typename _Traits> bool  
std::basic_ifstream<_CharT, _Traits>::is_open ( ) [inline]`

Wrapper to test for an open file.

### Returns

`rdbuf() -> is_open()`

Definition at line 505 of file `fstream`.

**5.364.5.31** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

### Parameters

`__ix` Index into the array.

### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.364.5.32** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

*c* The character to narrow.

*dfault* The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.364.5.33** `template<typename _CharT , typename _Traits > void  
std::basic_ifstream<_CharT, _Traits >::open ( const std::string &  
__s, ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 546 of file `fstream`.

**5.364.5.34** `template<typename _CharT, typename _Traits> void  
std::basic_ifstream<_CharT, _Traits>::open ( const char * __s,  
ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 526 of file `fstream`.

**5.364.5.35** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.364.5.36** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.364.5.37** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type  
&(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `istream` header.

Definition at line 124 of file `istream`.

**5.364.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.364.5.39** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

**5.364.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

## 5.364 std::basic\_istream<\_CharT, \_Traits > Class Template Reference 1579

---

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

```
5.364.5.41 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( float & __f
) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

```
5.364.5.42 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( double &
__f ) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

```
5.364.5.43 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( bool & __n
) [inline, inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

**5.364.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

**5.364.5.45** `template<typename _CharT , typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

## 5.364 `std::basic_ifstream<_CharT, _Traits >` Class Template Reference 1581

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

```
5.364.5.46 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( void *& __p
) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

```
5.364.5.47 template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream<_CharT, _Traits
>::operator>> ( __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.364.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

**5.364.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

**5.364.5.50** `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.364.5.51** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream< _CharT, _Traits >::operator>> ( unsigned int & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

**5.364.5.52** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream< _CharT, _Traits >::operator>> ( __istream_type &(&)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

```
5.364.5.53 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> ( long & __n
) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

```
5.364.5.54 template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream<_CharT, _Traits
>::peek ( void ) [inherited]
```

Looking ahead in the stream.

#### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

```
5.364.5.55 streamsize std::ios_base::precision ( streamsize __prec )
[inline, inherited]
```

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.364.5.56** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.364.5.57** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback ( char_type __c ) [inherited]`

Unextracting a single character.

**Parameters**

*c* The character to push back into the input stream.

**Returns**

`*this`

If `rdbuf ()` is not null, calls `rdbuf ()->sputbackc (c)`.

If `rdbuf ()` is null or if `sputbackc ()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount ()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

#### 5.364.5.58 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

##### Parameters

`__ix` Index into the array.

##### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

#### 5.364.5.59 `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_ifstream<_CharT, _Traits>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

##### Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 497 of file `fstream`.

**5.364.5.60** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.364.5.61** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline, inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.364.5.62** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read ( char_type * __s, streamsize __n ) [inherited]`

Extraction without delimiters.

**Parameters**

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.364.5.63** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

**Parameters**

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets `eofbit` and extracts no characters
- if *A* == 0, extracts no characters

## 5.364 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1589

- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.364.5.64 `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

### 5.364.5.65 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

`pos` A file position object.

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets `failbit`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.364.5.66** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.364.5.67** `fmtflags std::ios_base::setf( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

## 5.364 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1591

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.364.5.68** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )`  
[`inline`, `inherited`]

Setting new format flags.

### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.364.5.69** `template<typename _CharT, typename _Traits> void`  
`std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )`  
[`inline`, `inherited`]

Sets additional flags in the error state.

### Parameters

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`,

`std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.364.5.70** `template<typename _CharT, typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.364.5.71** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

## 5.364 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1593

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.364.5.72** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg( void ) [inherited]`

Getting the current read position.

### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.364.5.73** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.364.5.74** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( basic_ostream<_CharT, _Traits > * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

**5.364.5.75** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::unget  
( void ) [inherited]`

Unextracting the previous character.

#### Returns

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sungetc()`.

**5.364.5.76** `void std::ios_base::unsetf ( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.364.5.77** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> (getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.364.5.78** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.364.5.79** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of [width\(\)](#).

Definition at line 652 of file `ios_base.h`.

**5.364.5.80** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.364.6 Member Data Documentation

**5.364.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.364.6.2 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutputIter >::do_put()`.

**5.364.6.3 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.364.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.364.6.5 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such `///` as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.364.6.6 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.364.6.7 `const seekdir std::ios_base::beg` [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

#### 5.364.6.8 `const openmode std::ios_base::binary` [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### 5.364.6.9 `const fmtflags std::ios_base::boolalpha` [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.364.6.10 `const seekdir std::ios_base::cur` [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### 5.364.6.11 `const fmtflags std::ios_base::dec` [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

#### 5.364.6.12 `const seekdir std::ios_base::end` [static, inherited]

Request a seek relative to the current end of the sequence.

## 5.364 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference 1599

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

### 5.364.6.13 const iostate std::ios\_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

### 5.364.6.14 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

### 5.364.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

### 5.364.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.364.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.364.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.364.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::underflow()`.

**5.364.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

**5.364.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

**5.364.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.364.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::seekp()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

**5.364.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file `ios_base.h`.

**5.364.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.364.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

**5.364.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.364.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.364.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.364.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.364.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.364.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

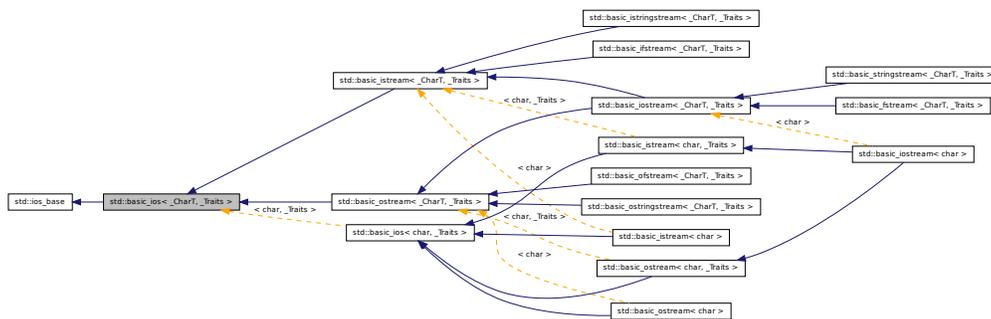
- [fstream](#)

## 5.365 std::basic\_ios< \_CharT, \_Traits > Class Template Reference

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Inheritance diagram for `std::basic_ios< _CharT, _Traits >`:



### Public Types

- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
  
- typedef `_CharT` `char_type`
- typedef `_Traits::int_type` `int_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_Traits::off_type` `off_type`
- typedef `_Traits` `traits_type`
  
- typedef `ctype< _CharT >` `__ctype_type`

- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > _-`  
`num_put_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > _-`  
`num_get_type`

## Public Member Functions

- `basic_ios (basic_streambuf< _CharT, _Traits > *__sb)`
- virtual `~basic_ios ()`
- const `locale & _M_getloc () const`
- void `_M_setstate (iostate __state)`
- bool `bad () const`
- void `clear (iostate __state=goodbit)`
- `basic_ios & copyfmt (const basic_ios &__rhs)`
- bool `eof () const`
- `iostate exceptions () const`
- void `exceptions (iostate __except)`
- bool `fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `locale getloc () const`
- bool `good () const`
- `locale imbue (const locale &__loc)`
- long & `isword (int __ix)`
- char `narrow (char_type __c, char __dfault) const`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- void \*& `pword (int __ix)`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
- `basic_streambuf< _CharT, _Traits > * rdbuf () const`
- `iostate rdstate () const`
- void `register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- void `setstate (iostate __state)`
- `basic_ostream< _CharT, _Traits > * tie () const`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__-`  
`__tiestr)`
- void `unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`

- `streamsize width () const`
- `streamsize width (streamsize __wide)`
  - `operator void * () const`
  - `bool operator! () const`

### Static Public Member Functions

- static `bool sync_with_stdio (bool __sync=true)`
- static `int xalloc () throw ()`

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosate eofbit`
- static const `iosate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- `basic_ios` ()
- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

### 5.365.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ios< _CharT,
_Traits >
```

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 62 of file `basic_ios.h`.

## 5.365.2 Member Typedef Documentation

### 5.365.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type`

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<char, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 82 of file `basic_ios.h`.

### 5.365.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_get_type`

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<char, \\_Traits>](#), and [std::basic\\_istream<char>](#).

Definition at line 86 of file `basic_ios.h`.

### 5.365.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type`

These are non-standard types.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

### 5.365.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ios<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 71 of file `basic_ios.h`.

#### 5.365.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` **[inherited]**

The type of an event callback function.

##### Parameters

- event* One of the members of the event enum.
- ios\_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

#### 5.365.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`

- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

#### 5.365.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits >`, `std::basic_ofstream<_CharT, _Traits >`, `std::basic_fstream<_CharT, _Traits >`, `std::basic_istream<_CharT, _Traits >`, `std::basic_iostream<_CharT, _Traits >`, `std::basic_ostream<_CharT, _Traits >`, `std::basic_istreamstream<_CharT, _Traits, _Alloc >`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc >`, `std::basic_stringstream<_CharT, _Traits, _Alloc >`, `std::basic_istream<char, _Traits >`, `std::basic_istream<char >`, `std::basic_iostream<char >`, `std::basic_ostream<char, _Traits >`, and `std::basic_ostream<char >`.

Definition at line 72 of file `basic_ios.h`.

#### 5.365.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 338 of file ios\_base.h.

#### 5.365.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 74 of file basic\_ios.h.

#### 5.365.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 369 of file ios\_base.h.

**5.365.2.11** `template<typename _CharT, typename _Traits> typedef`  
`_Traits::pos_type std::basic_ios<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 73 of file `basic_ios.h`.

**5.365.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir` `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

**5.365.2.13** `template<typename _CharT, typename _Traits> typedef`  
`_Traits`  
`std::basic_ios<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 75 of file basic\_ios.h.

### 5.365.3 Member Enumeration Documentation

#### 5.365.3.1 enum `std::ios_base::event` `[inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file ios\_base.h.

### 5.365.4 Constructor & Destructor Documentation

#### 5.365.4.1 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios ( basic_streambuf<_CharT, _Traits> * __sb ) [inline, explicit]`

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 260 of file basic\_ios.h.

#### 5.365.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_ios<_CharT, _Traits>::~~basic_ios ( ) [inline, virtual]`

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by `rdbuf()`.

Definition at line 272 of file basic\_ios.h.

#### 5.365.4.3 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios ( ) [inline, protected]`

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 450 of file basic\_ios.h.

## 5.365.5 Member Function Documentation

### 5.365.5.1 `const locale& std::ios_base::_M_getloc ( ) const` [`inline`, `inherited`]

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

### 5.365.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const` [`inline`]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

### 5.365.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.365.5.4** `template<typename _CharT, typename _Traits > basic_ios<_CharT, _Traits > & std::basic_ios<_CharT, _Traits >::copyfmt ( const basic_ios<_CharT, _Traits > & __rhs )`

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits >::exceptions()`, `std::basic_ios<_CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.365.5.5** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::eof( ) const [inline]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_istream<_CharT, _Traits >::putback()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.365.5.6** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions( ) const [inline]`

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

#### 5.365.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline]`

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file basic\_ios.h.

#### 5.365.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const [inline]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.365.5.9** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.365.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline]`

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.365.5.11 fmtflags std::ios\_base::flags ( ) const [inline, inherited]**

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 550 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_put<\_CharT, \_OutIter>::do\_put(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator<<(), and std::operator>>().

**5.365.5.12 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline, inherited]**

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

**5.365.5.13 locale std::ios\_base::getloc ( ) const [inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 694 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::money\_put<\_CharT, \_OutIter>::do\_put(), std::basic\_ios<\_CharT, \_Traits>::imbue(), std::operator>>(), and std::ws().

**5.365.5.14** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::good ( ) const [inline]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 170 of file basic\_ios.h.

**5.365.5.15** `template<typename _CharT , typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file basic\_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.365.5.16** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic\_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.365.5.17** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.365.5.18** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> (getloc()) .narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.365.5.19** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

**5.365.5.20** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

**5.365.5.21** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.365.5.22** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

#### Parameters

*prec* The new precision value.

#### Returns

The previous value of `precision()`.

Definition at line 629 of file ios\_base.h.

#### 5.365.5.23 void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]

Access to void pointer array.

##### Parameters

*\_\_ix* Index into the array.

##### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios\_base.h.

#### 5.365.5.24 template<typename \_CharT, typename \_Traits> basic\_streambuf<\_CharT, \_Traits> \* std::basic\_ios<\_CharT, \_Traits>::rdbuf ( basic\_streambuf<\_CharT, \_Traits> \* \_\_sb )

Changing the underlying buffer.

##### Parameters

*sb* The new stream buffer.

##### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits >::clear()`.

**5.365.5.25** `template<typename _CharT, typename _Traits>`  
`basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,`  
`_Traits >::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits >`, `std::basic_ofstream<_CharT, _Traits >`, `std::basic_fstream<_CharT, _Traits >`, `std::basic_istringstream<_CharT, _Traits, _Alloc >`, `std::basic_ostringstream<_CharT, _Traits, _Alloc >`, and `std::basic_stringstream<_CharT, _Traits, _Alloc >`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::imbue()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, `std::basic_istream<_CharT, _Traits >::unget()`, and `std::ws()`.

**5.365.5.26** `template<typename _CharT, typename _Traits> iostate`  
`std::basic_ios<_CharT, _Traits >::rdstate ( ) const [inline]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.365.5.27** void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]

Add the callback \_\_fn with parameter \_\_index.

**Parameters**

*\_\_fn* The function to add.

*\_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.365.5.28** fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl ) [inline, inherited]

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios\_base.h.

Referenced by std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::internal(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**5.365.5.29** fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl, fmtflags \_\_mask ) [inline, inherited]

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.365.5.30** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline]`

Sets additional flags in the error state.

**Parameters**

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.365.5.31** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

```
5.365.5.32 template<typename _CharT, typename _Traits>
    basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>
    >::tie( ) const [inline]
```

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

```
5.365.5.33 template<typename _CharT, typename _Traits>
    basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>
    >::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline]
```

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic\_ios.h.

```
5.365.5.34 void std::ios_base::unsetf( fmtflags __mask ) [inline,
    inherited]
```

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.365.5.35** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits >::widen ( char __c ) const  
[inline]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic\_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

**5.365.5.36** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits >::copyfmt(), std::num\_put<\_CharT, \_OutIter >::do\_put(), and std::operator>>().

**5.365.5.37** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.365.5.38** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.365.6 Member Data Documentation

**5.365.6.1** `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.365.6.2** `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.365.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.365.6.4 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.365.6.5 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.365.6.6 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

**5.365.6.7 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 5.365.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.365.6.9 const seekdir std::ios\_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 5.365.6.10 const fmtflags std::ios\_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### 5.365.6.11 const seekdir std::ios\_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

#### 5.365.6.12 const iostate std::ios\_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter

>::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

#### 5.365.6.13 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

#### 5.365.6.14 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

#### 5.365.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of *setf*.

Definition at line 324 of file ios\_base.h.

#### 5.365.6.16 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),

`std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### 5.365.6.17 `const fmtflags std::ios_base::hex` [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

#### 5.365.6.18 `const openmode std::ios_base::in` [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

#### 5.365.6.19 `const fmtflags std::ios_base::internal` [static, inherited]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

#### 5.365.6.20 `const fmtflags std::ios_base::left` [static, inherited]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.365.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.365.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for ofstream and fstream.

Definition at line 386 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::seekp(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.365.6.23 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file ios\_base.h.

**5.365.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.365.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios\_base.h.

**5.365.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.365.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

## **5.366 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1633**

Definition at line 305 of file `ios_base.h`.

### **5.365.6.28 `const fmtflags std::ios_base::skipws` [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

### **5.365.6.29 `const openmode std::ios_base::trunc` [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

### **5.365.6.30 `const fmtflags std::ios_base::unitbuf` [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

### **5.365.6.31 `const fmtflags std::ios_base::uppercase` [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

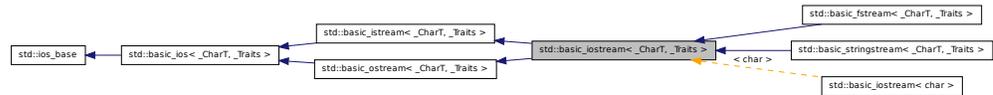
- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## **5.366 `std::basic_iostream<_CharT, _Traits>` Class Template Reference**

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Inheritance diagram for `std::basic_istream<_CharT, _Traits >`:



## Public Types

- typedef `ctype<_CharT > __ctype_type`
- typedef `ctype<_CharT > __ctype_type`
- typedef `basic_ios<_CharT, _Traits > __ios_type`
- typedef `basic_ios<_CharT, _Traits > __ios_type`
- typedef `basic_istream<_CharT, _Traits > __istream_type`
- typedef `basic_istream<_CharT, _Traits > __istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > __num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __num_put_type`
- typedef `basic_ostream<_CharT, _Traits > __ostream_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `_Traits::int_type int_type`
- typedef `_Traits::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `_Traits::off_type off_type`
- typedef `_Traits::off_type off_type`
- typedef `int open_mode`
- typedef `_Ios_Openmode openmode`
- typedef `_Traits::pos_type pos_type`
- typedef `_Traits::pos_type pos_type`
- typedef `int seek_dir`
- typedef `_Ios_Seekdir seekdir`
- typedef `std::streamoff streamoff`

- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`
- typedef `_Traits` `traits_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits >>` `__num_put_type`

## Public Member Functions

- `basic_istream` (`basic_streambuf<_CharT, _Traits > *__sb`)
- virtual `~basic_istream` ()
- const `locale &_M_getloc` () const
- void `_M_setstate` (`iosstate __state`)
- bool `bad` () const
- void `clear` (`iosstate __state=goodbit`)
- `basic_ios &copyfmt` (const `basic_ios &__rhs`)
- bool `eof` () const
- `iosstate exceptions` () const
- void `exceptions` (`iosstate __except`)
- bool `fail` () const
- `char_type fill` () const
- `char_type fill` (`char_type __ch`)
- `fmtflags flags` (`fmtflags __fmtfl`)
- `fmtflags flags` () const
- `__ostream_type &flush` ()
- `streamsize gcount` () const
- template<>  
`basic_istream< char > &getline` (`char_type *__s`, `streamsize __n`, `char_type __delim`)
- template<>  
`basic_istream< wchar_t > &getline` (`char_type *__s`, `streamsize __n`, `char_type __delim`)
- `locale getloc` () const
- bool `good` () const
- template<>  
`basic_istream< wchar_t > &ignore` (`streamsize __n`)
- template<>  
`basic_istream< wchar_t > &ignore` (`streamsize __n`, `int_type __delim`)
- template<>  
`basic_istream< char > &ignore` (`streamsize __n`)
- template<>  
`basic_istream< char > &ignore` (`streamsize __n`, `int_type __delim`)
- `locale imbue` (const `locale &__loc`)

- long & `word` (int \_\_ix)
  - char `narrow` (char\_type \_\_c, char \_\_dfault) const
  - `streamsize precision` () const
  - `streamsize precision` (streamsize \_\_prec)
  - void \*& `pword` (int \_\_ix)
  - `basic_streambuf`< \_CharT, \_Traits > \* `rdbuf` () const
  - `basic_streambuf`< \_CharT, \_Traits > \* `rdbuf` (`basic_streambuf`< \_CharT, \_Traits > \* \_\_sb)
  - `iostate rdstate` () const
  - void `register_callback` (event\_callback \_\_fn, int \_\_index)
  - `__ostream_type` & `seekp` (pos\_type)
  - `__ostream_type` & `seekp` (off\_type, ios\_base::seekdir)
  - `fmtflags setf` (fmtflags \_\_fmtfl)
  - `fmtflags setf` (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - void `setstate` (iostate \_\_state)
  - `pos_type tellp` ()
  - `basic_ostream`< \_CharT, \_Traits > \* `tie` (`basic_ostream`< \_CharT, \_Traits > \* \_\_tistr)
  - `basic_ostream`< \_CharT, \_Traits > \* `tie` () const
  - void `unsetf` (fmtflags \_\_mask)
  - `char_type widen` (char \_\_c) const
  - `streamsize width` (streamsize \_\_wide)
  - `streamsize width` () const
- 
- `__istream_type` & `operator>>` (`__istream_type` &(\*\_\_pf)(`__istream_type` &))
  - `__istream_type` & `operator>>` (`__ios_type` &(\*\_\_pf)(`__ios_type` &))
  - `__istream_type` & `operator>>` (`ios_base` &(\*\_\_pf)(`ios_base` &))

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type` & `operator>>` (bool & \_\_n)
- `__istream_type` & `operator>>` (short & \_\_n)

- `__istream_type & operator>>` (unsigned short &\_\_n)
- `__istream_type & operator>>` (int &\_\_n)
- `__istream_type & operator>>` (unsigned int &\_\_n)
- `__istream_type & operator>>` (long &\_\_n)
- `__istream_type & operator>>` (unsigned long &\_\_n)
- `__istream_type & operator>>` (long long &\_\_n)
- `__istream_type & operator>>` (unsigned long long &\_\_n)
- `__istream_type & operator>>` (float &\_\_f)
- `__istream_type & operator>>` (double &\_\_f)
- `__istream_type & operator>>` (long double &\_\_f)
- `__istream_type & operator>>` (void \*&\_\_p)
- `__istream_type & operator>>` (`__streambuf_type` \*\_\_sb)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
  - `__istream_type & get (char_type &__c)`
  - `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & get (char_type *__s, streamsize __n)`
  - `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
  - `__istream_type & get (__streambuf_type &__sb)`
  - `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & getline (char_type *__s, streamsize __n)`
  - `__istream_type & ignore ()`
  - `__istream_type & ignore (streamsize __n)`
  - `__istream_type & ignore (streamsize __n, int_type __delim)`
  - `int_type peek ()`
  - `__istream_type & read (char_type *__s, streamsize __n)`
  - `streamsize readsome (char_type *__s, streamsize __n)`
  - `__istream_type & putback (char_type __c)`
  - `__istream_type & unget ()`
  - `int sync ()`
  - `pos_type tellg ()`
  - `__istream_type & seekg (pos_type)`
  - `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`

- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type * __sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type * __s, streamsize __n)`
- `__ostream_type & write (const char_type * __s, streamsize __n)`

## Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync=true`)
- static int `xalloc` () throw ()

## Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
[\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
[\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## Friends

- class **sentry**
- class **sentry**

### 5.366.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _CharT, _Traits >
```

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 769 of file istream.

### 5.366.2 Member Typedef Documentation

```
5.366.2.1 template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_istream< _CharT, _Traits  
>::_ctype_type [inherited]
```

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file istream.

```
5.366.2.2 template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_ostream< _CharT, _Traits  
>::_ctype_type [inherited]
```

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file ostream.

```
5.366.2.3 template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_istream< _CharT, _Traits >::_num_get_type  
[inherited]
```

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file istream.

**5.366.2.4** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits >::__num_put_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.366.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits >::__num_put_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits >](#), [std::basic\\_ostream<char, \\_Traits >](#), and [std::basic\\_ostream<char >](#).

Definition at line 84 of file basic\_ios.h.

**5.366.2.6** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_istream<_CharT, _Traits >::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits >](#), and [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 59 of file istream.

**5.366.2.7** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_iostream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream<\\_CharT, \\_Traits >](#), and [std::basic\\_istreamstream<\\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 777 of file istream.

**5.366.2.8 typedef void(\* std::ios\_base::event\_callback)(event, ios\_base &, int) [inherited]**

The type of an event callback function.

**Parameters**

*event* One of the members of the event enum.

*ios\_base* Reference to the *ios\_base* object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several *ios\_base* and *basic\_ios* functions, specifically *imbue()*, *copyfmt()*, and *~ios()*.

Definition at line 444 of file ios\_base.h.

**5.366.2.9 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags [inherited]**

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos

- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios\_base.h.

**5.366.2.10** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream< \\_CharT, \\_Traits >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 778 of file istream.

**5.366.2.11** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 60 of file istream.

**5.366.2.12** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

#### 5.366.2.13 `template<typename _CharT, typename _Traits> typedef` `_Traits::off_type std::basic_iostream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Reimplemented in `std::basic_fstream<_CharT, _Traits>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 780 of file `istream`.

#### 5.366.2.14 `template<typename _CharT, typename _Traits> typedef` `_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type` `[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istringstream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

#### 5.366.2.15 `typedef _Ios_Openmode std::ios_base::openmode` `[inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`

- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

**5.366.2.16** `template<typename _CharT, typename _Traits> typedef  
_Traits::pos_type std::basic_istream< _CharT, _Traits  
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 779 of file `istream`.

**5.366.2.17** `template<typename _CharT, typename _Traits> typedef  
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file `istream`.

**5.366.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`

- cur, equivalent to SEEK\_CUR in the C standard library.
- end, equivalent to SEEK\_END in the C standard library.

Definition at line 401 of file ios\_base.h.

**5.366.2.19** `template<typename _CharT, typename _Traits> typedef  
_Traits std::basic_istream< _CharT, _Traits >::traits_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 63 of file istream.

**5.366.2.20** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_iostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream< \\_CharT, \\_Traits >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 781 of file istream.

### **5.366.3 Member Enumeration Documentation**

**5.366.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 427 of file ios\_base.h.

## 5.366.4 Constructor & Destructor Documentation

**5.366.4.1** `template<typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits >::basic_istream (   
basic_streambuf< _CharT, _Traits > * __sb ) [inline,  
explicit]`

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 794 of file istream.

**5.366.4.2** `template<typename _CharT, typename _Traits> virtual  
std::basic_istream< _CharT, _Traits >::~~basic_istream ( )  
[inline, virtual]`

Destructor does nothing.

Definition at line 801 of file istream.

## 5.366.5 Member Function Documentation

**5.366.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline,  
inherited]`

Locale access.

### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

**5.366.5.2** `template<typename _CharT, typename _Traits> void  
std::basic_ostream<_CharT, _Traits>::_M_write ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

**5.366.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

**5.366.5.4** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.366.5.5** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt ( const basic_ios<_CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.366.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.366.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

**5.366.5.8** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const [inline, inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.366.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.366.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( char_type __ch ) [inline, inherited]`

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.366.5.11** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( ) const [inline, inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

## 5.366 std::basic\_ostream<\_CharT, \_Traits> Class Template Reference 1653

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt().

### 5.366.5.12 fmtflags std::ios\_base::flags( ) const [inline, inherited]

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 550 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_put<\_CharT, \_OutIter>::do\_put(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator<<(), and std::operator>>().

### 5.366.5.13 fmtflags std::ios\_base::flags( fmtflags \_\_fmtfl ) [inline, inherited]

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

### 5.366.5.14 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::flush( ) [inherited]

Synchronizing the stream buffer.

#### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**5.366.5.15** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::gcount ( ) const  
[inline, inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

**5.366.5.16** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits  
>::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.366.5.17** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

#### Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**5.366.5.18** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type & __c ) [inherited]`

Simple extraction.

**Parameters**

*c* The character in which to store data.

**Returns**

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits >::rdbuf(), and std::basic\_ios<\_CharT, \_Traits >::setstate().

**5.366.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

**Parameters**

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns get(s,n,widen('\n')).

Definition at line 333 of file istream.

**5.366.5.20** `template<typename _CharT , typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

*delim* A "stop" character.

### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), std::basic\_streambuf< \_CharT, \_Traits >::snextc(), and std::basic\_streambuf< \_CharT, \_Traits >::sputc().

**5.366.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

### Parameters

*sb* A streambuf in which to store data.

### Returns

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

**5.366.5.22** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim )` **[inherited]**

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.366.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n )` **[inline, inherited]**

String extraction.

**Parameters**

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.366.5.24 locale std::ios\_base::getloc ( ) const [inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutputIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.366.5.25 template<typename \_CharT, typename \_Traits> bool std::basic\_ios<\_CharT, \_Traits>::good ( ) const [inline, inherited]**

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.366.5.26** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::ignore ( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_streambuf<_CharT, _Traits >::sbumpc()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.366.5.27** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

## 5.366 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1661

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

### 5.366.5.28 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

### 5.366.5.29 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios\\_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.366.5.30** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.366.5.31** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.366.5.32** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

**Parameters**

- c* The character to narrow.  
*dfault* The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>>(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.366.5.33** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.366.5.34** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.366.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n )` [`inline`, `inherited`]

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

**5.366.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

**5.366.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

**5.366.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**5.366.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ostream_type &(*)(__ostream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.366.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.366.5.41** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.366.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 180 of file ostream.

**5.366.5.43** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< ( int __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.366.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

**5.366.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

**5.366.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( long double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

**5.366.5.47** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( float __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

**5.366.5.48** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

**5.366.5.49** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<<( _streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.366.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

**5.366.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.366.5.52** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file istream.

**5.366.5.53** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

## 5.366 std::basic\_istream<\_CharT, \_Traits> Class Template Reference 1671

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

```
5.366.5.54 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int
& __n ) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

```
5.366.5.55 template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits
>::operator>> ( __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

**5.366.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

**5.366.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

## 5.366 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1673

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

```
5.366.5.59 template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n  
) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

```
5.366.5.60 template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long & __n ) [inline, inherited]
```

Basic arithmetic extractors.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.366.5.61** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>>( short & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.366.5.62** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>>( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

**5.366.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>>( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file istream.

**5.366.5.64** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( double & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file istream.

**5.366.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

**5.366.5.66** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

**5.366.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

## 5.366 std::basic\_istream<\_CharT, \_Traits> Class Template Reference 1677

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file `istream`.

**5.366.5.69** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void ) [inherited]`

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.70** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.366.5.71** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.366.5.72** `template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::put ( char_type __c ) [inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits >::rdbuf(), and std::basic\_ios<\_CharT, \_Traits >::setstate().

Referenced by std::endl(), and std::ends().

**5.366.5.73** `template<typename _CharT , typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::putback ( char_type __c ) [inherited]`

Unextracting a single character.

#### Parameters

*c* The character to push back into the input stream.

#### Returns

\*this

If `rdbuf ()` is not null, calls `rdbuf ()->sputbackc (c)`.

If `rdbuf ()` is null or if `sputbackc ()` fails, sets badbit in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**5.366.5.74 void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]**

Access to void pointer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

**5.366.5.75 template<typename \_CharT, typename \_Traits> basic\_streambuf<\_CharT, \_Traits>\* std::basic\_ios<\_CharT, \_Traits>::rdbuf ( ) const [inline, inherited]**

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.366.5.76** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.366.5.77** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.366.5.78** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(  
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

**Parameters**

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.79** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.80** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

- `__fn` The function to add.
- `__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.366.5.81** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.366.5.82** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.84** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

## 5.366 std::basic\_ostream<\_CharT, \_Traits> Class Template Reference 1685

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.366.5.85 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

### 5.366.5.86 `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.366.5.87** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.366.5.88** `template<typename _CharT , typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf ()` is a null pointer, returns -1.

Otherwise, calls `rdbuf () ->pubsync ()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount ()`.

Definition at line 777 of file `istream.tcc`.

## 5.366 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1687

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.366.5.89** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
[static, inherited]

Interaction with the standard C I/O objects.

### Parameters

*sync* Whether to synchronize or not.

### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.366.5.90** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg ( void )` [inherited]

Getting the current read position.

### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.366.5.91** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.366.5.92** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( basic_ostream<_CharT, _Traits>* __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

**5.366.5.93** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

## 5.366 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1689

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.366.5.94 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ungetc ( void ) [inherited]`

Unextracting the previous character.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

### 5.366.5.95 `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.366.5.96** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.366.5.97** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.366.5.98** `streamsize std::ios_base::width ( streamsize __wide ) [inline,  
inherited]`

Changing flags.

#### Parameters

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.366.5.99** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

**Parameters**

*s* The array to insert.

*n* Maximum number of characters to insert.

**Returns**

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**5.366.5.100** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.366.6 Member Data Documentation

#### 5.366.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount` [**protected, inherited**]

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.366.6.2 `const fmtflags std::ios_base::adjustfield` [**static, inherited**]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outputter >::do_put()`.

#### 5.366.6.3 `const openmode std::ios_base::app` [**static, inherited**]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

#### 5.366.6.4 `const openmode std::ios_base::ate` [**static, inherited**]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### 5.366.6.5 `const iostate std::ios_base::badbit` [**static, inherited**]

Indicates a loss of integrity in an input or output sequence (such `///` as an irrecoverable read error from a file).

## 5.366 std::basic\_iostream<\_CharT, \_Traits> Class Template Reference 1693

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ostream<\_CharT, \_Traits>::flush(), std::basic\_istream<\_CharT, \_Traits>::get(), std::basic\_istream<\_CharT, \_Traits>::getline(), std::basic\_istream<\_CharT, \_Traits>::ignore(), std::basic\_ios<\_CharT, \_Traits>::init(), std::operator<<(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator>>(), std::basic\_istream<\_CharT, \_Traits>::operator>>(), std::basic\_istream<\_CharT, \_Traits>::peek(), std::basic\_ostream<\_CharT, \_Traits>::put(), std::basic\_istream<\_CharT, \_Traits>::putback(), std::basic\_istream<\_CharT, \_Traits>::read(), std::basic\_istream<\_CharT, \_Traits>::readsome(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::sync(), std::basic\_istream<\_CharT, \_Traits>::tellg(), std::basic\_ostream<\_CharT, \_Traits>::tellp(), and std::basic\_istream<\_CharT, \_Traits>::unget().

### 5.366.6.6 const fmtflags std::ios\_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 321 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, InIter>::do\_get(), and std::basic\_ostream<\_CharT, \_Traits>::operator<<().

### 5.366.6.7 const seekdir std::ios\_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

### 5.366.6.8 const openmode std::ios\_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf<\_CharT, \_Traits>::showmanyc().

### 5.366.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### **5.366.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### **5.366.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

#### **5.366.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### **5.366.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

#### **5.366.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

## 5.366 std::basic\_iostream<\_CharT, \_Traits> Class Template Reference 1695

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::basic\_istream<\_CharT, \_Traits>::get(), std::basic\_istream<\_CharT, \_Traits>::getline(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::basic\_istream<\_CharT, \_Traits>::operator>>(), std::operator>>(), std::basic\_istream<\_CharT, \_Traits>::read(), std::basic\_istream<\_CharT, \_Traits>::seekg(), and std::basic\_ostream<\_CharT, \_Traits>::seekp().

### 5.366.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

### 5.366.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

### 5.366.6.17 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::basic\_ostream<\_CharT, \_Traits>::flush(), std::basic\_istream<\_CharT, \_Traits>::get(), std::basic\_istream<\_CharT, \_Traits>::getline(), std::basic\_istream<\_CharT, \_Traits>::ignore(), std::basic\_ios<\_CharT, \_Traits>::init(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator>>(), std::basic\_istream<\_CharT, \_Traits>::operator>>(), std::basic\_istream<\_CharT, \_Traits>::peek(), std::basic\_ostream<\_CharT, \_Traits>::put(), std::basic\_istream<\_CharT, \_Traits>::putback(), std::basic\_istream<\_CharT, \_Traits>::read(), std::basic\_istream<\_CharT, \_Traits>::readsome(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::sync(), and std::basic\_istream<\_CharT, \_Traits>::unget().

### 5.366.6.18 const fmtflags std::ios\_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### **5.366.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::underflow()`.

#### **5.366.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

#### **5.366.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

#### **5.366.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### **5.366.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

## **5.366 std::basic\_ostream<\_CharT, \_Traits> Class Template Reference 1697**

Referenced by `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

### **5.366.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

### **5.366.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

### **5.366.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

### **5.366.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 302 of file `ios_base.h`.

### **5.366.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a `+` sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

### **5.366.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

### **5.366.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

### 5.366.6.31 `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

### 5.366.6.32 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

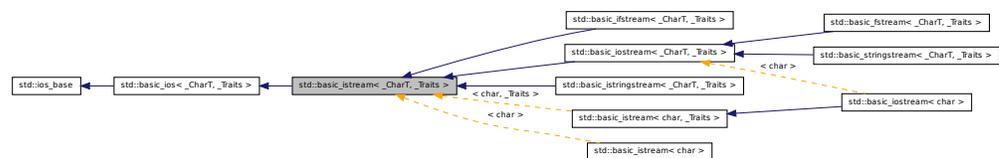
- [istream](#)

## 5.367 `std::basic_istream< _CharT, _Traits >` Class Template Reference

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.

Inheritance diagram for `std::basic_istream< _CharT, _Traits >`:



## Classes

- class [sentry](#)

*Performs setup work for input streams.*

## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

## Public Member Functions

- `basic_istream` (`__streambuf_type * __sb`)
- virtual `~basic_istream` ()
- const `locale & _M_getloc` () const
- void `_M_setstate` (`iostate __state`)
- bool `bad` () const
- void `clear` (`iostate __state=goodbit`)
- `basic_ios & copyfmt` (const `basic_ios & __rhs`)
- bool `eof` () const
- `iostate exceptions` () const

- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- template<>  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- [streamsize](#) [precision](#) () const
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- void [setstate](#) ([iostate](#) \_\_state)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_tiestr)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [char\\_type](#) [widen](#) ([char](#) \_\_c) const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)

- `streamsize width () const`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type * __sb)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
  
- `operator void * () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`

- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  `__istream_type` & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`

- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- [const](#) **\_\_num\_get\_type** \* **\_M\_num\_get**
- [const](#) **\_\_num\_put\_type** \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- [int](#) **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## Friends

- class **sentry**

### 5.367.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _-
CharT, _Traits >
```

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.

Definition at line 55 of file istream.

### 5.367.2 Member Typedef Documentation

```
5.367.2.1 template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_istream< _CharT, _Traits
>::__ctype_type
```

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 71 of file istream.

**5.367.2.2** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_istream<_CharT, _Traits>::_num_get_type`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `istream`.

**5.367.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::_num_put_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

**5.367.2.4** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_istream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), and [std::basic\\_istreamringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 59 of file `istream`.

**5.367.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file [ios\\_base.h](#).

#### 5.367.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file [ios\\_base.h](#).

---

### 5.367.2.7 `template<typename _CharT, typename _Traits> typedef` `_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 60 of file `istream`.

### 5.367.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

### 5.367.2.9 `template<typename _CharT, typename _Traits> typedef` `_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

### 5.367.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

#### **5.367.2.11** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file `istream`.

#### **5.367.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

### 5.367.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 63 of file `istream`.

## 5.367.3 Member Enumeration Documentation

### 5.367.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

## 5.367.4 Constructor & Destructor Documentation

### 5.367.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits>::basic_istream ( __streambuf_type * __sb ) [inline, explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 91 of file `istream`.

### 5.367.4.2 `template<typename _CharT, typename _Traits> virtual std::basic_istream<_CharT, _Traits>::~~basic_istream ( ) [inline, virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 101 of file `istream`.

## 5.367.5 Member Function Documentation

### 5.367.5.1 `const locale& std::ios_base::_M_getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter >::do_get()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_date()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_time()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::time_put<_CharT, _OutIter >::do_put()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::time_put<_CharT, _OutIter >::put()`.

### 5.367.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

### 5.367.5.3 `template<typename _CharT, typename _Traits > void std::basic_ios<_CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

## 5.367 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1711

---

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.367.5.4** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt ( const basic_ios<_CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

### Parameters

`__rhs` The source values for the copies.

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.367.5.5** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.367.5.6** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

**5.367.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions ( ) const [inline, inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.367.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**5.367.5.9** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill( char_type __ch ) [inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.367.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill( ) const [inline, inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

#### 5.367.5.11 `fmtflags std::ios_base::flags( ) const [inline, inherited]`

Access to format flags.

##### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

#### 5.367.5.12 `fmtflags std::ios_base::flags( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags all at once.

##### Parameters

*fmtfl* The new flags to set.

##### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

#### 5.367.5.13 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount( ) const [inline]`

Character counting.

##### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

**5.367.5.14** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get ( void )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.367.5.15** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type & __c )`

Simple extraction.

#### Parameters

*c* The character in which to store data.

#### Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.367.5.16** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim )`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.367.5.17** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline]`

Simple multiple-character extraction.

**Parameters**

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

### 5.367.5.18 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim )`

Extraction into another streambuf.

**Parameters**

- sb* A streambuf in which to store data.
- delim* A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, `failbit` is set in the stream's error state.

Definition at line 356 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

**5.367.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

**5.367.5.20** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim )`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

*delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

#### 5.367.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline]`

String extraction.

##### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

##### Returns

`*this`

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

#### 5.367.5.22 `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.367.5.23** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::good ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.367.5.24** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore  
( void )`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.367.5.25** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.367.5.26** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.367.5.27** `template<typename _CharT , typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.367.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.367.5.29** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.367.5.30** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

*c* The character to narrow.

*dfault* The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> (getloc()).narrow(c, dfault)
```

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.367.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.367.5.32** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.367.5.33** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned int & n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

**5.367.5.34** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> ( __streambuf_type * sb )`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.367.5.35** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( short & __n )`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**5.367.5.37** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

**5.367.5.38** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file istream.

**5.367.5.39** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __ios_type &(__ios_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

**5.367.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( bool & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

**5.367.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.367.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**5.367.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

**5.367.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned short & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file `istream`.

**5.367.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( void *& __p ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

**5.367.5.46** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( float & __f ) [inline]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

**5.367.5.47** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( double & __f ) [inline]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

**5.367.5.48** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( int & __n )`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & _f ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

**5.367.5.50** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void )`

Looking ahead in the stream.

#### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.51** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.367.5.52** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.367.5.53** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback ( char_type __c )`

Unextracting a single character.

**Parameters**

*c* The character to push back into the input stream.

**Returns**

\*this

If `rdbuf ()` is not null, calls `rdbuf ()->sputbackc (c)`.

If `rdbuf ()` is null or if `sputbackc ()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount ()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

### 5.367.5.54 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

### 5.367.5.55 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf ( ) const [inline, inherited]`

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

#### 5.367.5.56 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

##### Parameters

*sb* The new stream buffer.

##### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.367.5.57** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

**5.367.5.58** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (   
char_type * __s, streamsize __n )`

Extraction without delimiters.

#### Parameters

*s* A character array.

*n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References [std::basic\\_istream< \\_CharT, \\_Traits >::\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**5.367.5.59** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n )`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() ->in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.60** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

- `__fn` The function to add.
- `__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.367.5.61** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( pos_type __pos )`

Changing the current read position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.367.5.62** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( off_type __off, ios_base::seekdir __dir )`

Changing the current read position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.63 `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`**

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.367.5.64 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline, inherited]`**

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.367.5.65** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.367.5.66** `template<typename _CharT , typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.367.5.67** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
**[static, inherited]**

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.367.5.68** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg ( void )`

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.367.5.69** `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits`  
`>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.367.5.70** `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits`  
`>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,`  
`inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic\_ios.h.

**5.367.5.71** `template<typename _CharT, typename _Traits> basic_istream<`  
`_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget`  
`( void )`

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

### 5.367.5.72 `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

### 5.367.5.73 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen ( char __c ) const [inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

#### 5.367.5.74 `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

##### Parameters

*wide* The new width value.

##### Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

#### 5.367.5.75 `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

##### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outputter>::do_put()`, and `std::operator>>()`.

#### 5.367.5.76 `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `isword` and `isword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `isword` and `isword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `isword` and `isword` arrays.

**5.367.6 Member Data Documentation****5.367.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::M_gcount` [protected]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.367.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.367.6.3 `const openmode std::ios_base::app` [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.367.6.4 `const openmode std::ios_base::ate` [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

#### 5.367.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such `///` as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

#### 5.367.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### 5.367.6.7 `const seekdir std::ios_base::beg` [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

#### 5.367.6.8 `const openmode std::ios_base::binary` [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `///` This is probably not what you think it is; see `///` <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

**5.367.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.367.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.367.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

**5.367.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.367.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

**5.367.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.367.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.367.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.367.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.367.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.367.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

**5.367.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

**5.367.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.367.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.367.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file ios\_base.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### **5.367.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios\_base.h.

#### **5.367.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

#### **5.367.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file ios\_base.h.

#### **5.367.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 302 of file ios\_base.h.

#### **5.367.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

#### **5.367.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.367.6.30** `const openmode std::ios_base::trunc` [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

**5.367.6.31** `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

**5.367.6.32** `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 5.368 `std::basic_istream<_CharT, _Traits>::sentry` Class Reference

Performs setup work for input streams.

### Public Types

- typedef `__istream_type::__ctype_type` `__ctype_type`
- typedef `_Traits::int_type` `__int_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_Traits` `traits_type`

### Public Member Functions

- `sentry` (`basic_istream<_CharT, _Traits> &__is`, `bool __noskipws=false`)
- `operator bool` () const

### 5.368.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _-
CharT, _Traits >::sentry
```

Performs setup work for input streams. Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 633 of file istream.

### 5.368.2 Member Typedef Documentation

```
5.368.2.1 template<typename _CharT, typename _Traits> typedef _Traits
std::basic_istream< _CharT, _Traits >::sentry::traits_type
```

Easy access to dependant types.

Definition at line 640 of file istream.

### 5.368.3 Constructor & Destructor Documentation

```
5.368.3.1 template<typename _CharT, typename _Traits> std::basic_istream<
_CharT, _Traits >::sentry::sentry ( basic_istream< _CharT, _Traits
> & __is, bool __noskipws = false ) [explicit]
```

The constructor performs all the work.

#### Parameters

*is* The input stream to guard.

*noskipws* Whether to consume whitespace or not.

If the stream state is good (*is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1751

---

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::__ctype_abstract_base<_CharT >::is()`, `std::basic_streambuf<_CharT, _Traits >::sgetc()`, and `std::basic_streambuf<_CharT, _Traits >::snextc()`.

### 5.368.4 Member Function Documentation

#### 5.368.4.1 `template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits >::sentry::operator bool ( ) const` [`inline`, `explicit`]

Quick status checking.

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 681 of file `istream`.

The documentation for this class was generated from the following files:

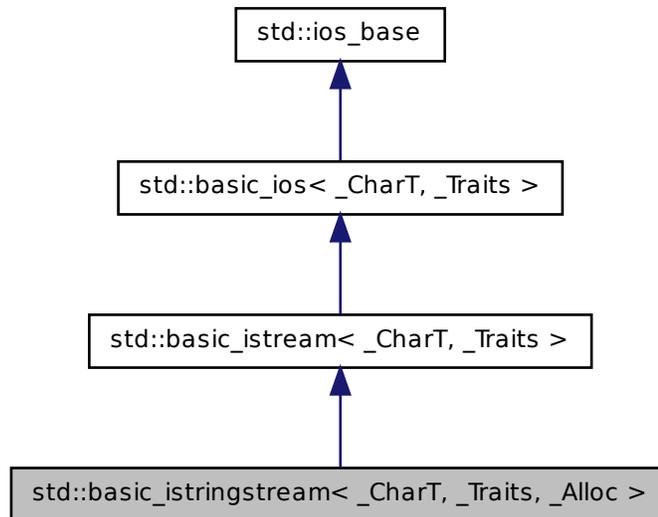
- [istream](#)
- [istream.tcc](#)

## 5.369 `std::basic_istream<_CharT, _Traits, _-Alloc >` Class Template Reference

Controlling input for `std::string`.

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_istringstream<_CharT, _Traits, _Alloc >`:



## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<char_type, traits_type>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
- typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
- typedef `_Alloc` `allocator_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `traits_type::int_type` `int_type`
- typedef `int` `io_state`

- typedef `_Ios_Iostate` `iostate`
  - typedef `traits_type::off_type` `off_type`
  - typedef `int` `open_mode`
  - typedef `_Ios_Openmode` `openmode`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef `int` `seek_dir`
  - typedef `_Ios_Seekdir` `seekdir`
  - typedef `std::streamoff` `streamoff`
  - typedef `std::streampos` `streampos`
  - typedef `_Traits` `traits_type`
- 
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > >` `__num_put_type`

## Public Member Functions

- `basic_istream` (`ios_base::openmode` `__mode=ios_base::in`)
- `basic_istream` (`const __string_type &__str`, `ios_base::openmode` `__mode=ios_base::in`)
- `~basic_istream` ()
- `const locale &_M_getloc` () const
- `void _M_setstate` (`iostate` `__state`)
- `bool bad` () const
- `void clear` (`iostate` `__state=goodbit`)
- `basic_ios &copyfmt` (`const basic_ios &__rhs`)
- `bool eof` () const
- `iostate exceptions` () const
- `void exceptions` (`iostate` `__except`)
- `bool fail` () const
- `char_type fill` () const
- `char_type fill` (`char_type` `__ch`)
- `fmtflags flags` () const
- `fmtflags flags` (`fmtflags` `__fmtfl`)
- `streamsize gcount` () const
- `template<>`  
`basic_istream< char > &getline` (`char_type *__s`, `streamsize` `__n`, `char_type` `__delim`)
- `template<>`  
`basic_istream< wchar_t > &getline` (`char_type *__s`, `streamsize` `__n`, `char_type` `__delim`)
- `locale getloc` () const
- `bool good` () const

- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__stringbuf_type * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void str (const __string_type &__s)`
  - `__string_type str () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>>` (bool &\_\_n)
- `__istream_type & operator>>` (short &\_\_n)
- `__istream_type & operator>>` (unsigned short &\_\_n)
- `__istream_type & operator>>` (int &\_\_n)
- `__istream_type & operator>>` (unsigned int &\_\_n)
- `__istream_type & operator>>` (long &\_\_n)
- `__istream_type & operator>>` (unsigned long &\_\_n)
- `__istream_type & operator>>` (long long &\_\_n)
- `__istream_type & operator>>` (unsigned long long &\_\_n)
- `__istream_type & operator>>` (float &\_\_f)
- `__istream_type & operator>>` (double &\_\_f)
- `__istream_type & operator>>` (long double &\_\_f)
- `__istream_type & operator>>` (void \*&\_\_p)
- `__istream_type & operator>>` (`__streambuf_type` \*\_\_sb)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`

- `streamsize` `readsome` (`char_type * __s`, `streamsize __n`)
- `__istream_type` & `putback` (`char_type __c`)
- `__istream_type` & `unget` ()
- `int` `sync` ()
- `pos_type` `tellg` ()
- `__istream_type` & `seekg` (`pos_type`)
- `__istream_type` & `seekg` (`off_type`, `ios_base::seekdir`)
  
- `operator void *` () `const`
- `bool operator!` () `const`

### Static Public Member Functions

- `static bool` `sync_with_stdio` (`bool __sync=true`)
- `static int` `xalloc` () `throw` ()

### Static Public Attributes

- `static const` `fmtflags` `adjustfield`
- `static const` `openmode` `app`
- `static const` `openmode` `ate`
- `static const` `iosstate` `badbit`
- `static const` `fmtflags` `basefield`
- `static const` `seekdir` `beg`
- `static const` `openmode` `binary`
- `static const` `fmtflags` `boolalpha`
- `static const` `seekdir` `cur`
- `static const` `fmtflags` `dec`
- `static const` `seekdir` `end`
- `static const` `iosstate` `eofbit`
- `static const` `iosstate` `failbit`
- `static const` `fmtflags` `fixed`
- `static const` `fmtflags` `floatfield`
- `static const` `iosstate` `goodbit`
- `static const` `fmtflags` `hex`
- `static const` `openmode` `in`
- `static const` `fmtflags` `internal`
- `static const` `fmtflags` `left`
- `static const` `fmtflags` `oct`
- `static const` `openmode` `out`
- `static const` `fmtflags` `right`
- `static const` `fmtflags` `scientific`
- `static const` `fmtflags` `showbase`

- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale &__loc`)
- void `_M_call_callbacks` (`event __ev`) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename `_ValueT` >  
`__istream_type &_M_extract` (`_ValueT &__v`)
- `_Words &_M_grow_words` (int `__index`, bool `__iword`)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf< _CharT, _Traits > *__sb`)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- const `__ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- bool `_M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word` [`_S_local_word_size`]
- const `__num_get_type * _M_num_get`
- const `__num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- int `_M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

### 5.369.1 Detailed Description

**template<typename `_CharT`, typename `_Traits`, typename `_Alloc`> class `std::basic_istringstream<_CharT, _Traits, _Alloc >`**

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as `sb`.

Definition at line 256 of file `sstream`.

### 5.369.2 Member Typedef Documentation

**5.369.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream<_CharT, _Traits >::_ctype_type [inherited]`**

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 71 of file `istream`.

**5.369.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_istream<_CharT, _Traits >::_num_get_type [inherited]`**

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file `istream`.

**5.369.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits >::_num_put_type [inherited]`**

These are non-standard types.

### 5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc >` Class Template Reference 1759

---

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits >](#), [std::basic\\_ostream<char, \\_Traits >](#), and [std::basic\\_ostream<char >](#).

Definition at line 84 of file `basic_ios.h`.

**5.369.2.4** `template<typename _CharT, typename _Traits, typename _Alloc > typedef _CharT std::basic_istringstream<_CharT, _Traits, _Alloc >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits >](#).

Definition at line 260 of file `sstream`.

**5.369.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

**5.369.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file ios\_base.h.

**5.369.2.7** `template<typename _CharT , typename _Traits , typename _Alloc >  
typedef traits_type::int_type std::basic_istream< _CharT,  
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream< _CharT, _Traits >`.

Definition at line 265 of file sstream.

**5.369.2.8** `typedef _Ios_Iostate std::ios_base::iostate` **[inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

**5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc >` Class Template Reference**

---

1761

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

**5.369.2.9 `template<typename _CharT, typename _Traits, typename _Alloc > typedef traits_type::off_type std::basic_istringstream<_CharT, _Traits, _Alloc >::off_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits >](#).

Definition at line 267 of file `sstream`.

**5.369.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

---

**5.369.2.11** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef traits_type::pos_type std::basic_istream< _CharT,  
_Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 266 of file `sstream`.

**5.369.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file `ios_base.h`.

**5.369.2.13** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef _Traits std::basic_istream< _CharT, _Traits, _Alloc  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 261 of file `sstream`.

### 5.369.3 Member Enumeration Documentation

**5.369.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

## 5.369.4 Constructor & Destructor Documentation

**5.369.4.1** `template<typename _CharT, typename _Traits, typename _Alloc > std::basic_istream<_CharT, _Traits, _Alloc >::basic_istream ( ios_base::openmode __mode = ios_base::in ) [inline, explicit]`

Default constructor starts with an empty string buffer.

### Parameters

*mode* Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using `mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 292 of file `sstream`.

**5.369.4.2** `template<typename _CharT, typename _Traits, typename _Alloc > std::basic_istream<_CharT, _Traits, _Alloc >::basic_istream ( const __string_type & __str, ios_base::openmode __mode = ios_base::in ) [inline, explicit]`

Starts with an existing string buffer.

### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using `str` and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 310 of file `sstream`.

**5.369.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istream<_CharT, _Traits, _Alloc>::~~basic_istream ( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 321 of file sstream.

## 5.369.5 Member Function Documentation

**5.369.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline, inherited]`

Locale access.

### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**5.369.5.2** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

**5.369 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 1765

---

**5.369.5.3** `template<typename _CharT, typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.369.5.4** `template<typename _CharT, typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (  
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

**Parameters**

*\_\_rhs* The source values for the copies.

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.369.5.5** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::eof ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic\_ios.h.

Referenced by `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_istream<_CharT, _Traits >::putback()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.369.5.6** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions ( ) const` [**inline**, **inherited**]

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.369.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions ( iostate __except )` [**inline**, **inherited**]

Throwing exceptions on errors.

**Parameters**

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

1767

```
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

### 5.369.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail( ) const` [`inline`, `inherited`]

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::tellg()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

### 5.369.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits >::fill( ) const` [`inline`, `inherited`]

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

**5.369.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.369.5.11** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.369.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

### 5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1769

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

**5.369.5.13** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::gcount ( ) const  
[inline, inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

**5.369.5.14** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits  
>::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.369.5.15** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (   
char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

*c* The character in which to store data.

**Returns**

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.369.5.16** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

**Parameters**

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

**5.369.5.17** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, `std::basic_streambuf<_CharT, _Traits >::sgetc()`, `std::basic_streambuf<_CharT, _Traits >::snextc()`, and `std::basic_streambuf<_CharT, _Traits >::sputc()`.

**5.369.5.18** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim )`  
[inherited]

Simple multiple-character extraction.

**Parameters**

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored

- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.369.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file istream.

**5.369.5.20** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

1773

*n* Maximum number of characters to extract.

*delim* A "stop" character.

### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3.  $n-1$  characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_streambuf<_CharT, _Traits >::sbumpc()`, `std::basic_ios<_CharT, _Traits >::setstate()`, `std::basic_streambuf<_CharT, _Traits >::sgetc()`, and `std::basic_streambuf<_CharT, _Traits >::snextc()`.

**5.369.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

### Returns

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

### 5.369.5.22 locale `std::ios_base::getloc ( ) const` [`inline`, `inherited`]

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

### 5.369.5.23 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const` [`inline`, `inherited`]

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

### 5.369.5.24 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n )` [`inherited`]

Simple extraction.

#### Returns

A character, or `eof()`.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1775

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

### 5.369.5.25 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.369.5.26** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.369.5.27** `template<typename _CharT , typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference

1777

**5.369.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.369.5.29** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

### Parameters

`__ix` Index into the array.

### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.369.5.30** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits >::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

### Parameters

`c` The character to narrow.

`dfault` The character to narrow.

### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.369.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.369.5.32** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.369.5.33** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( short & __n )` [`inherited`]

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

## 5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc >` Class Template Reference

1779

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.369.5.34** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

**5.369.5.35** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

**5.369.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file `istream`.

```
5.369.5.37 template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits >::operator>> ( void *& __p  
) [inline, inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

```
5.369.5.38 template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream<_CharT, _Traits  
>::operator>> ( int & __n ) [inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1781

---

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.369.5.39** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.369.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned int & __n ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

**5.369.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

**5.369.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

**5.369.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

**5.369 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 1783

---

**5.369.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

**5.369.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.369.5.46** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

```
5.369.5.47 template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits >::operator>> ( long long &  
__n ) [inline, inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

```
5.369.5.48 template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits >::operator>> ( bool & __n  
) [inline, inherited]
```

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

**5.369 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 1785

---

**5.369.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

**5.369.5.50** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek ( void ) [inherited]`

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.369.5.51** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 5.369.5.52 `streamsize std::ios_base::precision ( streamsize __prec )` [inline, inherited]

Changing flags.

##### Parameters

*prec* The new precision value.

##### Returns

The previous value of `precision()`.

Definition at line 629 of file ios\_base.h.

#### 5.369.5.53 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::putback ( char_type __c )` [inherited]

Unextracting a single character.

##### Parameters

*c* The character to push back into the input stream.

##### Returns

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

##### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios<`

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1787

`_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

### 5.369.5.54 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

### 5.369.5.55 `template<typename _CharT, typename _Traits, typename _Alloc > __stringbuf_type* std::basic_istream<_CharT, _Traits, _Alloc >::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits >`.

Definition at line 332 of file `sstream`.

### 5.369.5.56 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf ( basic_streambuf<_CharT, _Traits > * __sb ) [inherited]`

Changing the underlying buffer.

**Parameters**

*sb* The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.369.5.57** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

**5.369.5.58** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(  
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

**Parameters**

*s* A character array.

*n* Maximum number of characters to store.

### Returns

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_istream< _CharT, _Traits >::rdbuf()`, and `std::basic_istream< _CharT, _Traits >::setstate()`.

**5.369.5.59** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream< _CharT, _Traits >::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

### Parameters

*s* A character array.

*n* Maximum number of characters to store.

### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets `eofbit` and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### 5.369.5.60 `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

##### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.369.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

##### Parameters

`off` A file offset object.

`dir` The direction in which to seek.

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc >` Class Template Reference 1791

---

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.369.5.62** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

### Parameters

*pos* A file position object.

### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.369.5.63** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline, inherited]`

Setting new format flags.

### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

#### 5.369.5.64 `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

##### Parameters

*fmtfl* Additional flags to set.

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

#### 5.369.5.65 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate ( iostate __state ) [inline, inherited]`

Sets additional flags in the error state.

##### Parameters

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream<`

### 5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template

#### Reference

1793

`_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::unget()`, and `std::ws()`.

**5.369.5.66** `template<typename _CharT, typename _Traits, typename _Alloc > void std::basic_istringstream<_CharT, _Traits, _Alloc >::str ( const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

*s* The string to use as a new sequence.

Calls `rdbuf() ->str(s)`.

Definition at line 350 of file `sstream`.

**5.369.5.67** `template<typename _CharT, typename _Traits, typename _Alloc > __string_type std::basic_istringstream<_CharT, _Traits, _Alloc >::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

`rdbuf() ->str()`

Definition at line 340 of file `sstream`.

**5.369.5.68** `template<typename _CharT, typename _Traits > int std::basic_istream<_CharT, _Traits >::sync ( void ) [inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.369.5.69** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
**[static, inherited]**

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.369.5.70** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg ( void )`  
**[inherited]**

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

### 5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1795

---

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.369.5.71** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.369.5.72** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

**5.369.5.73** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

#### 5.369.5.74 `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

#### 5.369.5.75 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen ( char __c ) const [inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

## 5.369 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1797

---

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>>(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

### 5.369.5.76 `streamsize std::ios_base::width ( ) const` [`inline`, `inherited`]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

### 5.369.5.77 `streamsize std::ios_base::width ( streamsize __wide )` [`inline`, `inherited`]

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

### 5.369.5.78 `static int std::ios_base::xalloc ( ) throw ()` [`static`, `inherited`]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.369.6 Member Data Documentation****5.369.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount` [protected, inherited]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.369.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

**5.369.6.3 `const openmode std::ios_base::app` [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.369.6.4 `const openmode std::ios_base::ate` [static, inherited]**

Open and seek to end immediately after opening.

## 5.369 `std::basic_istringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1799

---

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

### 5.369.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.369.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

### 5.369.6.7 `const seekdir std::ios_base::beg` [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

### 5.369.6.8 `const openmode std::ios_base::binary` [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

#### **5.369.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

#### **5.369.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `std::basic_istream<_CharT, _Traits >::tellg()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

#### **5.369.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file `ios_base.h`.

#### **5.369.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

#### **5.369.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_date()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_time()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits`

### 5.369 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1801

---

`>>::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

#### 5.369.6.14 `const ios_base::failbit` [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, and `std::basic_ostream< _CharT, _Traits >::seekp()`.

#### 5.369.6.15 `const fmtflags std::ios_base::fixed` [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

#### 5.369.6.16 `const fmtflags std::ios_base::floatfield` [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

#### 5.369.6.17 `const ios_base::goodbit` [static, inherited]

Indicates all is well.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`,

`std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### **5.369.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### **5.369.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

#### **5.369.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

#### **5.369.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

#### **5.369.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

## 5.369 `std::basic_istringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1803

---

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

### 5.369.6.23 `const openmode std::ios_base::out` [static, inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

### 5.369.6.24 `const fmtflags std::ios_base::right` [static, inherited]

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file `ios_base.h`.

### 5.369.6.25 `const fmtflags std::ios_base::scientific` [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

### 5.369.6.26 `const fmtflags std::ios_base::showbase` [static, inherited]

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

### 5.369.6.27 `const fmtflags std::ios_base::showpoint` [static, inherited]

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 302 of file `ios_base.h`.

### 5.369.6.28 `const fmtflags std::ios_base::showpos` [static, inherited]

Generates a `+` sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

**5.369.6.29** `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

**5.369.6.30** `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

**5.369.6.31** `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

**5.369.6.32** `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

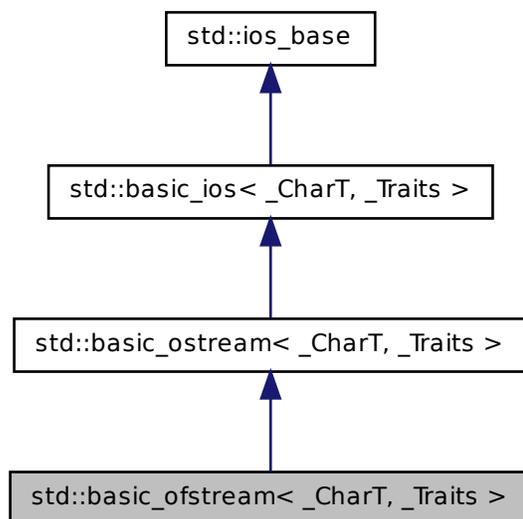
- [sstream](#)

## **5.370** `std::basic_ofstream<_CharT, _Traits >` Class Template Reference

Controlling output for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `s_b`.

Inheritance diagram for `std::basic_ofstream<_CharT, _Traits >`:



## Public Types

- typedef `ctype<_CharT > __ctype_type`
- typedef `basic_filebuf<char_type, traits_type > __filebuf_type`
- typedef `basic_ios<_CharT, _Traits > __ios_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __num_put_type`
- typedef `basic_ostream<char_type, traits_type > __ostream_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int **open\_mode**
  - typedef `_Ios_Openmode` **openmode**
  - typedef `traits_type::pos_type` **pos\_type**
  - typedef int **seek\_dir**
  - typedef `_Ios_Seekdir` **seekdir**
  - typedef `std::streamoff` **streamoff**
  - typedef `std::streampos` **streampos**
  - typedef `_Traits` **traits\_type**
- 
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > __-`  
`num_get_type`

## Public Member Functions

- `basic_ofstream` ()
- `basic_ofstream` (const char \*\_\_s, `ios_base::openmode` \_\_mode=`ios_`  
`base::out|ios_base::trunc`)
- `basic_ofstream` (const `std::string` &\_\_s, `ios_base::openmode` \_\_mode=`ios_`  
`base::out|ios_base::trunc`)
- `~basic_ofstream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` \_\_state)
- bool `bad` () const
- void `clear` (`iostate` \_\_state=`goodbit`)
- void `close` ()
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` (`iostate` \_\_except)
- bool `fail` () const
- `char_type` `fill` () const
- `char_type` `fill` (`char_type` \_\_ch)
- `fmtflags` `flags` () const
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- `__ostream_type` & `flush` ()
- `locale` `getloc` () const
- bool `good` () const
- `locale` `imbue` (const `locale` &\_\_loc)
- bool `is_open` ()
- bool `is_open` () const
- long & `isword` (int \_\_ix)
- char `narrow` (`char_type` \_\_c, char \_\_dfault) const

- `void open (const char *__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
  - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__filebuf_type * rdbuf () const`
  - `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `__ostream_type & seekp (pos_type)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `pos_type tellp ()`
  - `basic_ostream<_CharT, _Traits> * tie () const`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __t_fiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`

- `__ostream_type & operator<<` (unsigned long \_\_n)
- `__ostream_type & operator<<` (bool \_\_n)
- `__ostream_type & operator<<` (short \_\_n)
- `__ostream_type & operator<<` (unsigned short \_\_n)
- `__ostream_type & operator<<` (int \_\_n)
- `__ostream_type & operator<<` (unsigned int \_\_n)
- `__ostream_type & operator<<` (long long \_\_n)
- `__ostream_type & operator<<` (unsigned long long \_\_n)
- `__ostream_type & operator<<` (double \_\_f)
- `__ostream_type & operator<<` (float \_\_f)
- `__ostream_type & operator<<` (long double \_\_f)
- `__ostream_type & operator<<` (const void \*\_\_p)
- `__ostream_type & operator<<` (`__streambuf_type` \*\_\_sb)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put` (char\_type \_\_c)
- `void _M_write` (const char\_type \*\_\_s, streamsize \_\_n)
- `__ostream_type & write` (const char\_type \*\_\_s, streamsize \_\_n)
- `operator void * () const`
- `bool operator!` () const

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`

- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosate eofbit`
- static const `iosate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename `_ValueT`>  
  `__ostream_type` & `_M_insert` (`_ValueT` \_\_v)
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

### 5.370.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ofstream< _-  
CharT, _Traits >
```

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 583 of file `fstream`.

## 5.370.2 Member Typedef Documentation

**5.370.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ofstream<_CharT, _Traits >::_ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 71 of file `ostream`.

**5.370.2.2** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > std::basic_ios<_CharT, _Traits >::_num_get_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits >](#), [std::basic\\_istream<char, \\_Traits >](#), and [std::basic\\_istream<char >](#).

Definition at line 86 of file `basic_ios.h`.

**5.370.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > std::basic_ofstream<_CharT, _Traits >::_num_put_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file `ostream`.

**5.370.2.4** `template<typename _CharT, typename _Traits > typedef _CharT std::basic_ofstream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ofstream<\\_CharT, \\_Traits >](#).

Definition at line 587 of file `fstream`.

**5.370.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`  
[*inherited*]

The type of an event callback function.

**Parameters**

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

**5.370.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags` [*inherited*]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

**5.370.2.7** `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ofstream<_CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 589 of file `fstream`.

**5.370.2.8** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

**5.370.2.9** `template<typename _CharT , typename _Traits > typedef traits_type::off_type std::basic_ofstream<_CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Definition at line 591 of file `fstream`.

#### 5.370.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

#### 5.370.2.11 `template<typename _CharT, typename _Traits > typedef traits_type::pos_type std::basic_ofstream<_CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Definition at line 590 of file `fstream`.

#### 5.370.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1815

Definition at line 401 of file `ios_base.h`.

**5.370.2.13** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_ofstream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 588 of file `fstream`.

### 5.370.3 Member Enumeration Documentation

**5.370.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

### 5.370.4 Constructor & Destructor Documentation

**5.370.4.1** `template<typename _CharT, typename _Traits>  
std::basic_ofstream<_CharT, _Traits>::basic_ofstream ( )  
[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 609 of file `fstream`.

**5.370.4.2** `template<typename _CharT, typename _Traits>  
std::basic_ofstream<_CharT, _Traits>::basic_ofstream ( const char  
* __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc )  
[inline, explicit]`

Create an output file stream.

**Parameters**

*s* Null terminated string specifying the filename.  
*mode* Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::out](#)|[ios\\_base::trunc](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 624 of file `fstream`.

```
5.370.4.3 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream
( const std::string & __s, ios_base::openmode __mode =
ios_base::out|ios_base::trunc ) [inline, explicit]
```

Create an output file stream.

**Parameters**

*s* `std::string` specifying the filename.  
*mode* Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::out](#)|[ios\\_base::trunc](#) is automatically included in *mode*.

Definition at line 642 of file `fstream`.

```
5.370.4.4 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ( )
[inline]
```

The destructor does nothing.

The file is closed by the `filebuf` object, not the formatting stream.

Definition at line 657 of file `fstream`.

**5.370.5 Member Function Documentation**

```
5.370.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline,
inherited]
```

Locale access.

**Returns**

A reference to the current locale.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1817

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**5.370.5.2** `template<typename _CharT, typename _Traits> void std::basic_ofstream<_CharT, _Traits>::_M_write ( const char_type * __s, streamsize __n ) [inline, inherited]`

Simple insertion.

### Parameters

*c* The character to insert.

### Returns

\*this

Tries to insert *c*.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

**5.370.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

**5.370.5.4** `template<typename _CharT , typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.370.5.5** `template<typename _CharT , typename _Traits > void  
std::basic_ofstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 737 of file `fstream`.

**5.370.5.6** `template<typename _CharT , typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (   
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

*\_\_rhs* The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1819

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

### 5.370.5.7 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.370.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.370.5.9 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

**5.370.5.10** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::fail ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.370.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.370.5.12** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill( ) const [inline,  
inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.370.5.13** `fmtflags std::ios_base::flags( ) const [inline, inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.370.5.14** `fmtflags std::ios_base::flags( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

**5.370.5.15** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::flush ( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.370.5.16** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

**Returns**

A copy of the current locale.

If `imbue (loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale ( )`, the global C++ locale.

Definition at line 694 of file ios\_base.h.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::money_put<_CharT, _OutIter >::do_put()`, `std::basic_ios<_CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.370.5.17** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good ( ) const [inline,  
inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.370.5.18** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.370.5.19** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.370.5.20** `template<typename _CharT, typename _Traits> bool  
std::basic_ofstream<_CharT, _Traits>::is_open ( ) [inline]`

Wrapper to test for an open file.

**Returns**

`rdbuf ()->is_open ()`

Definition at line 676 of file fstream.

**5.370.5.21** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.370.5.22** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1825

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>>(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

```
5.370.5.23 template<typename _CharT , typename _Traits > void  
std::basic_ofstream<_CharT, _Traits >::open ( const char * __s,  
ios_base::openmode __mode = ios_base::out | ios_base::trunc )  
[inline]
```

Opens an external file.

### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 697 of file `fstream`.

```
5.370.5.24 template<typename _CharT , typename _Traits > void  
std::basic_ofstream<_CharT, _Traits >::open ( const std::string &  
__s, ios_base::openmode __mode = ios_base::out | ios_base::trunc  
) [inline]
```

Opens an external file.

### Parameters

*s* The name of the file.

*mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 718 of file `fstream`.

**5.370.5.25** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

**5.370.5.26** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

**5.370.5.27** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.370.5.28** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1827

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file `ostream`.

```
5.370.5.29 template<typename _CharT, typename _Traits> basic_ostream<  
    _CharT, _Traits> & std::basic_ofstream<_CharT, _Traits  
    >::operator<<( __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

```
5.370.5.30 template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( ios_base  
&(*)(ios_base &) __pf ) [inline, inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**5.370.5.31** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.370.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

**5.370.5.33** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( float __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

**5.370.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits >::operator<< ( long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

**5.370.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits >::operator<< ( long double __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

**5.370.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

**5.370.5.37** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.370.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1831

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

**5.370.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<<( bool __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file `ostream`.

**5.370.5.40** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.370.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

**5.370.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

**5.370.5.43** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1833

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

**5.370.5.44** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.370.5.45** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.370.5.46** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put ( char_type __c ) [inherited]`

Simple insertion.

### Parameters

*c* The character to insert.

### Returns

\*this

Tries to insert *c*.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

### 5.370.5.47 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

### Parameters

`__ix` Index into the array.

### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios\_base.h.

### 5.370.5.48 `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_ofstream<_CharT, _Traits>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

### Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 668 of file fstream.

**5.370.5.49** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.370.5.50** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline, inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.370.5.51** `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

*\_\_fn* The function to add.

*\_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.370.5.52** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.370.5.53** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1837

---

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.370.554 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` [`inline`, `inherited`]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

### 5.370.555 `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` [`inline`, `inherited`]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

```
5.370.5.56 template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]
```

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

```
5.370.5.57 static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]
```

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.370.5.58** `template<typename _CharT, typename _Traits > basic_ofstream<_CharT, _Traits >::pos_type std::basic_ofstream<_CharT, _Traits >::tellp( ) [inherited]`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits >::rdbuf()`.

**5.370.5.59** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::tie( basic_ofstream<_CharT, _Traits >* __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

**5.370.5.60** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

#### 5.370.5.61 `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

##### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

#### 5.370.5.62 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits >::widen ( char __c ) const [inline, inherited]`

Widens characters.

##### Parameters

*c* The character to widen.

##### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.370.5.63** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.370.5.64** `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.370.5.65** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

**Parameters**

*s* The array to insert.

*n* Maximum number of characters to insert.

**Returns**

`*this`

Characters are copied from *s* and inserted into the stream until one of the following happens:

- $n$  characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**5.370.5.66 static int std::ios\_base::xalloc ( ) throw () [static, inherited]**

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.370.6 Member Data Documentation****5.370.6.1 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.370.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.370.6.3 `const openmode std::ios_base::ate` [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**5.370.6.4 `const iostate std::ios_base::badbit` [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such `///` as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_istream<_CharT, _Traits>::unset()`.

**5.370.6.5 `const fmtflags std::ios_base::basefield` [static, inherited]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.370.6.6 `const seekdir std::ios_base::beg` [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

**5.370.6.7 `const openmode std::ios_base::binary` [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `///` This is probably not what you think it is; see `///`

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### **5.370.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### **5.370.6.9 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### **5.370.6.10 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### **5.370.6.11 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### **5.370.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, and `std::time_get< _CharT, _InIter >::do_get_time()`.

## 5.370 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 1845

`>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, and `std::ws()`.

### 5.370.6.13 `const ios_base::failbit` [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

### 5.370.6.14 `const fmtflags std::ios_base::fixed` [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file `ios_base.h`.

### 5.370.6.15 `const fmtflags std::ios_base::floatfield` [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

### 5.370.6.16 `const ios_base::goodbit` [static, inherited]

Indicates all is well.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`,

std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unset().

#### 5.370.6.17 const fmtflags std::ios\_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.370.6.18 const openmode std::ios\_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 383 of file ios\_base.h.

Referenced by std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

#### 5.370.6.19 const fmtflags std::ios\_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 280 of file ios\_base.h.

#### 5.370.6.20 const fmtflags std::ios\_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.370.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.370.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for ostream and fstream.

Definition at line 386 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::seekp(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.370.6.23 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file ios\_base.h.

**5.370.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.370.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios\_base.h.

**5.370.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.370.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

#### **5.370.6.28** `const fmtflags std::ios_base::skipws` [`static`, `inherited`]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

#### **5.370.6.29** `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

#### **5.370.6.30** `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

#### **5.370.6.31** `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

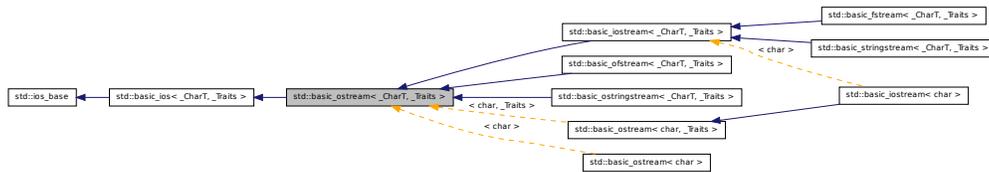
- [fstream](#)

## **5.371** `std::basic_ostream<_CharT, _Traits >` Class Template Reference

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.

Inheritance diagram for `std::basic_ostream<_CharT, _Traits >`:



## Classes

- class `sentry`  
*Performs setup work for output streams.*

## Public Types

- typedef `ctype<_CharT > __ctype_type`
- typedef `basic_ios<_CharT, _Traits > __ios_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __num_put_type`
- typedef `basic_ostream<_CharT, _Traits > __ostream_type`
- typedef `basic_streambuf<_CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `_Traits::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `_Traits::off_type off_type`
- typedef `int open_mode`
- typedef `_Ios_Openmode openmode`
- typedef `_Traits::pos_type pos_type`
- typedef `int seek_dir`
- typedef `_Ios_Seekdir seekdir`
- typedef `std::streamoff streamoff`
- typedef `std::streampos streampos`
- typedef `_Traits traits_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > __num_get_type`

## Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- virtual [~basic\\_ostream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [\\_\\_ostream\\_type](#) & [seekp](#) ([pos\\_type](#))
- [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)
- [pos\\_type](#) [tellp](#) ()
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_tiestr)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [char\\_type](#) [widen](#) ([char](#) \_\_c) const
- [streamsize](#) [width](#) () const

- `streamsize width (streamsize __wide)`
  - `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`

- void `_M_write` (const `char_type` \*`__s`, `streamsize` `__n`)
- `__ostream_type` & `write` (const `char_type` \*`__s`, `streamsize` `__n`)
- `operator void *` () const
- `bool operator!` () const

### Static Public Member Functions

- static `bool sync_with_stdio` (bool `__sync=true`)
- static `int xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename `_ValueT` >  
`__ostream_type` & `_M_insert` (`_ValueT` \_\_v)
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

## Friends

- class `sentry`

### 5.371.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ostream< _CharT, _Traits >
```

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.

Definition at line 55 of file ostream.

### 5.371.2 Member Typedef Documentation

```
5.371.2.1 template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_ostream< _CharT, _Traits  
>::_ctype_type
```

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file ostream.

```
5.371.2.2 template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_ios< _CharT, _Traits >::_num_get_type  
[inherited]
```

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<char, \\_Traits>](#), and [std::basic\\_istream<char>](#).

Definition at line 86 of file basic\_ios.h.

```
5.371.2.3 template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >  
std::basic_ostream< _CharT, _Traits >::_num_put_type
```

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file ostream.

#### 5.371.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ostream<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits >`.

Reimplemented in `std::basic_ofstream<_CharT, _Traits >`, `std::basic_fstream<_CharT, _Traits >`, `std::basic_iostream<_CharT, _Traits >`, `std::basic_ostringstream<_CharT, _Traits, _Alloc >`, `std::basic_stringstream<_CharT, _Traits, _Alloc >`, and `std::basic_istream<char >`.

Definition at line 59 of file `ostream`.

#### 5.371.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

##### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

#### 5.371.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`

- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

#### **5.371.2.7** `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ostream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 60 of file `ostream`.

**5.371.2.8** `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

**5.371.2.9** `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream<_CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits >`.

Reimplemented in `std::basic_ofstream<_CharT, _Traits >`, `std::basic_fstream<_CharT, _Traits >`, `std::basic_iostream<_CharT, _Traits >`, `std::basic_ostringstream<_CharT, _Traits, _Alloc >`, `std::basic_stringstream<_CharT, _Traits, _Alloc >`, and `std::basic_istream<char >`.

Definition at line 62 of file `ostream`.

**5.371.2.10** `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`

- out
- trunc

Definition at line 369 of file ios\_base.h.

#### 5.371.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ostream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 61 of file ostream.

#### 5.371.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file ios\_base.h.

#### 5.371.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream<`

[\\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), and [std::basic\\_ostream< char >](#).

Definition at line 63 of file ostream.

### **5.371.3 Member Enumeration Documentation**

#### **5.371.3.1 enum std::ios\_base::event [inherited]**

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 427 of file ios\_base.h.

### **5.371.4 Constructor & Destructor Documentation**

#### **5.371.4.1 template<typename \_CharT, typename \_Traits> std::basic\_ostream< \_CharT, \_Traits >::basic\_ostream ( \_\_streambuf\_type \* \_\_sb ) [inline, explicit]**

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 82 of file ostream.

#### **5.371.4.2 template<typename \_CharT, typename \_Traits> virtual std::basic\_ostream< \_CharT, \_Traits >::~~basic\_ostream ( ) [inline, virtual]**

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 91 of file ostream.

### **5.371.5 Member Function Documentation**

#### **5.371.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline, inherited]**

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**5.371.5.2** `template<typename _CharT, typename _Traits> void  
std::basic_ostream<_CharT, _Traits>::_M_write ( const char_type  
* __s, streamsize __n ) [inline]`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

**5.371.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

**5.371.5.4** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.371.5.5** `template<typename _CharT, typename _Traits> basic_ios<  
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (   
const basic_ios<_CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

*\_\_rhs* The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.371.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits>::eof ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic\_ios.h.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.371.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions( ) const` [**inline**, **inherited**]

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.371.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions( iostate __except )` [**inline**, **inherited**]

Throwing exceptions on errors.

**Parameters**

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

```

#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file `basic_ios.h`.

#### 5.371.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail( ) const` [`inline`, `inherited`]

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

#### 5.371.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill( ) const` [`inline`, `inherited`]

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.371.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.371.5.12** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.371.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

**5.371.5.14** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( )`

Synchronizing the stream buffer.

#### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.371.5.15** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.371.5.16** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.371.5.17** `template<typename _CharT, typename _Traits> locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.371.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.371.5.19** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `isword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.371.5.20** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.371.5.21** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

**5.371.5.22** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

**5.371.5.23** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( __ios_type  
&(*)(__ios_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.371.5.24** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned  
short __n ) [inline]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file `ostream`.

**5.371.5.25** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< ( __streambuf_type * __sb )`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.371.5.26** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long __n ) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

```
5.371.5.27 template<typename _CharT, typename _Traits > basic_ostream<
    _CharT, _Traits > & std::basic_ostream< _CharT, _Traits
    >::operator<<( int __n )
```

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

```
5.371.5.28 template<typename _CharT, typename _Traits> __ostream_type&
    std::basic_ostream< _CharT, _Traits >::operator<<(
    __ostream_type &(*)(__ostream_type &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

```
5.371.5.29 template<typename _CharT, typename _Traits> __ostream_type&
    std::basic_ostream< _CharT, _Traits >::operator<<( unsigned int
    __n ) [inline]
```

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

## 5.371 `std::basic_ostream<_CharT, _Traits >` Class Template Reference 1871

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file `ostream`.

**5.371.5.30** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( unsigned long __n ) [inline]`

Basic arithmetic inserters.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

**5.371.5.31** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( long long __n ) [inline]`

Basic arithmetic inserters.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file `ostream`.

**5.371.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned long long __n ) [inline]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.371.5.33** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( bool __n ) [inline]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

**5.371.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( const void * __p ) [inline]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file `ostream`.

**5.371.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( double __f ) [inline]`

Basic arithmetic inserters.

**Parameters**

`A` variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file `ostream`.

**5.371.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `io manip` header.

Definition at line 127 of file `ostream`.

**5.371.5.37** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( short __n )`

Basic arithmetic inserters.

**Parameters**

`A` variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.371.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( float __f ) [inline]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file `ostream`.

**5.371.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long double __f ) [inline]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file `ostream`.

---

**5.371.5.40** `streamsize std::ios_base::precision ( streamsize __prec )`  
`[inline, inherited]`

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.371.5.41** `streamsize std::ios_base::precision ( ) const` `[inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.371.5.42** `template<typename _CharT, typename _Traits> basic_ostream<`  
`_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (`  
`char_type __c )`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

`*this`

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

#### 5.371.5.43 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

##### Parameters

`__ix` Index into the array.

##### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios\_base.h.

#### 5.371.5.44 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

##### Parameters

`sb` The new stream buffer.

##### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

## 5.371 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 1877

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.371.5.45** `template<typename _CharT, typename _Traits>`  
`basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,`  
`_Traits>::rdbuf( ) const [inline, inherited]`

Accessing the underlying buffer.

### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.371.5.46** `template<typename _CharT, typename _Traits> iostate`  
`std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,`  
`inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file [basic\\_ios.h](#).

**5.371.5.47** `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.371.5.48** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( pos_type __pos )`

Changing the current write position.

**Parameters**

`pos` A file position object.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file [ostream.tcc](#).

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**5.371.5.49** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( off_type __off, ios_base::seekdir __dir )`

Changing the current write position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.371.5.50** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios\_base.h.

**5.371.5.51** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.371.5.52** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

**Parameters**

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic\_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.371.5.53** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

#### 5.371.554 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp ( )`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

#### 5.371.555 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline, inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or `NULL` if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.371.5.56** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( basic_ostream<_CharT, _Traits > * tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

**5.371.5.57** `void std::ios_base::unsetf ( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.371.5.58** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits >::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

## 5.371 `std::basic_ostream<_CharT, _Traits>` Class Template Reference 1883

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>>(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

### 5.371.5.59 `streamsize std::ios_base::width ( ) const` [`inline`, `inherited`]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

### 5.371.5.60 `streamsize std::ios_base::width ( streamsize __wide )` [`inline`, `inherited`]

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

### 5.371.5.61 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n )`

Character string insertion.

**Parameters**

- s* The array to insert.
- n* Maximum number of characters to insert.

**Returns**

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**5.371.5.62 static int std::ios\_base::xalloc ( ) throw () [static, inherited]**

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.371.6 Member Data Documentation****5.371.6.1 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

**5.371.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 372 of file ios\_base.h.

**5.371.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.371.6.4 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.371.6.5 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.371.6.6 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

#### **5.371.6.7 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

#### **5.371.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### **5.371.6.9 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### **5.371.6.10 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### **5.371.6.11 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.371.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

**5.371.6.13 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.371.6.14 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.371.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.371.6.16 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.371.6.17 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.371.6.18 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::underflow()`.

**5.371.6.19 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

**5.371.6.20 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

**5.371.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.371.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.371.6.23 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file `ios_base.h`.

**5.371.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.371.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

**5.371.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.371.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.371.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.371.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.371.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.371.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

## 5.372 `std::basic_ostream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for output streams.

### Public Member Functions

- [sentry](#) (`basic_ostream< _CharT, _Traits > &__os`)
- [~sentry](#) ()
- [operator bool](#) () const

#### 5.372.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ostream< _CharT, _Traits >::sentry`

Performs setup work for output streams. Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 377 of file ostream.

#### 5.372.2 Constructor & Destructor Documentation

**5.372.2.1** `template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::sentry::sentry ( basic_ostream< _CharT, _Traits > & __os ) [explicit]`

The constructor performs preparatory work.

#### Parameters

*os* The output stream to guard.

If the stream state is good (*os.good()* is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

**5.372.2.2** `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits >::sentry::~sentry ( )`  
`[inline]`

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 405 of file `ostream`.

References `std::uncaught_exception()`.

### 5.372.3 Member Function Documentation

**5.372.3.1** `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits >::sentry::operator bool ( )`  
`const [inline, explicit]`

Quick status checking.

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

Definition at line 426 of file `ostream`.

The documentation for this class was generated from the following files:

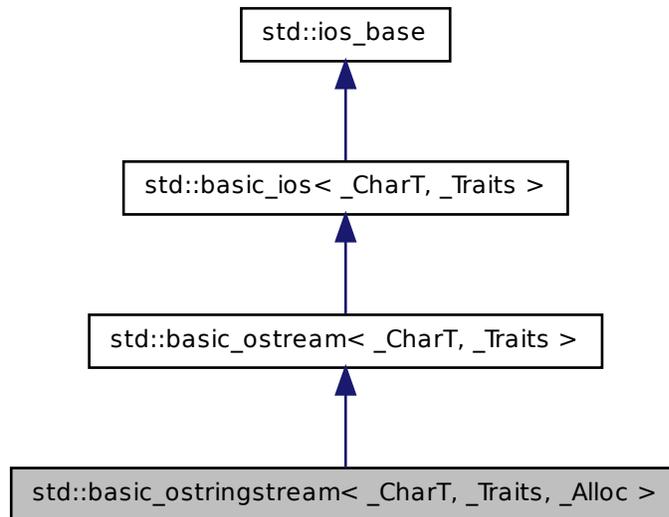
- [ostream](#)
- [ostream.tcc](#)

## 5.373 `std::basic_ostringstream< _CharT, _Traits, _- Alloc >` Class Template Reference

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ostringstream<_CharT, _Traits, _Alloc>`:



## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<char_type, traits_type>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
- typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
- typedef `_Alloc` `allocator_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `traits_type::int_type` `int_type`
- typedef `int` `io_state`

- typedef `_Ios_Iostate` [iostate](#)
  - typedef `traits_type::off_type` [off\\_type](#)
  - typedef `int` **open\_mode**
  - typedef `_Ios_Openmode` [openmode](#)
  - typedef `traits_type::pos_type` [pos\\_type](#)
  - typedef `int` **seek\_dir**
  - typedef `_Ios_Seekdir` [seekdir](#)
  - typedef `std::streamoff` **streamoff**
  - typedef `std::streampos` **streampos**
  - typedef `_Traits` [traits\\_type](#)
- 
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits > > __-`  
`num_get_type`

## Public Member Functions

- [basic\\_ostringstream](#) (`ios_base::openmode __mode=ios_base::out`)
- [basic\\_ostringstream](#) (`const __string_type &__str, ios_base::openmode __-`  
`mode=ios_base::out`)
- [~basic\\_ostringstream](#) ()
- `const locale &_M_getloc` () const
- `void _M_setstate` (`iostate __state`)
- `bool bad` () const
- `void clear` (`iostate __state=goodbit`)
- [basic\\_ios](#) & [copyfmt](#) (`const basic_ios &__rhs`)
- `bool eof` () const
- `iostate exceptions` () const
- `void exceptions` (`iostate __except`)
- `bool fail` () const
- `char_type fill` () const
- `char_type fill` (`char_type __ch`)
- `fmtflags flags` (`fmtflags __fmtfl`)
- `fmtflags flags` () const
- `__ostream_type & flush` ()
- `locale getloc` () const
- `bool good` () const
- `locale imbue` (`const locale &__loc`)
- `long & iword` (`int __ix`)
- `char narrow` (`char_type __c, char __dfault`) const
- `streamsize precision` () const
- `streamsize precision` (`streamsize __prec`)
- `void *& pword` (`int __ix`)

- `__stringbuf_type * rdbuf ()` const
  - `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
  - `iosstate rdstate ()` const
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `void setstate (iosstate __state)`
  - `void str (const __string_type & __s)`
  - `__string_type str ()` const
  - `pos_type tellp ()`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tistr)`
  - `basic_ostream<_CharT, _Traits> * tie ()` const
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c)` const
  - `streamsize width ()` const
  - `streamsize width (streamsize __wide)`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`

- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type * __sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type * __s, streamsize __n)`
- `__ostream_type & write (const char_type * __s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync=true`)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`

- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

## Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename [\\_ValueT](#) >  
[\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

### 5.373.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_ostringstream<_CharT, _Traits, _Alloc>
```

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 366 of file `sstream`.

## 5.373.2 Member Typedef Documentation

**5.373.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream<_CharT, _Traits>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file `ostream`.

**5.373.2.2** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<char, \\_Traits>](#), and [std::basic\\_istream<char>](#).

Definition at line 86 of file `basic_ios.h`.

**5.373.2.3** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ostream<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `ostream`.

**5.373.2.4** `template<typename _CharT, typename _Traits, typename _Alloc> typedef _CharT std::basic_ostringstream<_CharT, _Traits, _Alloc>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 370 of file `sstream`.

**5.373.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`  
**[inherited]**

The type of an event callback function.

**Parameters**

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

**5.373.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

### 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1901

---

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

**5.373.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef traits_type::int_type std::basic_ostringstream<_CharT,`  
`_Traits, _Alloc>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 375 of file `sstream`.

**5.373.2.8** `typedef _Ios_Iostate std::ios_base::iostate` **[inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

**5.373.2.9** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef traits_type::off_type std::basic_ostringstream<_CharT,`  
`_Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Definition at line 377 of file sstream.

#### 5.373.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

#### 5.373.2.11 `template<typename _CharT, typename _Traits, typename _Alloc> typedef traits_type::pos_type std::basic_ostringstream<_CharT, _Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits >](#).

Definition at line 376 of file sstream.

#### 5.373.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1903

---

Definition at line 401 of file `ios_base.h`.

**5.373.2.13** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef _Traits std::basic_ostringstream< _CharT, _Traits, _Alloc`  
`>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 371 of file `sstream`.

### 5.373.3 Member Enumeration Documentation

**5.373.3.1** `enum std::ios_base::event` `[inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

### 5.373.4 Constructor & Destructor Documentation

**5.373.4.1** `template<typename _CharT, typename _Traits, typename`  
`_Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc`  
`>::basic_ostringstream ( ios_base::openmode __mode =`  
`ios_base::out ) [inline, explicit]`

Default constructor starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 402 of file `sstream`.

**5.373.4.2** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream ( const __string_type & __str, ios_base::openmode __mode = ios_base::out ) [inline, explicit]`

Starts with an existing string buffer.

#### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *str* and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 420 of file `sstream`.

**5.373.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc >::~~basic_ostringstream ( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 431 of file `sstream`.

## 5.373.5 Member Function Documentation

**5.373.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get<`

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference 1905**

---

\_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::time\_put< \_CharT, \_OutIter >::put().

**5.373.5.2** `template<typename _CharT, typename _Traits> void  
std::basic_ostream< _CharT, _Traits >::M_write ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

**5.373.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

**5.373.5.4** `template<typename _CharT, typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.373.5.5** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt ( const basic_ios<_CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

**Parameters**

*\_\_rhs* The source values for the copies.

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.373.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

## 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1907

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### 5.373.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const` [`inline`, `inherited`]

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

### 5.373.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except )` [`inline`, `inherited`]

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");
```

```

std::cerr << "Setting badbit\n";
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file basic\_ios.h.

#### 5.373.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail( ) const` `[inline, inherited]`

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic\_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### 5.373.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( ) const` `[inline, inherited]`

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 360 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

#### 5.373.5.11 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( char_type __ch )` `[inline, inherited]`

Sets a new *empty* character.

## 5.373 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1909

---

### Parameters

*ch* The new character.

### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

### 5.373.5.12 `fmtflags std::ios_base::flags ( ) const` [`inline`, `inherited`]

Access to format flags.

### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

### 5.373.5.13 `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` [`inline`, `inherited`]

Setting new format flags all at once.

### Parameters

*fmtfl* The new flags to set.

### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

**5.373.5.14** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush ( ) [inherited]`

Synchronizing the stream buffer.

#### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**5.373.5.15** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.373.5.16** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1911**

---

**5.373.5.17** `template<typename _CharT, typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.373.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.373.5.19** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

**5.373.5.20** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits >::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

*c* The character to narrow.

*dfault* The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

**5.373.5.21** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits >::operator void *( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1913**

---

**5.373.5.22** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

**5.373.5.23** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( long __n )  
[inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file `ostream`.

**5.373.5.24** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file `ostream`.

**5.373.5.25** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( __ostream_type &(*)(__ostream_type &) _pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.373.5.26** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long double _f ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file `ostream`.

**5.373.5.27** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( bool _n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file `ostream`.

**5.373 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** **1915**

---

**5.373.5.28** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

**5.373.5.29** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `omanip` header.

Definition at line 117 of file ostream.

**5.373.5.30** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

**Parameters**

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into \*this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.373.5.31** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

**5.373.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

### 5.373 `std::basic_ostream<_CharT, _Traits, _Alloc>` Class Template Reference 1917

---

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file `ostream`.

**5.373.5.33** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.373.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

**5.373.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

**5.373.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

**5.373.5.37** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< ( short __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference 1919**

---

**5.373.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( float __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

**5.373.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.373.5.40** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.373.5.41** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.373.5.42** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put ( char_type __c ) [inherited]`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

`*this`

Tries to insert *c*.

**Note**

This function is not overloaded on signed `char` and unsigned `char`.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

## 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1921

---

### 5.373.5.43 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

### 5.373.5.44 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

`sb` The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits >::clear()`.

**5.373.5.45** `template<typename _CharT, typename _Traits, typename _Alloc>  
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits,  
_Alloc >::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits >`.

Definition at line 442 of file `sstream`.

**5.373.5.46** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits >::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.373.5.47** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1923**

---

**5.373.5.48** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.373.5.49** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.373.5.50** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )`  
`[inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.373.5.51** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.373.5.52** `template<typename _CharT, typename _Traits> void`  
`std::basic_ios<_CharT, _Traits >::setstate ( iostate __state )`  
`[inline, inherited]`

Sets additional flags in the error state.

## 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference

1925

### Parameters

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.373.5.53** `template<typename _CharT, typename _Traits, typename _Alloc>  
__string_type std::basic_ostringstream<_CharT, _Traits, _Alloc>::str( ) const [inline]`

Copying out the string buffer.

### Returns

`rdbuf() ->str()`

Definition at line 450 of file `sstream`.

Referenced by `std::operator<<()`.

**5.373.5.54** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_ostringstream<_CharT, _Traits, _Alloc>::str (   
const __string_type & __s ) [inline]`

Setting a new buffer.

### Parameters

*s* The string to use as a new sequence.

Calls `rdbuf() ->str(s)`.

Definition at line 460 of file `sstream`.

**5.373.5.55** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
`[static, inherited]`

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.373.5.56** `template<typename _CharT, typename _Traits> basic_ostream<`  
`_CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits`  
`>::tellp ( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits>::rdbuf()`.

**5.373.5.57** `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits>* std::basic_ios< _CharT, _Traits`  
`>::tie ( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

### 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1927

---

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.373.5.58** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( basic_ostream<_CharT, _Traits > * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

**5.373.5.59** `void std::ios_base::unsetf ( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.373.5.60** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits >::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

### 5.373.5.61 `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

### 5.373.5.62 `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.373 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** **1929**

---

**5.373.5.63** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

**Parameters**

- s* The array to insert.
- n* Maximum number of characters to insert.

**Returns**

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**5.373.5.64** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.373.6 Member Data Documentation

### 5.373.6.1 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

### 5.373.6.2 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

### 5.373.6.3 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

### 5.373.6.4 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such `///` as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

## 5.373 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 1931

---

### 5.373.6.5 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

### 5.373.6.6 `const seekdir std::ios_base::beg` [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 404 of file `ios_base.h`.

### 5.373.6.7 `const openmode std::ios_base::binary` [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

### 5.373.6.8 `const fmtflags std::ios_base::boolalpha` [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

### 5.373.6.9 `const seekdir std::ios_base::cur` [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

### 5.373.6.10 `const fmtflags std::ios_base::dec` [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### 5.373.6.11 `const seekdir std::ios_base::end` **[static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

#### 5.373.6.12 `const iostate std::ios_base::eofbit` **[static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_date()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_time()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, and `std::ws()`.

#### 5.373.6.13 `const iostate std::ios_base::failbit` **[static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::seekg()`, and `std::basic_ostream<_CharT, _Traits >::seekp()`.

#### 5.373.6.14 `const fmtflags std::ios_base::fixed` **[static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

## 5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference 1933

---

### 5.373.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 324 of file `ios_base.h`.

### 5.373.6.16 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

### 5.373.6.17 const fmtflags std::ios\_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

### 5.373.6.18 const openmode std::ios\_base::in [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.373.6.19 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file `ios_base.h`.

**5.373.6.20 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

**5.373.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.373.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::seekp()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

**5.373.6.23 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 291 of file `ios_base.h`.

**5.373.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.373 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **1935**

---

**5.373.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios\_base.h.

**5.373.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.373.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.373.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.373.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ofstream.

Definition at line 389 of file ios\_base.h.

**5.373.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.373.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

The documentation for this class was generated from the following file:

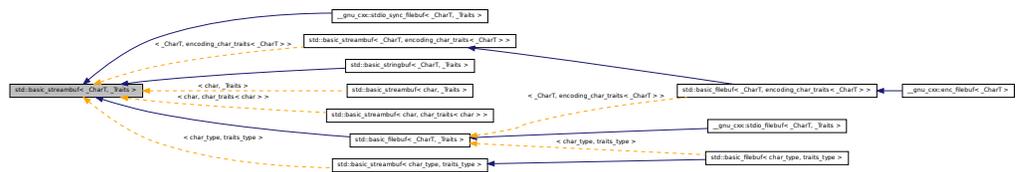
- [sstream](#)

## 5.374 `std::basic_streambuf< _CharT, _Traits >` Class Template Reference

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Inheritance diagram for `std::basic_streambuf< _CharT, _Traits >`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::pos_type` [pos\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)
- typedef `basic_streambuf< char_type, traits_type >` [\\_\\_streambuf\\_type](#)

### Public Member Functions

- [streamsize](#) `in_avail` ()
- `int_type` `sbumpc` ()
- `int_type` `sgetc` ()
- `streamsize` `sgetn` (`char_type` \*`_s`, `streamsize` `__n`)
- `int_type` `snextc` ()
- `int_type` `sputbackc` (`char_type` `__c`)
- `int_type` `sputc` (`char_type` `__c`)
- `streamsize` `sputn` (`const char_type` \*`_s`, `streamsize` `__n`)

- `void stossc ()`
- `int_type sungetc ()`

## Protected Member Functions

- `basic_streambuf ()`
  - `void gbump (int __n)`
  - `virtual void imbue (const locale &)`
  - `virtual int_type overflow (int_type=traits_type::eof())`
  - `virtual int_type pbackfail (int_type=traits_type::eof())`
  - `void pbump (int __n)`
  - `virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
  - `virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
  - `virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)`
  - `void setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)`
  - `void setp (char_type * __pbeg, char_type * __pend)`
  - `virtual streamsize showmanyc ()`
  - `virtual int sync ()`
  - `virtual int_type uflow ()`
  - `virtual int_type underflow ()`
  - `virtual streamsize xsgetn (char_type * __s, streamsize __n)`
  - `virtual streamsize xspn (const char_type * __s, streamsize __n)`
- 
- `char_type * eback () const`
  - `char_type * gptr () const`
  - `char_type * egptr () const`
- 
- `char_type * pbase () const`
  - `char_type * pptr () const`
  - `char_type * epptr () const`

## Friends

- `template<bool _IsMove, typename _CharT2 > __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *, bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`

- class **basic\_ostream**< **char\_type**, **traits\_type** >
- template<typename **\_CharT2** >  
 \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< **\_CharT2** >::\_\_value, [istreambuf\\_iterator](#)< **\_CharT2** > >::\_\_type **find** ([istreambuf\\_iterator](#)< **\_CharT2** >, [istreambuf\\_iterator](#)< **\_CharT2** >, const **\_CharT2** &)
- template<typename **\_CharT2**, typename **\_Traits2**, typename **\_Alloc** >  
[basic\\_istream](#)< **\_CharT2**, **\_Traits2** > & **getline** ([basic\\_istream](#)< **\_CharT2**, **\_Traits2** > &, [basic\\_string](#)< **\_CharT2**, **\_Traits2**, **\_Alloc** > &, **\_CharT2**)
- class **istreambuf\_iterator**< **char\_type**, **traits\_type** >
- template<typename **\_CharT2**, typename **\_Traits2**, typename **\_Alloc** >  
[basic\\_istream](#)< **\_CharT2**, **\_Traits2** > & **operator**>> ([basic\\_istream](#)< **\_CharT2**, **\_Traits2** > &, [basic\\_string](#)< **\_CharT2**, **\_Traits2**, **\_Alloc** > &)
- template<typename **\_CharT2**, typename **\_Traits2** >  
[basic\\_istream](#)< **\_CharT2**, **\_Traits2** > & **operator**>> ([basic\\_istream](#)< **\_CharT2**, **\_Traits2** > &, **\_CharT2** \*)
- class **ostreambuf\_iterator**< **char\_type**, **traits\_type** >
  
- **char\_type** \* **\_M\_in\_beg**
- **char\_type** \* **\_M\_in\_cur**
- **char\_type** \* **\_M\_in\_end**
- **char\_type** \* **\_M\_out\_beg**
- **char\_type** \* **\_M\_out\_cur**
- **char\_type** \* **\_M\_out\_end**
- **locale** **\_M\_buf\_locale**
- virtual **~basic\_streambuf** ()
- **locale** **pubimbue** (const **locale** &\_\_loc)
- **locale** **getloc** () const
- **\_\_streambuf\_type** \* **pubsetbuf** (**char\_type** \*\_\_s, **streamsize** \_\_n)
- **pos\_type** **pubseekoff** (**off\_type** \_\_off, **ios\_base::seekdir** \_\_way, **ios\_base::openmode** \_\_mode=**ios\_base::in**|**ios\_base::out**)
- **pos\_type** **pubseekpos** (**pos\_type** \_\_sp, **ios\_base::openmode** \_\_mode=**ios\_base::in**|**ios\_base::out**)
- int **pubsync** ()

### 5.374.1 Detailed Description

template<typename **\_CharT**, typename **\_Traits**> class std::basic\_streambuf< **\_CharT**, **\_Traits** >

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output. Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
  - The controlled input sequence can be not readable.
  - The controlled output sequence can be not writable.
  - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
  - The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
  - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
  - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and  $xbeg < xnext$  for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 115 of file streambuf.

## 5.374.2 Member Typedef Documentation

**5.374.2.1** `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits >::_streambuf_type`

This is a non-standard type.

Reimplemented in `std::basic_filebuf<_CharT, _Traits >`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT >>`, and `std::basic_filebuf<char_type, traits_type >`.

Definition at line 133 of file streambuf.

**5.374.2.2** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_streambuf<_CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf<_CharT, _Traits >`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT >>`, and `std::basic_filebuf<char_type, traits_type >`.

Definition at line 124 of file streambuf.

**5.374.2.3** `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_streambuf<_CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf<_CharT, _Traits >`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT >>`, and `std::basic_filebuf<char_type, traits_type >`.

Definition at line 126 of file streambuf.

## **5.374 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 1941**

**5.374.2.4** `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 128 of file streambuf.

**5.374.2.5** `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::enc_filebuf< _CharT >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 127 of file streambuf.

**5.374.2.6** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::enc_filebuf< _CharT >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 125 of file streambuf.

### 5.374.3 Constructor & Destructor Documentation

**5.374.3.1** `template<typename _CharT, typename _Traits> virtual  
std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf ( )  
[inline, virtual]`

Destructor deallocates no buffer space.

Definition at line 193 of file streambuf.

**5.374.3.2** `template<typename _CharT, typename _Traits>  
std::basic_streambuf< _CharT, _Traits >::basic_streambuf ( )  
[inline, protected]`

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the [basic\\_streambuf](#) class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 441 of file streambuf.

### 5.374.4 Member Function Documentation

**5.374.4.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eback ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

## 5.374 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 1943

**5.374.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.374.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr ( ) const  
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xspn()`.

**5.374.4.4** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )  
[inline, protected]`

Moving the read position.

**Parameters**

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

**5.374.4.5** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::getloc ( ) const  
[inline]`

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

**5.374.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::gptr ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.374.4.7** `template<typename _CharT, typename _Traits> virtual void  
std::basic_streambuf< _CharT, _Traits >::imbue ( const locale & )  
[inline, protected, virtual]`

Changes translations.

## 5.374 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 1945

### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

### Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 554 of file `streambuf`.

**5.374.4.8** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::in_avail( ) [inline]`

Looking ahead into the stream.

### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file `streambuf`.

**5.374.4.9** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::overflow( int_type =  
traits_type::eof() ) [inline, protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

### Parameters

*c* An additional character to consume.

### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 746 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.374.4.10** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::pbackfail( int_type =  
traits_type::eof() ) [inline, protected, virtual]`

Tries to back up the input sequence.

### Parameters

*c* The character to be inserted back into the sequence.

### Returns

`eof()` on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 702 of file `streambuf`.

## 5.374 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 1947

**5.374.4.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pbase ( ) const  
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

**5.374.4.12** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )  
[inline, protected]`

Moving the write position.

### Parameters

- *n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.374.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr ( ) const  
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.374.4.14** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale  
& __loc ) [inline]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

**5.374.4.15** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file streambuf.

## 5.374 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 1949

**5.374.4.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

**5.374.4.17** `template<typename _CharT, typename _Traits>  
__streambuf_type* std::basic_streambuf<_CharT, _Traits  
>::pubsetbuf ( char_type * __s, streamsize __n ) [inline]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

**5.374.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync ( )  
[inline]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**5.374.4.19** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits >::sbumpc ( ) [inline]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, and `std::istreambuf_iterator<_CharT, _Traits >::operator++()`.

**5.374.4.20** `template<typename _CharT, typename _Traits> virtual  
pos_type std::basic_streambuf<_CharT, _Traits >::seekoff  
( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [inline, protected,  
virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

**5.374.4.21** `template<typename _CharT, typename _Traits> virtual pos_type  
std::basic_streambuf<_CharT, _Traits >::seekpos ( pos_type,  
ios_base::openmode = ios_base::in | ios_base::out ) [inline,  
protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

## 5.374 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 1951

Definition at line 592 of file streambuf.

```
5.374.4.22 template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type, _Traits>* std::basic_streambuf<
_CharT, _Traits>::setbuf( char_type*, streamsize ) [inline,
protected, virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

### Note

Base class version does nothing, returns `this`.

Definition at line 569 of file streambuf.

```
5.374.4.23 template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setg( char_type *
__gbeg, char_type * __gnext, char_type * __gend ) [inline,
protected]
```

Setting the three read area pointers.

### Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

```
5.374.4.24 template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setp( char_type *
__pbeg, char_type * __pend ) [inline, protected]
```

Setting the three write area pointers.

**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.374.4.25** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sgetc( ) [inline]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.374.4.26** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn( char_type * __s,  
streamsize __n ) [inline]`

Entry point for `xsggetn`.

**Parameters**

*s* A buffer area.

*n* A count.

Returns `xsggetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

## 5.374 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 1953

**5.374.4.27** `template<typename _CharT, typename _Traits> virtual streamsize  
std::basic_streambuf<_CharT, _Traits>::showmanyc ( )  
[inline, protected, virtual]`

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 627 of file `streambuf`.

**5.374.4.28** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline]`

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.374.4.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputbackc ( char_type  
__c ) [inline]`

Pushing characters back into the input stream.

#### Parameters

*c* The character to push back.

#### Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.374.4.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

#### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

## 5.374 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 1955

**5.374.4.31** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type  
* __s, streamsize __n ) [inline]`

Entry point for all single-character output functions.

### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xsputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.374.4.32** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stossc ( ) [inline]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.374.4.33** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline]`

Moving backwards in the input stream.

### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.374.4.34** `template<typename _CharT, typename _Traits> virtual int  
std::basic_streambuf< _CharT, _Traits >::sync ( void )  
[inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT >>](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 605 of file streambuf.

**5.374.4.35** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 678 of file streambuf.

**5.374.4.36** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::underflow ( )  
[inline, protected, virtual]`

Fetches more data from the controlled sequence.

## 5.374 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 1957

### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 665 of file `streambuf`.

**5.374.4.37** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsgetn ( char_type *  
__s, streamsize __n ) [protected, virtual]`

Multiple character extraction.

### Parameters

*s* A buffer area.

*n* Maximum number of characters to assign.

### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 45 of file streambuf.tcc.

References `std::min()`.

**5.374.4.38** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsputn( const char_type  
* __s, streamsize __n ) [protected, virtual]`

Multiple character insertion.

#### Parameters

*s* A buffer area.

*n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 79 of file streambuf.tcc.

References `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

## 5.374.5 Member Data Documentation

**5.374.5.1** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale  
[protected]`

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

## 5.374 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference 1959

**5.374.5.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_beg  
[protected]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

**5.374.5.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.374.5.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.374.5.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

**5.374.5.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

**5.374.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

## 5.375 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference ¶961

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

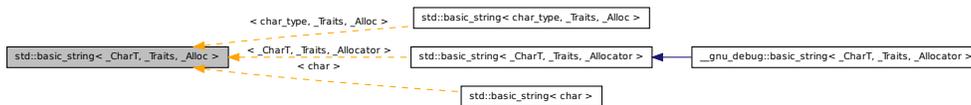
The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

## 5.375 `std::basic_string<_CharT, _Traits, _Alloc >` Class Template Reference

Managing sequences of characters and character-like objects.

Inheritance diagram for `std::basic_string<_CharT, _Traits, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- [basic\\_string](#) ()
- [basic\\_string](#) (const [\\_Alloc](#) &\_\_a)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) (size\_type \_\_n, [CharT](#) \_\_c, const [\\_Alloc](#) &\_\_a=[Alloc](#)())
- [basic\\_string](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const [\\_Alloc](#) &\_\_a)
- [basic\\_string](#) ([initializer\\_list](#)< [CharT](#) > \_\_l, const [\\_Alloc](#) &\_\_a=[Alloc](#)())
- [template](#)<class [\\_InputIterator](#) >  
[basic\\_string](#) ([\\_InputIterator](#) \_\_beg, [\\_InputIterator](#) \_\_end, const [\\_Alloc](#) &\_\_a=[\\_Alloc](#)())
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [CharT](#) \*\_\_s, size\_type \_\_n, const [\\_Alloc](#) &\_\_a=[Alloc](#)())
- [basic\\_string](#) (const [CharT](#) \*\_\_s, const [\\_Alloc](#) &\_\_a=[Alloc](#)())
- [~basic\\_string](#) ()
- [template](#)<typename [\\_InIterator](#) >  
[CharT](#) \* [\\_S\\_construct](#) ([\\_InIterator](#) \_\_beg, [\\_InIterator](#) \_\_end, const [\\_Alloc](#) &\_\_a, [forward\\_iterator\\_tag](#))
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & [append](#) (const [CharT](#) \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [append](#) (const [CharT](#) \*\_\_s)
- [basic\\_string](#) & [append](#) (size\_type \_\_n, [CharT](#) \_\_c)
- [basic\\_string](#) & [append](#) ([initializer\\_list](#)< [CharT](#) > \_\_l)
- [template](#)<class [\\_InputIterator](#) >  
[basic\\_string](#) & [append](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [basic\\_string](#) & [assign](#) ([initializer\\_list](#)< [CharT](#) > \_\_l)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const [CharT](#) \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (size\_type \_\_n, [CharT](#) \_\_c)
- [basic\\_string](#) & [assign](#) (const [CharT](#) \*\_\_s)
- [template](#)<class [\\_InputIterator](#) >  
[basic\\_string](#) & [assign](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [const\\_reference](#) [at](#) (size\_type \_\_n) const
- [reference](#) [at](#) (size\_type \_\_n)
- [iterator](#) [begin](#) ()
- [const\\_iterator](#) [begin](#) () const

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

- const `_CharT * c_str` () const
- size\_type `capacity` () const
- const\_iterator `cbegin` () const
- const\_iterator `cend` () const
- void `clear` ()
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT * __s`, size\_type \_\_n2) const
- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string & __str`, size\_type \_\_pos2, size\_type \_\_n2) const
- int `compare` (const `basic_string & __str`) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string & __str`) const
- int `compare` (const `_CharT * __s`) const
- size\_type `copy` (`_CharT * __s`, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator `crbegin` () const
- const\_reverse\_iterator `crend` () const
- const `_CharT * data` () const
- bool `empty` () const
- iterator `end` ()
- const\_iterator `end` () const
- `basic_string & erase` (size\_type \_\_pos=0, size\_type \_\_n=npos)
- iterator `erase` (iterator \_\_position)
- iterator `erase` (iterator \_\_first, iterator \_\_last)
- size\_type `find` (`_CharT __c`, size\_type \_\_pos=0) const
- size\_type `find` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find` (const `basic_string & __str`, size\_type \_\_pos=0) const
- size\_type `find` (const `_CharT * __s`, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (`_CharT __c`, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (const `basic_string & __str`, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `_CharT * __s`, size\_type \_\_pos=0) const
- size\_type `find_first_of` (const `basic_string & __str`, size\_type \_\_pos=0) const
- size\_type `find_first_of` (const `_CharT * __s`, size\_type \_\_pos=0) const
- size\_type `find_first_of` (`_CharT __c`, size\_type \_\_pos=0) const
- size\_type `find_first_of` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_last_not_of` (const `_CharT * __s`, size\_type \_\_pos=npo) const
- size\_type `find_last_not_of` (const `_CharT * __s`, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_last_not_of` (const `basic_string & __str`, size\_type \_\_pos=npo) const
- size\_type `find_last_not_of` (`_CharT __c`, size\_type \_\_pos=npo) const

- size\_type [find\\_last\\_of](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const
- size\_type [find\\_last\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_of](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const
- size\_type [find\\_last\\_of](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const
- allocator\_type [get\\_allocator](#) () const
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator >  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, [initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s)
- iterator [insert](#) (iterator \_\_p, \_CharT \_\_c)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- size\_type [length](#) () const
- size\_type [max\\_size](#) () const
- [basic\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- reference [operator\[ \]](#) (size\_type \_\_pos)
- const\_reference [operator\[ \]](#) (size\_type \_\_pos) const
- void [push\\_back](#) (\_CharT \_\_c)
- [reverse\\_iterator](#) [rbegin](#) ()
- [const\\_reverse\\_iterator](#) [rbegin](#) () const
- [reverse\\_iterator](#) [rend](#) ()
- [const\\_reverse\\_iterator](#) [rend](#) () const
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [initializer\\_list](#)< \_CharT > \_\_l)

## 5.375 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference

- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str)
- `basic_string` & `replace` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \*\_\_s, size\_type \_\_n2)
- `template<class _InputIterator >`  
`basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const `basic_string` &\_\_str)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const `_CharT` \*\_\_s)
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, `_CharT` \_\_c)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const `_CharT` \*\_\_k1, const `_CharT` \*\_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, `_CharT` \_\_c)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, `_CharT` \*\_\_k1, `_CharT` \*\_\_k2)
- void `reserve` (size\_type \_\_res\_arg=0)
- void `resize` (size\_type \_\_n)
- void `resize` (size\_type \_\_n, `_CharT` \_\_c)
- size\_type `rfind` (`_CharT` \_\_c, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `rfind` (const `_CharT` \*\_\_s, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (const `basic_string` &\_\_str, size\_type \_\_pos=`npos`) const
- void `shrink_to_fit` ()
- size\_type `size` () const
- `basic_string` `substr` (size\_type \_\_pos=0, size\_type \_\_n=`npos`) const
- void `swap` (`basic_string` &\_\_s)

### Static Public Attributes

- static const size\_type `npos`

### 5.375.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_string<_CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects. Meets the requirements of a `container`, a `reversible container`, and a `sequence`. Of the `optional sequence requirements`, only `push_back`, `at`, and array access are supported.

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Documentation? What's that? Nathan Myers <[ncm@cantrip.org](mailto:ncm@cantrip.org)>.

A string looks like this:

```

[basic_string<char_type>]
_M_dataplus
_M_p ----->
                                     [_Rep]
                                     _M_length
                                     _M_capacity
                                     _M_refcount
                                     unnamed array of char_type

```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 105 of file `basic_string.h`.

**5.375.2 Constructor & Destructor Documentation**

**5.375.2.1** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic_string<_CharT, _Traits, _Alloc>::basic_string( )`  
`[inline]`

Default constructor creates an empty string.

Definition at line 423 of file `basic_string.h`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1967

**5.375.2.2** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc >::basic_string ( const  
_Alloc & __a ) [explicit]`

Construct an empty string using allocator *a*.

Definition at line 178 of file basic\_string.tcc.

**5.375.2.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc >::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc > & __str )`

Construct string with copy of value of *str*.

### Parameters

*str* Source string.

Definition at line 170 of file basic\_string.tcc.

**5.375.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc >::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc > & __str, size_type __pos,  
size_type __n = npos )`

Construct string as copy of a substring.

### Parameters

*str* Source string.

*pos* Index of first character to copy from.

*n* Number of characters to copy (default remainder).

Definition at line 184 of file basic\_string.tcc.

**5.375.2.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc >::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc > & __str, size_type __pos,  
size_type __n, const _Alloc & __a )`

Construct string as copy of a substring.

### Parameters

*str* Source string.

*pos* Index of first character to copy from.  
*n* Number of characters to copy.  
*a* Allocator to use.

Definition at line 194 of file basic\_string.tcc.

**5.375.2.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
 std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
 _CharT * __s, size_type __n, const _Alloc & __a = _Alloc() )`

Construct string initialized by a character array.

#### Parameters

*s* Source character array.  
*n* Number of characters to copy.  
*a* Allocator to use (default is default allocator).

NB: *s* must have at least *n* characters, `'\0'` has no special meaning.

Definition at line 206 of file basic\_string.tcc.

**5.375.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
 std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
 _CharT * __s, const _Alloc & __a = _Alloc() )`

Construct string as copy of a C string.

#### Parameters

*s* Source C string.  
*a* Allocator to use (default is default allocator).

Definition at line 213 of file basic\_string.tcc.

**5.375.2.8** `template<typename _CharT, typename _Traits, typename _Alloc>  
 std::basic_string<_CharT, _Traits, _Alloc>::basic_string (   
 size_type __n, _CharT __c, const _Alloc & __a = _Alloc() )`

Construct string as multiple characters.

#### Parameters

*n* Number of characters.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1969

*c* Character to use.

*a* Allocator to use (default is default allocator).

Definition at line 220 of file basic\_string.tcc.

```
5.375.2.9 template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
    basic_string<_CharT, _Traits, _Alloc > && __str ) [inline]
```

Move construct string.

### Parameters

*str* Source string.

The newly-created string contains the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 493 of file basic\_string.h.

```
5.375.2.10 template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
    initializer_list<_CharT > __l, const _Alloc & __a = _Alloc() )
```

Construct string from an initializer list.

### Parameters

*l* [std::initializer\\_list](#) of characters.

*a* Allocator to use (default is default allocator).

Definition at line 235 of file basic\_string.tcc.

```
5.375.2.11 template<typename _CharT, typename _Traits, typename
_Alloc> template<typename _InputIterator > std::basic_string<
_CharT, _Traits, _Alloc >::basic_string ( _InputIterator __beg,
_InputIterator __end, const _Alloc & __a = _Alloc() )
```

Construct string as copy of a range.

### Parameters

*beg* Start of range.

*end* End of range.

*a* Allocator to use (default is default allocator).

Definition at line 228 of file basic\_string.tcc.

**5.375.2.12** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ( )  
[inline]`

Destroy the string instance.

Definition at line 524 of file basic\_string.h.

### 5.375.3 Member Function Documentation

**5.375.3.1** `template<typename _CharT , typename _Traits , typename _Alloc  
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<  
_CharT, _Traits, _Alloc >::append ( const basic_string< _CharT,  
_Traits, _Alloc > & __str, size_type __pos, size_type __n )`

Append a substring.

#### Parameters

*str* The string to append.

*pos* Index of the first character of *str* to append.

*n* The number of characters to append.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

Definition at line 342 of file basic\_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 1971

**5.375.3.2** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::append ( const _CharT * __s, size_type  
__n )`

Append a C substring.

### Parameters

- s* The C string to append.
- n* The number of characters to append.

### Returns

Reference to this string.

Definition at line 298 of file basic\_string.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.375.3.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append  
( const _CharT * __s ) [inline]`

Append a C string.

### Parameters

- s* The C string to append.

### Returns

Reference to this string.

Definition at line 954 of file basic\_string.h.

**5.375.3.4** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::append ( size_type __n, _CharT __c )`

Append multiple characters.

### Parameters

- n* The number of characters to append.

*c* The character to use.

**Returns**

Reference to this string.

Appends *n* copies of *c* to this string.

Definition at line 281 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**5.375.3.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::append  
( initializer_list<_CharT > __l ) [inline]`

Append an [initializer\\_list](#) of characters.

**Parameters**

*l* The [initializer\\_list](#) of characters to append.

**Returns**

Reference to this string.

Definition at line 978 of file `basic_string.h`.

Referenced by `std::basic_string<char >::append()`.

**5.375.3.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator > basic_string& std::basic_string<  
_CharT, _Traits, _Alloc >::append ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Append a range of characters.

**Parameters**

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

**Returns**

Reference to this string.

### 5.375 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 1973

Appends characters in the range [first,last) to this string.

Definition at line 992 of file basic\_string.h.

```
5.375.3.7 template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append ( const basic_string< _CharT,
_Traits, _Alloc > & __str )
```

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

Definition at line 325 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::capacity(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::size().

Referenced by std::collate< \_CharT >::do\_transform(), std::operator+(), std::operator>>(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::resize().

```
5.375.3.8 template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::assign ( const basic_string< _CharT,
_Traits, _Alloc > & __str )
```

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

Definition at line 243 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::get\_allocator().

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str().

**5.375.3.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign (  
basic_string<_CharT, _Traits, _Alloc > && __str ) [inline]`

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1027 of file basic\_string.h.

**5.375.3.10** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign  
( const basic_string<_CharT, _Traits, _Alloc > & __str, size_type  
__pos, size_type __n ) [inline]`

Set value to a substring of a string.

#### Parameters

*str* The string to use.

*pos* Index of the first character of *str*.

*n* Number of characters to use.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1047 of file basic\_string.h.

Referenced by `std::basic_string< char >::assign()`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference

**5.375.3.11** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::assign ( const _CharT * __s, size_type  
__n )`

Set value to a C substring.

### Parameters

- s* The C string to use.
- n* Number of characters to use.

### Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

Definition at line 259 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc>::size().

**5.375.3.12** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign  
( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

### Parameters

- n* Length of the resulting string.
- c* The character to use.

### Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1091 of file basic\_string.h.

**5.375.3.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator > basic_string& std::basic_string<  
_CharT, _Traits, _Alloc>::assign ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Set value to a range of characters.

**Parameters**

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1104 of file basic\_string.h.

```
5.375.3.14 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
( initializer_list<_CharT > __l ) [inline]
```

Set value to an [initializer\\_list](#) of characters.

**Parameters**

*l* The [initializer\\_list](#) of characters to assign.

**Returns**

Reference to this string.

Definition at line 1114 of file basic\_string.h.

Referenced by std::basic\_string< char >::assign().

```
5.375.3.15 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
( const _CharT * __s ) [inline]
```

Set value to contents of a C string.

**Parameters**

*s* The C string to use.

**Returns**

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1075 of file basic\_string.h.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1977

**5.375.3.16** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reference std::basic_string<_CharT, _Traits, _Alloc >::at (  
size_type __n ) const [inline]`

Provides access to the data contained in the string.

### Parameters

*n* The index of the character to access.

### Returns

Read-only (const) reference to the character.

### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws *out\_of\_range* if the check fails.

Definition at line 847 of file basic\_string.h.

**5.375.3.17** `template<typename _CharT, typename _Traits, typename _Alloc>  
reference std::basic_string<_CharT, _Traits, _Alloc >::at (  
size_type __n ) [inline]`

Provides access to the data contained in the string.

### Parameters

*n* The index of the character to access.

### Returns

Read/write reference to the character.

### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws *out\_of\_range* if the check fails. Success results in unsharing the string.

Definition at line 866 of file basic\_string.h.

**5.375.3.18** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( )  
[inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 591 of file basic\_string.h.

**5.375.3.19** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 602 of file basic\_string.h.

**5.375.3.20** `template<typename _CharT, typename _Traits, typename _Alloc>  
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str ( ) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1723 of file basic\_string.h.

Referenced by `std::collate< _CharT >::do_compare()`, `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::collate< _CharT >::do_transform()`, and `std::basic_filebuf< char_type, traits_type >::open()`.

**5.375.3.21** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity ( ) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 759 of file basic\_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

**5.375.3.22** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc >::cbegin  
( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 666 of file basic\_string.h.

**5.375.3.23** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 674 of file basic\_string.h.

**5.375.3.24** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string<_CharT, _Traits, _Alloc >::clear ( ) [inline]`

Erases the string, making it empty.

Definition at line 786 of file basic\_string.h.

Referenced by std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc >::setbuf().

**5.375.3.25** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc >::compare ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) const`

Compare substring against a character array.

### Parameters

- pos1* Index of first character of substring.
- n1* Number of characters in substring.
- s* character array to compare against.
- n2* Number of characters of *s*.

### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer  $< 0$  if this substring is ordered before the string from *s*,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

Definition at line 980 of file `basic_string.tcc`.

References `std::min()`.

```
5.375.3.26  template<typename _CharT , typename _Traits , typename _Alloc
            > int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1, size_type __n1, const basic_string< _CharT,
            _Traits, _Alloc > & __str, size_type __pos2, size_type __n2 ) const
```

Compare substring to a substring.

#### Parameters

*pos1* Index of first character of substring.

*n1* Number of characters in substring.

*str* String to compare against.

*pos2* Index of first character of substring of *str*.

*n2* Number of characters in substring of *str*.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer  $< 0$  if this substring is ordered before the substring of *str*,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 931 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::min()`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1981

**5.375.3.27** `template<typename _CharT, typename _Traits, typename _Alloc>  
int std::basic_string<_CharT, _Traits, _Alloc >::compare (  
const basic_string<_CharT, _Traits, _Alloc > & __str ) const  
[inline]`

Compare to a string.

### Parameters

*str* String to compare against.

### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2130 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==(())`, `std::operator>()`, and `std::operator>=()`.

**5.375.3.28** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc >::compare (  
size_type __pos, size_type __n, const basic_string<_CharT,  
_Traits, _Alloc > & __str ) const`

Compare substring to a string.

### Parameters

*pos* Index of first character of substring.

*n* Number of characters in substring.

*str* String to compare against.

### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *str*, 0 if their values are equivalent, or > 0

if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 916 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**5.375.3.29** `template<typename _CharT, typename _Traits, typename _Alloc >  
int std::basic_string<_CharT, _Traits, _Alloc >::compare ( const  
_CharT * __s ) const`

Compare to a C string.

#### Parameters

*s* C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 949 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::length()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

**5.375.3.30** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc >::compare (   
size_type __pos, size_type __n1, const _CharT * __s ) const`

Compare substring to a C string.

#### Parameters

*pos* Index of first character of substring.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1983

*n1* Number of characters in substring.

*s* C string to compare against.

### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *r1en* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 964 of file `basic_string.tcc`.

References `std::min()`.

**5.375.3.31** `template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::copy ( _CharT * __s, size_type __n, size_type __pos = 0 ) const`

Copy substring into C string.

### Parameters

*s* C string to copy value into.

*n* Number of characters to copy.

*pos* Index of first character to copy.

### Returns

Number of characters actually copied

### Exceptions

[\*std::out\\_of\\_range\*](#) If `pos > size()`.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than `size()`, `out_of_range` is thrown.

Definition at line 723 of file `basic_string.tcc`.

**5.375.3.32** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc  
>::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 683 of file basic\_string.h.

**5.375.3.33** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc  
>::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 692 of file basic\_string.h.

**5.375.3.34** `template<typename _CharT, typename _Traits, typename _Alloc>  
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::data ( ) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1733 of file basic\_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, `std::basic_string< char >::find()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< char >::find_first_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::rfind()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

**5.375.3.35** `template<typename _CharT, typename _Traits, typename _Alloc>  
bool std::basic_string< _CharT, _Traits, _Alloc >::empty ( ) const [inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 794 of file basic\_string.h.

Referenced by `std::operator>>()`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

**5.375.3.36** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string<_CharT, _Traits, _Alloc >::end ( )  
[inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 610 of file basic\_string.h.

**5.375.3.37** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 621 of file basic\_string.h.

**5.375.3.38** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::erase ( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

### Parameters

*pos* Index of first character to remove (default 0).

*n* Number of characters to remove (default remainder).

### Returns

Reference to this string.

### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are < *n* characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, [\*out\\_of\\_range\*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1304 of file basic\_string.h.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc >::resize()`.

**5.375.3.39** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (  
iterator __position ) [inline]`

Remove one character.

#### Parameters

*position* Iterator referencing the character to remove.

#### Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1320 of file basic\_string.h.

**5.375.3.40** `template<typename _CharT , typename _Traits , typename  
_Alloc > basic_string< _CharT, _Traits, _Alloc >::iterator  
std::basic_string< _CharT, _Traits, _Alloc >::erase ( iterator  
__first, iterator __last )`

Remove a range of characters.

#### Parameters

*first* Iterator referencing the first character to remove.

*last* Iterator referencing the end of the range.

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 391 of file basic\_string.tcc.

**5.375.3.41** `template<typename _CharT, typename _Traits , typename  
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type  
std::basic_string< _CharT, _Traits, _Alloc >::find ( const _CharT *  
__s, size_type __pos, size_type __n ) const`

Find position of a C substring.

### 5.375 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 1987

#### Parameters

*s* C string to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

#### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 737 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::size().

Referenced by std::basic\_string< \_CharT, \_Traits, \_Alloc >::find(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_first\_of().

```
5.375.3.42 template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const
basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos =
0 ) const [inline]
```

Find position of a string.

#### Parameters

*str* String to locate.

*pos* Index of character to search from (default 0).

#### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1768 of file basic\_string.h.

Referenced by std::basic\_string< char >::find().

```
5.375.3.43 template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find ( const
_CharT * __s, size_type __pos = 0 ) const [inline]
```

Find position of a C string.

**Parameters**

*s* C string to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1782 of file `basic_string.h`.

```
5.375.3.44 template<typename _CharT, typename _Traits , typename
    _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
    std::basic_string< _CharT, _Traits, _Alloc >::find ( _CharT __c,
    size_type __pos = 0 ) const
```

Find position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 760 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::find()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.375.3.45 template<typename _CharT, typename _Traits , typename
    _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
    std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    _CharT __c, size_type __pos = 0 ) const
```

Find position of a different character.

**Parameters**

*c* Character to avoid.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1989

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 864 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

```
5.375.3.46 template<typename _CharT, typename _Traits, typename
    _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc
    >::find_first_not_of( const basic_string<_CharT, _Traits, _Alloc
    > & __str, size_type __pos = 0 ) const [inline]
```

Find position of a character not in string.

### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1992 of file `basic_string.h`.

Referenced by `std::basic_string<char >::find_first_not_of()`.

```
5.375.3.47 template<typename _CharT, typename _Traits, typename
    _Alloc> basic_string<_CharT, _Traits, _Alloc >::size_type
    std::basic_string<_CharT, _Traits, _Alloc >::find_first_not_of (
    const _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character not in C substring.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from.

*n* Number of characters from *s* to consider.

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 852 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

```
5.375.3.48 template<typename _CharT, typename _Traits, typename
             _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
             >::find_first_not_of( const _CharT * __s, size_type __pos = 0 )
             const [inline]
```

Find position of a character not in C string.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2021 of file `basic_string.h`.

```
5.375.3.49 template<typename _CharT, typename _Traits, typename _Alloc>
             size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
             ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
             __pos = 0 ) const [inline]
```

Find position of a character of string.

### Parameters

*str* String containing characters to locate.

*pos* Index of character to search from (default 0).

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1991

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1870 of file `basic_string.h`.

Referenced by `std::basic_string<char >::find_first_of()`.

```
5.375.3.50 template<typename _CharT, typename _Traits, typename
    _Alloc > basic_string<_CharT, _Traits, _Alloc >::size_type
    std::basic_string<_CharT, _Traits, _Alloc >::find_first_of ( const
    _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character of C substring.

### Parameters

*s* String containing characters to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 816 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::find()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

```
5.375.3.51 template<typename _CharT, typename _Traits, typename _Alloc>
    size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_of
    ( const _CharT * __s, size_type __pos = 0 ) const [inline]
```

Find position of a character of C string.

### Parameters

*s* String containing characters to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1898 of file `basic_string.h`.

```
5.375.3.52 template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
( _CharT __c, size_type __pos = 0 ) const [inline]
```

Find position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1917 of file `basic_string.h`.

```
5.375.3.53 template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
_CharT __c, size_type __pos = npos ) const
```

Find last position of a different character.

**Parameters**

*c* Character to avoid.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

### 5.375 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 1993

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 896 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.375.3.54 template<typename _CharT, typename _Traits, typename
    _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
    >::find_last_not_of ( const basic_string< _CharT, _Traits, _Alloc >
    & __str, size_type __pos = npos ) const [inline]
```

Find last position of a character not in string.

#### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2051 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

```
5.375.3.55 template<typename _CharT, typename _Traits , typename
    _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
    std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const _CharT * __s, size_type __pos, size_type __n ) const
```

Find last position of a character not in C substring.

#### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from.

*n* Number of characters from *s* to consider.

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 875 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

```
5.375.3.56 template<typename _CharT, typename _Traits, typename
                _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc
                >::find_last_not_of ( const _CharT * __s, size_type __pos = npow )
                const [inline]
```

Find last position of a character not in C string.

#### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2080 of file `basic_string.h`.

```
5.375.3.57 template<typename _CharT, typename _Traits, typename _Alloc>
                size_type std::basic_string<_CharT, _Traits, _Alloc >::find_last_of
                ( const basic_string<_CharT, _Traits, _Alloc > & __str, size_type
                __pos = npow ) const [inline]
```

Find last position of a character of string.

#### Parameters

*str* String containing characters to locate.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1995

Definition at line 1931 of file basic\_string.h.

Referenced by std::basic\_string<char >::find\_last\_of().

```
5.375.3.58 template<typename _CharT, typename _Traits, typename
_Alloc > basic_string<_CharT, _Traits, _Alloc >::size_type
std::basic_string<_CharT, _Traits, _Alloc >::find_last_of ( const
_CharT * __s, size_type __pos, size_type __n ) const
```

Find last position of a character of C substring.

### Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 831 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc >::size().

```
5.375.3.59 template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_last_of
( const _CharT * __s, size_type __pos = npow ) const [inline]
```

Find last position of a character of C string.

### Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1959 of file basic\_string.h.

**5.375.3.60** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of  
( _CharT __c, size_type __pos = npos ) const [inline]`

Find last position of a character.

#### Parameters

*c* Character to locate.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1978 of file `basic_string.h`.

**5.375.3.61** `template<typename _CharT, typename _Traits, typename _Alloc>  
allocator_type std::basic_string< _CharT, _Traits, _Alloc  
>::get_allocator ( ) const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1740 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

**5.375.3.62** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator  
__p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

*p* Iterator referencing location in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 1997

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts  $n$  copies of character  $c$  starting at the position referenced by iterator  $p$ . If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1131 of file `basic_string.h`.

**5.375.3.63** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator > void std::basic_string< _CharT,  
_Traits, _Alloc >::insert ( iterator __p, _InputIterator __beg,  
_InputIterator __end ) [inline]`

Insert a range of characters.

### Parameters

*p* Iterator referencing location in string to insert at.

*beg* Start of range.

*end* End of range.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1147 of file `basic_string.h`.

**5.375.3.64** `template<typename _CharT, typename _Traits , typename _Alloc  
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<  
_CharT, _Traits, _Alloc >::insert ( size_type __pos, const _CharT  
* __s, size_type __n )`

Insert a C substring.

### Parameters

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

*n* The number of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If `pos` is beyond the end of this string.

Inserts the first `n` characters of `s` starting at `pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 360 of file `basic_string.tcc`.

**5.375.3.65** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (  
size_type __pos, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

**Parameters**

`pos` Index in string to insert at.

`n` Number of characters to insert

`c` The character to insert.

**Returns**

Reference to this string.

**Exceptions**

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If `pos` is beyond the end of this string.

Inserts `n` copies of character `c` starting at index `pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1263 of file `basic_string.h`.

**5.375.3.66** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string<_CharT, _Traits, _Alloc >::insert ( iterator  
__p, initializer_list<_CharT > __l ) [inline]`

Insert an `initializer_list` of characters.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference1999

### Parameters

- p* Iterator referencing location in string to insert at.
- l* The [initializer\\_list](#) of characters to insert.

### Exceptions

- [std::length\\_error](#) If new length exceeds [max\\_size\(\)](#).

Definition at line 1158 of file `basic_string.h`.

```
5.375.3.67  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
            size_type __pos1, const basic_string<_CharT, _Traits, _Alloc > &
            __str ) [inline]
```

Insert value of a string.

### Parameters

- pos1* Iterator referencing location in string to insert at.
- str* The string to insert.

### Returns

Reference to this string.

### Exceptions

- [std::length\\_error](#) If new length exceeds [max\\_size\(\)](#).

Inserts value of *str* starting at *pos1*. If adding characters causes the length to exceed [max\\_size\(\)](#), [length\\_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1177 of file `basic_string.h`.

Referenced by `std::basic_string<char >::insert()`.

```
5.375.3.68  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
            size_type __pos1, const basic_string<_CharT, _Traits, _Alloc > &
            __str, size_type __pos2, size_type __n ) [inline]
```

Insert a substring.

**Parameters**

*pos1* Iterator referencing location in string to insert at.  
*str* The string to insert.  
*pos2* Start of characters in *str* to insert.  
*n* Number of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.  
[\*std::out\\_of\\_range\*](#) If *pos1* > `size()` or *pos2* > `str.size()`.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, [`length\_error`](#) is thrown. If *pos1* is beyond the end of this string or *pos2* is beyond the end of *str*, [`out\_of\_range`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1199 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

**5.375.3.69** `template<typename _CharT, typename _Traits, typename _Alloc>  
 basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (  
 size_type __pos, const _CharT * __s ) [inline]`

Insert a C string.

**Parameters**

*pos* Iterator referencing location in string to insert at.  
*s* The C string to insert.

**Returns**

Reference to this string.

**Exceptions**

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.  
[\*std::out\\_of\\_range\*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, [`length\_error`](#) is thrown. If *pos* is beyond `end()`, [`out\_of\_range`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1240 of file `basic_string.h`.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

**5.375.3.70** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string<_CharT, _Traits, _Alloc >::insert (  
iterator __p, _CharT __c ) [inline]`

Insert one character.

### Parameters

- p* Iterator referencing position in string to insert at.
- c* The character to insert.

### Returns

Iterator referencing newly inserted char.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, *length\_error* is thrown. If *p* is beyond end of string, *out\_of\_range* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1280 of file `basic_string.h`.

**5.375.3.71** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc >::length ( )  
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 707 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::collate<_CharT >::do_compare()`, and `std::collate<_CharT >::do_transform()`.

**5.375.3.72** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc >::max_size (  
) const [inline]`

Returns the `size()` of the largest possible string.

Definition at line 712 of file `basic_string.h`.

Referenced by `std::getline()`, and `std::operator>>()`.

**5.375.3.73** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator+=( const basic_string< _CharT, _Traits, _Alloc > &  
_str ) [inline]`

Append a string to this string.

**Parameters**

*str* The string to append.

**Returns**

Reference to this string.

Definition at line 881 of file basic\_string.h.

**5.375.3.74** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator+=( const _CharT * _s ) [inline]`

Append a C string.

**Parameters**

*s* The C string to append.

**Returns**

Reference to this string.

Definition at line 890 of file basic\_string.h.

**5.375.3.75** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator+=( _CharT _c ) [inline]`

Append a character.

**Parameters**

*c* The character to append.

**Returns**

Reference to this string.

Definition at line 899 of file basic\_string.h.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2003

**5.375.3.76** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator+=( initializer_list<_CharT > __l ) [inline]`

Append an [initializer\\_list](#) of characters.

### Parameters

*l* The [initializer\\_list](#) of characters to be appended.

### Returns

Reference to this string.

Definition at line 912 of file `basic_string.h`.

**5.375.3.77** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator=( basic_string<_CharT, _Traits, _Alloc > && __str  
) [inline]`

Move assign the value of *str* to this string.

### Parameters

*str* Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 566 of file `basic_string.h`.

**5.375.3.78** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator=( initializer_list<_CharT > __l ) [inline]`

Set value to string constructed from initializer list.

### Parameters

*l* `std::initializer_list`.

Definition at line 578 of file `basic_string.h`.

**5.375.3.79** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator= ( const _CharT * __s ) [inline]`

Copy contents of *s* into this string.

#### Parameters

*s* Source null-terminated string.

Definition at line 540 of file basic\_string.h.

**5.375.3.80** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator= ( const basic_string< _CharT, _Traits, _Alloc > &  
__str ) [inline]`

Assign the value of *str* to this string.

#### Parameters

*str* Source string.

Definition at line 532 of file basic\_string.h.

**5.375.3.81** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc  
>::operator= ( _CharT __c ) [inline]`

Set value to string of length 1.

#### Parameters

*c* Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 551 of file basic\_string.h.

**5.375.3.82** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reference std::basic_string< _CharT, _Traits, _Alloc  
>::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

### Parameters

*pos* The index of the character to access.

### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 809 of file `basic_string.h`.

```
5.375.3.83 template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc >::operator[] (
size_type __pos ) [inline]
```

Subscript access to the data contained in the string.

### Parameters

*pos* The index of the character to access.

### Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 826 of file `basic_string.h`.

```
5.375.3.84 template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::push_back (
_CharT __c ) [inline]
```

Append a single character.

### Parameters

*c* Character to append.

Definition at line 1000 of file `basic_string.h`.

Referenced by `std::collate<_CharT >::do_transform()`, and `std::operator>>()`.

**5.375.3.85** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc  
>::rbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 639 of file `basic_string.h`.

**5.375.3.86** `template<typename _CharT, typename _Traits, typename _Alloc>  
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc  
>::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 630 of file `basic_string.h`.

**5.375.3.87** `template<typename _CharT, typename _Traits, typename _Alloc>  
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend  
( ) [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 648 of file `basic_string.h`.

**5.375.3.88** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc  
>::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 657 of file `basic_string.h`.

**5.375.3.89** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace  
( iterator __i1, iterator __i2, initializer_list< _CharT > __l )  
[inline]`

Replace range of characters with [initializer\\_list](#).

#### Parameters

*i1* Iterator referencing start of range to replace.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2007

*i2* Iterator referencing end of range to replace.

*l* The [initializer\\_list](#) of characters to insert.

### Returns

Reference to this string.

### Exceptions

[std::length\\_error](#) If new length exceeds [max\\_size\(\)](#).

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds [max\\_size\(\)](#), [length\\_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1616 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

**5.375.3.90** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace  
( iterator __i1, iterator __i2, const basic_string<_CharT, _Traits,  
_Alloc > & __str ) [inline]`

Replace range of characters with string.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*str* String value to insert.

### Returns

Reference to this string.

### Exceptions

[std::length\\_error](#) If new length exceeds [max\\_size\(\)](#).

Removes the characters in the range [i1,i2). In place, the value of *str* is inserted. If the length of result exceeds [max\\_size\(\)](#), [length\\_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1465 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

```

5.375.3.91 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace
( size_type __pos, size_type __n1, size_type __n2, _CharT __c )
[inline]

```

Replace characters with multiple characters.

#### Parameters

- pos* Index of first character to replace.
- n1* Number of characters to be replaced.
- n2* Number of characters to insert.
- c* Character to insert.

#### Returns

Reference to this string.

#### Exceptions

- [\*std::out\\_of\\_range\*](#) If *pos* > *size()*.
- [\*std::length\\_error\*](#) If new length exceeds *max\_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, [\*out\\_of\\_range\*](#) is thrown. If the length of result exceeds [\*max\\_size\(\)\*](#), [\*length\\_error\*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1447 of file `basic_string.h`.

```

5.375.3.92 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace
( iterator __i1, iterator __i2, size_type __n, _CharT __c )
[inline]

```

Replace range of characters with multiple characters.

#### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- n* Number of characters to insert.
- c* Character to insert.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference 2009

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1525 of file `basic_string.h`.

**5.375.3.93** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator > basic_string& std::basic_string<  
_CharT, _Traits, _Alloc >::replace ( iterator __i1, iterator __i2,  
_InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*k1* Iterator referencing start of range to insert.

*k2* Iterator referencing end of range to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1548 of file `basic_string.h`.

```

5.375.3.94 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace
( size_type __pos1, size_type __n1, const basic_string<_CharT,
_Traits, _Alloc > & __str, size_type __pos2, size_type __n2 )
[inline]

```

Replace characters with value from another string.

#### Parameters

- pos1* Index of first character to replace.
- n1* Number of characters to be replaced.
- str* String to insert.
- pos2* Index of first character of str to use.
- n2* Number of characters from str to use.

#### Returns

Reference to this string.

#### Exceptions

- [\*std::out\\_of\\_range\*](#) If *pos1* > *size()* or *pos2* > *str.size()*.
- [\*std::length\\_error\*](#) If new length exceeds *max\_size()*.

Removes the characters in the range [*pos1*, *pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, [\*out\\_of\\_range\*](#) is thrown. If the length of the result exceeds [\*max\\_size\(\)\*](#), [\*length\\_error\*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1381 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

```

5.375.3.95 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace
( size_type __pos, size_type __n1, const _CharT * __s )
[inline]

```

Replace characters with value of a C string.

#### Parameters

- pos* Index of first character to replace.
- n1* Number of characters to be replaced.

## 5.375 std::basic\_string<\_CharT, \_Traits, \_Alloc > Class Template Reference

*s* C string to insert.

### Returns

Reference to this string.

### Exceptions

*std::out\_of\_range* If *pos* > *size()*.

*std::length\_error* If new length exceeds *max\_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, *out\_of\_range* is thrown. If the length of result exceeds *max\_size()*, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file *basic\_string.h*.

**5.375.396** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc > & std::basic_string<  
_CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type  
__n1, const _CharT * __s, size_type __n2 )`

Replace characters with value of a C substring.

### Parameters

*pos* Index of first character to replace.

*n1* Number of characters to be replaced.

*s* C string to insert.

*n2* Number of characters from *s* to use.

### Returns

Reference to this string.

### Exceptions

*std::out\_of\_range* If *pos1* > *size()*.

*std::length\_error* If new length exceeds *max\_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, *out\_of\_range* is thrown. If the length of result exceeds *max\_size()*, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 414 of file *basic\_string.tcc*.

**5.375.3.97** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace  
( iterator __i1, iterator __i2, const _CharT * __s ) [inline]`

Replace range of characters with C string.

#### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*s* C string value to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1504 of file `basic_string.h`.

**5.375.3.98** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace  
( size_type __pos, size_type __n, const basic_string<_CharT,  
_Traits, _Alloc> & __str ) [inline]`

Replace characters with value from another string.

#### Parameters

*pos* Index of first character to replace.

*n* Number of characters to be replaced.

*str* String to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::out\_of\_range* If *pos* is beyond the end of this string.

## 5.375 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2013

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos+n)` from this string. In place, the value of `str` is inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1359 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

```
5.375.3.99  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
            ( iterator __i1, iterator __i2, const _CharT* __s, size_type __n
              ) [inline]
```

Replace range of characters with C substring.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*s* C string value to insert.

*n* Number of characters from *s* to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1, i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1483 of file `basic_string.h`.

```
5.375.3.100 template<typename _CharT, typename _Traits, typename _Alloc
                > void std::basic_string<_CharT, _Traits, _Alloc>::reserve (
                size_type __res_arg = 0 )
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

*res\_arg* Number of characters required.

**Exceptions**

*std::length\_error* If *res\_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 502 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc >::append()`, `std::num_get<_CharT, _InIter >::do_get()`, and `std::operator>>()`.

**5.375.3.101** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string<_CharT, _Traits, _Alloc >::resize (  
size_type __n ) [inline]`

Resizes the string to the specified number of characters.

**Parameters**

*n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 739 of file `basic_string.h`.

Referenced by `std::basic_string<char >::resize()`.

**5.375.3.102** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string<_CharT, _Traits, _Alloc >::resize (  
size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

## 5.375 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference 2015

### Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

Definition at line 640 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::append(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::erase(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::size().

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get().

**5.375.3.103** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (  
const basic_string< _CharT, _Traits, _Alloc > & __str, size_type  
__pos = npos ) const [inline]`

Find last position of a string.

### Parameters

- str* String to locate.
- pos* Index of character to search back from (default end).

### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1812 of file basic\_string.h.

Referenced by std::basic\_string< char >::rfind().

**5.375.3.104** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (  
const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a C string.

### Parameters

- s* C string to locate.

*pos* Index of character to start search at (default end).

### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1840 of file `basic_string.h`.

**5.375.3.105** `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind ( _CharT __c, size_type __pos = npos ) const`

Find last position of a character.

### Parameters

*c* Character to locate.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 799 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.375.3.106** `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

### Parameters

*s* C string to locate.

*pos* Index of character to search back from.

*n* Number of characters from *s* to search for.

## 5.375 std::basic\_string< \_CharT, \_Traits, \_Alloc > Class Template Reference

### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 778 of file `basic_string.tcc`.

References `std::min()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.375.3.107** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit (  
) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 745 of file `basic_string.h`.

**5.375.3.108** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string< _CharT, _Traits, _Alloc >::size ( )  
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 701 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::bitset< _Nb >::bitset()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::find()`, `std::basic_string< char >::find()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::insert()`, `std::operator+()`, `std::basic_string< char >::replace()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_string< char >::rfind()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`.

**5.375.3.109** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (  
size_type __pos = 0, size_type __n = npos ) const [inline]`

Get a substring.

**Parameters**

- pos* Index of first character (default 0).  
*n* Number of characters in substring (default remainder).

**Returns**

The new string.

**Exceptions**

[\*std::out\\_of\\_range\*](#) If `pos > size()`.

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, [\*out\\_of\\_range\*](#) is thrown.

Definition at line 2112 of file `basic_string.h`.

**5.375.3.110** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string< _CharT, _Traits, _Alloc >::swap (  
basic_string< _CharT, _Traits, _Alloc > & __s )`

Swap contents with another string.

**Parameters**

- s* String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

Definition at line 519 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

Referenced by `std::swap()`.

**5.375.4 Member Data Documentation**

**5.375.4.1** `template<typename _CharT, typename _Traits, typename  
_Alloc> const basic_string< _CharT, _Traits, _Alloc >::size_type  
std::basic_string< _CharT, _Traits, _Alloc >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 271 of file `basic_string.h`.

The documentation for this class was generated from the following files:

## 5.376 `std::basic_stringbuf< _CharT, _Traits, _Alloc >` Class Template Reference

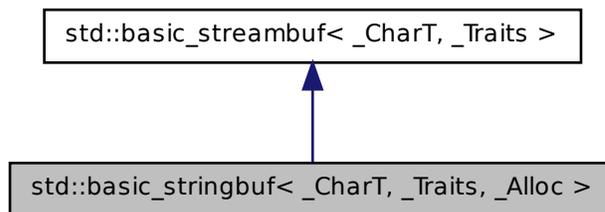
- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

### 5.376 `std::basic_stringbuf< _CharT, _Traits, _Alloc >` Class Template Reference

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

Inheritance diagram for `std::basic_stringbuf< _CharT, _Traits, _Alloc >`:



#### Public Types

- typedef `__string_type::size_type` **\_\_size\_type**
- typedef `basic_streambuf< char_type, traits_type >` **\_\_streambuf\_type**
- typedef `basic_string< char_type, _Traits, _Alloc >` **\_\_string\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `_Traits` **traits\_type**

## Public Member Functions

- [basic\\_stringbuf](#) ([ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- [basic\\_stringbuf](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- [streamsize](#) [in\\_avail](#) ()
- [int\\_type](#) [sbumpc](#) ()
- [int\\_type](#) [sgetc](#) ()
- [streamsize](#) [sgetn](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [int\\_type](#) [snextc](#) ()
- [int\\_type](#) [sputbackc](#) ([char\\_type](#) \_\_c)
- [int\\_type](#) [sputc](#) ([char\\_type](#) \_\_c)
- [streamsize](#) [sputn](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- void [stoss](#) ()
- void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
- [\\_\\_string\\_type](#) [str](#) () const
- [int\\_type](#) [sungetc](#) ()

## Protected Member Functions

- void [\\_M\\_stringbuf\\_init](#) ([ios\\_base::openmode](#) \_\_mode)
- void [\\_M\\_sync](#) ([char\\_type](#) \*\_\_base, [\\_\\_size\\_type](#) \_\_i, [\\_\\_size\\_type](#) \_\_o)
- void [\\_M\\_update\\_egptr](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) &)
- virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=[traits\\_type::eof\(\)](#))
- virtual [int\\_type](#) [overflow](#) ([int\\_type](#)=[traits\\_type::eof\(\)](#))
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=[traits\\_type::eof\(\)](#))
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#)=[traits\\_type::eof\(\)](#))
- void [pbump](#) (int \_\_n)
- virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in|ios\\_base::out](#))
- virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in|ios\\_base::out](#))
- virtual [basic\\_streambuf](#)< [char\\_type](#), [\\_Traits](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
- virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- void [setg](#) ([char\\_type](#) \*\_\_gbeg, [char\\_type](#) \*\_\_gnext, [char\\_type](#) \*\_\_gend)
- void [setp](#) ([char\\_type](#) \*\_\_pbeg, [char\\_type](#) \*\_\_pend)
- virtual [streamsize](#) [showmanyc](#) ()

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

- virtual int [sync](#) ()
- virtual [int\\_type uflow](#) ()
- virtual [int\\_type underflow](#) ()
- virtual [streamsize xsgetn](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- virtual [streamsize xspn](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
  - [char\\_type](#) \* [eback](#) () const
  - [char\\_type](#) \* [gptr](#) () const
  - [char\\_type](#) \* [egptr](#) () const
  - [char\\_type](#) \* [pbase](#) () const
  - [char\\_type](#) \* [pptr](#) () const
  - [char\\_type](#) \* [epptr](#) () const

### Protected Attributes

- [ios\\_base::openmode](#) [\\_M\\_mode](#)
- [\\_\\_string\\_type](#) [\\_M\\_string](#)

### Friends

- [template<bool\\_IsMove, typename \\_CharT2 > \\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, \\_CharT2 \\* >::\\_\\_type \\_\\_copy\\_move\\_a2](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [\\_CharT2](#) \*)
- [streamsize \\_\\_copy\\_streambufs\\_eof](#) ([\\_\\_streambuf\\_type](#) \*, [\\_\\_streambuf\\_type](#) \*, [bool](#) &)
- class [basic\\_ios](#)< [char\\_type](#), [traits\\_type](#) >
- class [basic\\_istream](#)< [char\\_type](#), [traits\\_type](#) >
- class [basic\\_ostream](#)< [char\\_type](#), [traits\\_type](#) >
- [template<typename \\_CharT2 > \\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, istreambuf\\_iterator< \\_CharT2 > >::\\_\\_type find](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, const [\\_CharT2](#) &)
- [template<typename \\_CharT2, typename \\_Traits2, typename \\_Alloc > basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [getline](#) ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [basic\\_string](#)< [\\_CharT2](#), [\\_Traits2](#), [\\_Alloc](#) > &, [\\_CharT2](#))
- class [istreambuf\\_iterator](#)< [char\\_type](#), [traits\\_type](#) >
- [template<typename \\_CharT2, typename \\_Traits2, typename \\_Alloc > basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [operator](#)>> ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [basic\\_string](#)< [\\_CharT2](#), [\\_Traits2](#), [\\_Alloc](#) > &)
- [template<typename \\_CharT2, typename \\_Traits2 > basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [operator](#)>> ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [\\_CharT2](#) \*)

- class `ostreambuf_iterator`< `char_type`, `traits_type` >
- `char_type` \* `_M_in_beg`
- `char_type` \* `_M_in_cur`
- `char_type` \* `_M_in_end`
- `char_type` \* `_M_out_beg`
- `char_type` \* `_M_out_cur`
- `char_type` \* `_M_out_end`
- `locale` `_M_buf_locale`
- `locale` `pubimbue` (`const locale` & `__loc`)
- `locale` `getloc` () `const`
- `__streambuf_type` \* `pubsetbuf` (`char_type` \* `__s`, `streamsize` `__n`)
- `pos_type` `pubseekoff` (`off_type` `__off`, `ios_base::seekdir` `__way`, `ios_base::openmode` `__mode=ios_base::in|ios_base::out`)
- `pos_type` `pubseekpos` (`pos_type` `__sp`, `ios_base::openmode` `__mode=ios_base::in|ios_base::out`)
- `int` `pubsync` ()

### 5.376.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.). For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 58 of file `sstream`.

### 5.376.2 Member Typedef Documentation

```
5.376.2.1 template<typename _CharT, typename _Traits, typename
_Alloc> typedef basic_streambuf<char_type, traits_type>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::__streambuf_type
```

This is a non-standard type.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 71 of file `sstream`.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

**5.376.2.2** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _CharT std::basic_stringbuf< _CharT, _Traits, _Alloc  
>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 62 of file sstream.

**5.376.2.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::int_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 67 of file sstream.

**5.376.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::off_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 69 of file sstream.

**5.376.2.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::pos_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 68 of file sstream.

**5.376.2.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
 typedef _Traits std::basic_stringbuf< _CharT, _Traits, _Alloc  
 >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 63 of file sstream.

### 5.376.3 Constructor & Destructor Documentation

**5.376.3.1** `template<typename _CharT, typename _Traits, typename _Alloc>  
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf  
 ( ios_base::openmode __mode = ios_base::in | ios_base::out )  
 [inline, explicit]`

Starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

Definition at line 92 of file sstream.

**5.376.3.2** `template<typename _CharT, typename _Traits, typename _Alloc>  
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf  
 ( const __string_type & __str, ios_base::openmode __mode =  
 ios_base::in | ios_base::out ) [inline, explicit]`

Starts with an existing string buffer.

#### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 105 of file sstream.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### 5.376.4 Member Function Documentation

**5.376.4.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eback ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.376.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.376.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 513 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.376.4.4** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

#### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

**5.376.4.5** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::getloc ( ) const  
[inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

**5.376.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

### 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#).

**5.376.4.7** `template<typename _CharT, typename _Traits> virtual void  
std::basic_streambuf< _CharT, _Traits >::imbue ( const locale & )  
[inline, protected, virtual, inherited]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 554 of file streambuf.

**5.376.4.8** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline,  
inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file `streambuf`.

```
5.376.4.9 template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits >::overflow ( int_type =
traits_type::eof() ) [inline, protected, virtual,
inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

*c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output `streambuf` can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits >`.

Definition at line 746 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits >::xsputn()`.

```
5.376.4.10 template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits >::pbackfail ( int_type =
traits_type::eof() ) [inline, protected, virtual,
inherited]
```

Tries to back up the input sequence.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### Parameters

*c* The character to be inserted back into the sequence.

### Returns

eof() on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns eof().

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 702 of file `streambuf`.

**5.376.4.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

**5.376.4.12** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

### Parameters

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits >::xsputn()`.

**5.376.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits >::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits >::sync()`, and `std::basic_streambuf<_CharT, _Traits >::xsputn()`.

**5.376.4.14** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits >::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

**5.376.4.15** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

**5.376.4.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

**5.376.4.17** `template<typename _CharT, typename _Traits>  
__streambuf_type* std::basic_streambuf< _CharT, _Traits  
>::pubsetbuf ( char_type * __s, streamsize __n ) [inline,  
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

**5.376.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf< _CharT, _Traits >::pubsync ( )  
[inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.376.4.19** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

**5.376.4.20** `template<typename _CharT, typename _Traits> virtual  
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff  
( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [inline, protected,  
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

**5.376.4.21** `template<typename _CharT, typename _Traits> virtual pos_type  
std::basic_streambuf< _CharT, _Traits >::seekpos ( pos_type,  
ios_base::openmode = ios_base::in | ios_base::out ) [inline,  
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

**5.376.4.22** `template<typename _CharT, typename _Traits, typename _Alloc>  
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::setbuf ( char_type * __s, streamsize __n ) [inline,  
protected, virtual]`

Manipulates the buffer.

### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

### Returns

`this`

If no buffer has already been created, and both *s* and *n* are non-zero, then *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 196 of file `sstream`.

References `std::basic_string< _CharT, _Traits, _Alloc >::clear()`.

**5.376.4.23** `template<typename _CharT, typename _Traits> virtual  
basic_streambuf<char_type,_Traits>* std::basic_streambuf<  
_CharT,_Traits >::setbuf ( char_type *, streamsize ) [inline,  
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

#### Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

**5.376.4.24** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT,_Traits >::setg ( char_type *  
__gbeg, char_type * __gnext, char_type * __gend ) [inline,  
protected, inherited]`

Setting the three read area pointers.

#### Parameters

*gbeg* A pointer.  
*gnext* A pointer.  
*gend* A pointer.

#### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

**5.376.4.25** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT,_Traits >::setp ( char_type  
* __pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

#### Parameters

*pbeg* A pointer.

## 5.376 std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc> Class Template Reference

*pend* A pointer.

### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

**5.376.4.26** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sgetc() [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.376.4.27** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn(char_type * __s,  
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

**5.376.4.28** `template<typename _CharT, typename _Traits, typename _Alloc>  
virtual streamsize std::basic_stringbuf<_CharT, _Traits, _Alloc  
>::showmanyc() [inline, protected, virtual]`

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 164 of file `sstream`.

**5.376.4.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::snextc( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.376.4.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputbackc( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

**Parameters**

`c` The character to push back.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.376.4.31** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

### Parameters

*c* A character to output.

### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**5.376.4.32** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

**5.376.4.33** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::stoss( ) [inline,  
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

**5.376.4.34** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( const  
__string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

`s` The string to use as a new sequence.

Deallocates any previous stored sequence, then copies `s` to use as a new one.

Definition at line 144 of file `sstream`.

References `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.376.4.35** `template<typename _CharT, typename _Traits, typename _Alloc>  
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1*

Definition at line 120 of file `sstream`.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

**5.376.4.36** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::ungetc()`.

**5.376.4.37** `template<typename _CharT, typename _Traits> virtual int  
std::basic_streambuf< _CharT, _Traits >::sync ( void )  
[inline, protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

### Note

Base class version does nothing, returns zero.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT >>`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 605 of file streambuf.

**5.376.4.38** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 678 of file `streambuf`.

```
5.376.4.39 template<class _CharT, class _Traits, class _Alloc >
             basic_stringbuf< _CharT, _Traits, _Alloc >::int_type
             std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ( )
             [protected, virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 131 of file `sstream.tcc`.

References `std::basic_stringbuf<_CharT, _Traits, _Alloc>::M_mode`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, and `std::ios_base::in`.

## 5.376 std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc> Class Template Reference

**5.376.4.40** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsgetn ( char_type *  
__s, streamsize __n ) [protected, virtual, inherited]`

Multiple character extraction.

### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

### Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

**5.376.4.41** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsputn ( const  
char_type * __s, streamsize __n ) [protected, virtual,  
inherited]`

Multiple character insertion.

### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 79 of file streambuf.tcc.

References [std::basic\\_streambuf<\\_CharT, \\_Traits>::eptr\(\)](#), [std::min\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::pbump\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::pptr\(\)](#).

## 5.376.5 Member Data Documentation

### 5.376.5.1 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::basic\\_filebuf\(\)](#).

### 5.376.5.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg` [protected, inherited]

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file streambuf.

### 5.376.5.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur` [protected, inherited]

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

## 5.376 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

**5.376.5.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

**5.376.5.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc  
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 77 of file `sstream`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

**5.376.5.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

```
5.376.5.7 template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, inherited]
```

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

```
5.376.5.8 template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]
```

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following files:

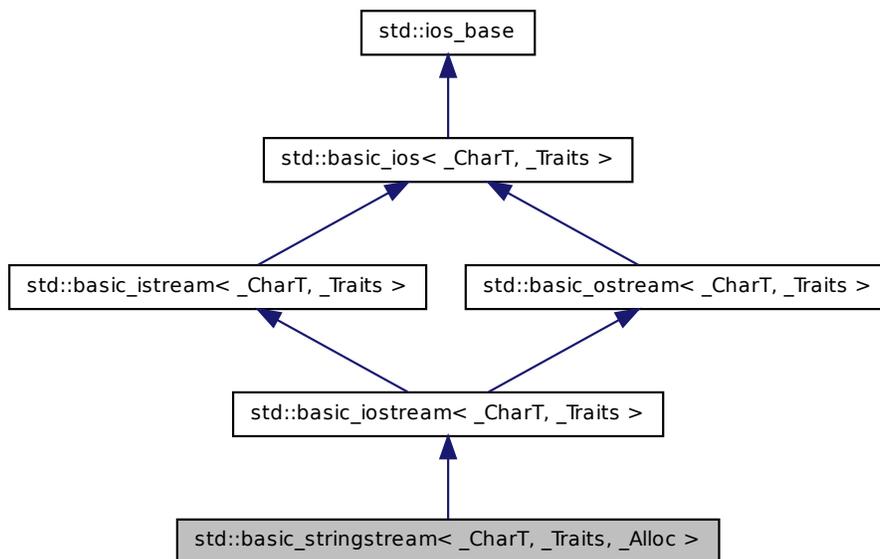
- [sstream](#)
- [sstream.tcc](#)

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_stringstream<_CharT, _Traits, _Alloc>`:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_iostream<char_type, traits_type>` `__iostream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`

- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
  - typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
  - typedef `basic_ostream< _CharT, _Traits > __ostream_type`
  - typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
  - typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
  - typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
  - typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
  - typedef `_Alloc allocator_type`
  - typedef `_CharT char_type`
  - typedef `_CharT char_type`
  - enum `event { erase_event, imbue_event, copyfmt_event }`
  - typedef `void(* event_callback)(event, ios_base &, int)`
  - typedef `_Ios_Fmtflags fmtflags`
  - typedef `_Traits::int_type int_type`
  - typedef `traits_type::int_type int_type`
  - typedef `int io_state`
  - typedef `_Ios_Iostate iostate`
  - typedef `traits_type::off_type off_type`
  - typedef `_Traits::off_type off_type`
  - typedef `int open_mode`
  - typedef `_Ios_Openmode openmode`
  - typedef `traits_type::pos_type pos_type`
  - typedef `_Traits::pos_type pos_type`
  - typedef `int seek_dir`
  - typedef `_Ios_Seekdir seekdir`
  - typedef `std::streamoff streamoff`
  - typedef `std::streampos streampos`
  - typedef `_Traits traits_type`
  - typedef `_Traits traits_type`
- 
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`

## Public Member Functions

- `basic_stringstream (ios_base::openmode __m=ios_base::out|ios_base::in)`
- `basic_stringstream (const __string_type &__str, ios_base::openmode __m=ios_base::out|ios_base::in)`
- `~basic_stringstream ()`
- `const locale & _M_getloc () const`

- `void _M_setstate (iostate __state)`
- `bool bad () const`
- `void clear (iostate __state=goodbit)`
- `basic_ios & copyfmt (const basic_ios &__rhs)`
- `bool eof () const`
- `iostate exceptions () const`
- `void exceptions (iostate __except)`
- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__stringbuf_type * rdbuf () const`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits`  
`> *__sb)`
- `iostate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`

- `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iostate __state)`
  - `__string_type str () const`
  - `void str (const __string_type &__s)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type * __sb)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
  - `__istream_type & get (char_type &__c)`
  - `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & get (char_type *__s, streamsize __n)`
  - `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
  - `__istream_type & get (__streambuf_type &__sb)`
  - `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `__istream_type & getline (char_type *__s, streamsize __n)`
  - `__istream_type & ignore ()`
  - `__istream_type & ignore (streamsize __n)`
  - `__istream_type & ignore (streamsize __n, int_type __delim)`
  - `int_type peek ()`
  - `__istream_type & read (char_type *__s, streamsize __n)`
  - `streamsize readsome (char_type *__s, streamsize __n)`
  - `__istream_type & putback (char_type __c)`
  - `__istream_type & unget ()`
  - `int sync ()`
  - `pos_type tellg ()`
  - `__istream_type & seekg (pos_type)`
  - `__istream_type & seekg (off_type, ios_base::seekdir)`
- 
- `operator void * () const`
  - `bool operator! () const`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`

- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename `_ValueT` >  
  `__istream_type` & `_M_extract` (`_ValueT` &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)

- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  \_\_ostream\_type & **\_M\_insert** (\_ValueT \_\_v)
- void **init** (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)

## Protected Attributes

- \_Callback\_list \* **\_M\_callbacks**
- const \_\_ctype\_type \* **\_M\_ctype**
- iostate **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- fmtflags **\_M\_flags**
- streamsize **\_M\_gcount**
- locale **\_M\_ios\_locale**
- \_Words **\_M\_local\_word** [\_S\_local\_word\_size]
- const \_\_num\_get\_type \* **\_M\_num\_get**
- const \_\_num\_put\_type \* **\_M\_num\_put**
- streamsize **\_M\_precision**
- basic\_streambuf< \_CharT, \_Traits > \* **\_M\_streambuf**
- iostate **\_M\_streambuf\_state**
- basic\_ostream< \_CharT, \_Traits > \* **\_M\_tie**
- streamsize **\_M\_width**
- \_Words \* **\_M\_word**
- int **\_M\_word\_size**
- \_Words **\_M\_word\_zero**

## Friends

- class **sentry**
- class **sentry**

### 5.377.1 Detailed Description

template<typename \_CharT, typename \_Traits, typename \_Alloc> class  
std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >

Controlling input and output for std::string.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 476 of file sstream.

## 5.377.2 Member Typedef Documentation

**5.377.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream<_CharT, _Traits>::_ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file `istream`.

**5.377.2.2** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream<_CharT, _Traits>::_ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 71 of file `ostream`.

**5.377.2.3** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_istream<_CharT, _Traits>::_num_get_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 70 of file `istream`.

**5.377.2.4** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::_num_put_type [inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_ostream<\\_CharT, \\_Traits>](#), [std::basic\\_ostream<char, \\_Traits>](#), and [std::basic\\_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

**5.377.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits >::_num_put_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Definition at line 70 of file ostream.

**5.377.2.6** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_istream<_CharT, _Traits >::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits >](#), and [std::basic\\_istringstream<\\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 59 of file istream.

**5.377.2.7** `template<typename _CharT, typename _Traits, typename _Alloc > typedef _CharT std::basic_stringstream<_CharT, _Traits, _Alloc >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits >](#).

Definition at line 480 of file sstream.

**5.377.2.8** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the [ios\\_base](#) object.

*int* The integer provided when the callback was registered.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference 2055**

---

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 444 of file [ios\\_base.h](#).

**5.377.2.9 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags [inherited]**

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 263 of file [ios\\_base.h](#).

**5.377.2.10** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 60 of file istream.

**5.377.2.11** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef traits_type::int_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 485 of file sstream.

**5.377.2.12** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file ios\_base.h.

**5.377 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** 2057

---

**5.377.2.13** `template<typename _CharT, typename _Traits, typename _Alloc  
> typedef traits_type::off_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 487 of file sstream.

**5.377.2.14** `template<typename _CharT, typename _Traits> typedef  
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 62 of file istream.

**5.377.2.15** `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file ios\_base.h.

**5.377.2.16** `template<typename _CharT , typename _Traits , typename _Alloc  
> typedef traits_type::pos_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 486 of file sstream.

**5.377.2.17** `template<typename _CharT, typename _Traits> typedef  
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 61 of file istream.

**5.377.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 401 of file ios\_base.h.

**5.377.2.19** `template<typename _CharT, typename _Traits> typedef  
_Traits std::basic_istream< _CharT, _Traits >::traits_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2059

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), and [std::basic\\_istringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 63 of file `istream`.

**5.377.2.20** `template<typename _CharT, typename _Traits, typename _Alloc> typedef _Traits std::basic_stringstream<_CharT, _Traits, _Alloc>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 481 of file `sstream`.

### 5.377.3 Member Enumeration Documentation

**5.377.3.1** `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 427 of file `ios_base.h`.

### 5.377.4 Constructor & Destructor Documentation

**5.377.4.1** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream ( ios_base::openmode __m = ios_base::out | ios_base::in ) [inline, explicit]`

Default constructor starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

Initializes `sb` using `mode`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 510 of file sstream.

```
5.377.4.2 template<typename _CharT , typename _Traits , typename
           _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
           >::basic_stringstream ( const __string_type & __str,
           ios_base::openmode __m = ios_base::out | ios_base::in )
           [inline, explicit]
```

Starts with an existing string buffer.

#### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

Initializes *sb* using *str* and *mode*, and passes *&sb* to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 526 of file sstream.

```
5.377.4.3 template<typename _CharT , typename _Traits , typename
           _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
           >::~~basic_stringstream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 537 of file sstream.

### 5.377.5 Member Function Documentation

```
5.377.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline,
           inherited]
```

Locale access.

#### Returns

A reference to the current locale.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2061

---

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

### 5.377.5.2 `template<typename _CharT, typename _Traits> void std::basic_ostream<_CharT, _Traits>::_M_write ( const char_type * __s, streamsize __n ) [inline, inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

### 5.377.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

**5.377.5.4** `template<typename _CharT, typename _Traits > void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.377.5.5** `template<typename _CharT, typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (   
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

*\_\_rhs* The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.377.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::eof ( ) const [inline, inherited]`

Fast error checking.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2063

### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.377.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const` [`inline`, `inherited`]

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.377.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except )` [`inline`, `inherited`]

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

```

#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file basic\_ios.h.

#### 5.377.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail( ) const` **[inline, inherited]**

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic\_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

#### 5.377.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits >::fill( ) const` **[inline, inherited]**

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 360 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.377.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

**5.377.5.12** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

**5.377.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios\_base.h.

**5.377.5.14** `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::basic_ostream<_CharT, _Traits >::flush ( ) [inherited]`

Synchronizing the stream buffer.

#### Returns

\*this

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( ) ->pubsync ( )`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.377.5.15** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits >::gcount ( ) const [inline, inherited]`

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

**5.377.5.16** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits >::int_type std::basic_istream<_CharT, _Traits >::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2067

---

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.17** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type & __c ) [inherited]`

Simple extraction.

### Parameters

*c* The character in which to store data.

### Returns

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.18** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

**5.377.5.19** `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::sputc()`.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2069

---

**5.377.5.20** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

**Returns**

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file istream.

**5.377.5.21** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

**Parameters**

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.377.5.22** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim )`  
**[inherited]**

String extraction.

**Parameters**

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

*delim* A "stop" character.

**Returns**

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2071

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

### 5.377.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

#### Returns

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

### 5.377.5.24 `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.377.5.25** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::good ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

**5.377.5.26** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore  
( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.377.5.27** `template<typename _CharT , typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore  
( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if  $n \neq \text{std::numeric\_limits}<\text{int}>::\text{max}()$ ,  $n$  characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.377.5.29** `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.377.5.30** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

**5.377.5.31** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2075

Definition at line 740 of file `ios_base.h`.

**5.377.5.32** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

### Parameters

- c* The character to narrow.
- dfault* The character to narrow.

### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> (getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

**5.377.5.33** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

**5.377.5.34** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

**5.377.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( float __f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

**5.377.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( bool __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

**5.377.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2077

### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file `ostream`.

**5.377.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ostream_type &(*)(__ostream_type &) _pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

**5.377.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type &(*)(__ios_type &) _pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

**5.377.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*)(ios_base &) _pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

**5.377.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

**5.377.5.42** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.377.5.43** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2079

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.377.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file `ostream`.

**5.377.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

**5.377.5.46** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( double __f ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

**5.377.5.47** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< ( long double __f ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

**5.377.5.48** `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< ( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

**Parameters**

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.49 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long __n ) [inline, inherited]`**

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

**5.377.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

**5.377.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

**5.377.5.52** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>( short & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2083

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.53** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

**5.377.5.54** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

**5.377.5.55** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( ios_base &(*)(ios_base &) _pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `istream` header.

Definition at line 131 of file `istream`.

**5.377.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( double & _f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

**5.377.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( long double & _f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc > Class Template Reference** 2085

---

**5.377.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

**5.377.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

**5.377.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

**5.377.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 194 of file istream.

**5.377.5.62** `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2087

---

**5.377.5.63** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

**Parameters**

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.64** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file istream.

**5.377.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 120 of file `istream`.

**5.377.5.66** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 124 of file `istream`.

**5.377.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

**5.377.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2089

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

**5.377.5.69** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.70** `streamsize std::ios_base::precision( ) const [inline, inherited]`

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.377.5.71** `streamsize std::ios_base::precision ( streamsize __prec )`  
`[inline, inherited]`

Changing flags.

#### Parameters

*prec* The new precision value.

#### Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

**5.377.5.72** `template<typename _CharT, typename _Traits > basic_ostream<`  
`_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (`  
`char_type __c ) [inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**5.377.5.73** `template<typename _CharT, typename _Traits > basic_istream<`  
`_CharT, _Traits > & std::basic_istream< _CharT, _Traits`  
`>::putback ( char_type __c ) [inherited]`

Unextracting a single character.

**Parameters**

*c* The character to push back into the input stream.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**5.377.5.74 void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]**

Access to void pointer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to a void\* associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

**5.377.5.75** `template<typename _CharT, typename _Traits, typename _Alloc  
> __stringbuf_type* std::basic_stringstream< _CharT, _Traits,  
_Alloc >::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

### Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 548 of file `sstream`.

**5.377.5.76** `template<typename _CharT, typename _Traits> basic_streambuf<  
_CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (  
basic_streambuf< _CharT, _Traits > * __sb ) [inherited]`

Changing the underlying buffer.

### Parameters

*sb* The new stream buffer.

### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::clear()`.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2093

---

**5.377.5.77** `template<typename _CharT, typename _Traits> istate  
std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

**5.377.5.78** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(  
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

**Parameters**

*s* A character array.

*n* Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.79** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.80** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

- `__fn` The function to add.
- `__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2095

---

**5.377.5.81** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.84** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2097

---

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.377.5.85 `fmtflags std::ios_base::setf( fmtflags __fmtfl )` [`inline`, `inherited`]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

### 5.377.5.86 `fmtflags std::ios_base::setf( fmtflags __fmtfl, fmtflags __mask )` [`inline`, `inherited`]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.377.5.87** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.377.5.88** `template<typename _CharT , typename _Traits , typename _Alloc  
> __string_type std::basic_stringstream< _CharT, _Traits, _Alloc  
>::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

`rdbuf () -> str ()`

Definition at line 556 of file `sstream`.

**5.377.5.89** `template<typename _CharT , typename _Traits , typename _Alloc  
> void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (   
const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

*s* The string to use as a new sequence.

## 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2099

Calls `rdbuf() ->str(s)`.

Definition at line 566 of file `sstream`.

**5.377.5.90** `template<typename _CharT, typename _Traits> int std::basic_istream<_CharT, _Traits>::sync ( void )`  
[*inherited*]

Synchronizing the stream buffer.

### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.377.5.91** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
[*static, inherited*]

Interaction with the standard C I/O objects.

### Parameters

*sync* Whether to synchronize or not.

### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.377.5.92** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg( void ) [inherited]`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.377.5.93** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.377.5.94** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file basic\_ios.h.

**5.377.5.95** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, [std::cin](#) is tied to [std::cout](#).

Definition at line 285 of file basic\_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.377.5.96** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.377.5.97** `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios\_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.377.5.98** `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen ( char __c ) const [inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> (getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic\_ios.h.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

**5.377 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference 2103**

---

**5.377.5.99** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.377.5.100** `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.377.5.101** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

**Parameters**

*s* The array to insert.

*n* Maximum number of characters to insert.

**Returns**

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- $n$  characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

### 5.377.5.102 `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.377.6 Member Data Documentation

### 5.377.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

### 5.377.6.2 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

### 5.377 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2105

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.377.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

#### 5.377.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

#### 5.377.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### 5.377.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 321 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.377.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

**5.377.6.8 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 380 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

**5.377.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

**5.377.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits >::imbue()`, `std::basic_istream<_CharT, _Traits >::tellg()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

**5.377.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

**5.377.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits >::open()`.

**5.377.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

**5.377.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

**5.377.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

**5.377.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 324 of file ios\_base.h.

**5.377.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::time_get<_CharT, _InIter >::do_get_monthname()`, `std::time_get<_CharT, _InIter >::do_get_weekday()`, `std::time_get<_CharT, _InIter >::do_get_year()`, `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, and `std::basic_istream<_CharT, _Traits >::unget()`.

**5.377.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.377.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_filebuf<_CharT, _Traits >::showmanyc()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf<_CharT, _Traits >::underflow()`.

**5.377.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

**5.377.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

**5.377.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::operator<<()`.

**5.377.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits >::seekp()`, and `std::basic_ostream<_CharT, _Traits >::tellp()`.

**5.377.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file `ios_base.h`.

**5.377.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 294 of file `ios_base.h`.

**5.377.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 298 of file `ios_base.h`.

**5.377.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.377.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file ios\_base.h.

**5.377.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 308 of file ios\_base.h.

**5.377.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for `ofstream`.

Definition at line 389 of file ios\_base.h.

**5.377.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 311 of file ios\_base.h.

**5.377.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 315 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

- [sstream](#)

## 5.378 `std::bernoulli_distribution` Class Reference

A Bernoulli random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef bool [result\\_type](#)

### Public Member Functions

- [bernoulli\\_distribution](#) (double \_\_p=0.5)
- **bernoulli\_distribution** (const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- void **param** (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void **reset** ()

#### 5.378.1 Detailed Description

A Bernoulli random number distribution. Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3240 of file random.h.

#### 5.378.2 Member Typedef Documentation

##### 5.378.2.1 typedef bool `std::bernoulli_distribution::result_type`

The type of the range of the distribution.

Definition at line 3244 of file random.h.

### 5.378.3 Constructor & Destructor Documentation

**5.378.3.1** `std::bernoulli_distribution::bernoulli_distribution ( double __p = 0.5 ) [inline, explicit]`

Constructs a Bernoulli distribution with likelihood  $p$ .

#### Parameters

`__p` [IN] The likelihood of a true result being returned. Must be in the interval  $[0, 1]$ .

Definition at line 3277 of file random.h.

### 5.378.4 Member Function Documentation

**5.378.4.1** `result_type std::bernoulli_distribution::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3327 of file random.h.

**5.378.4.2** `result_type std::bernoulli_distribution::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3320 of file random.h.

**5.378.4.3** `template<typename _UniformRandomNumberGenerator> result_type std::bernoulli_distribution::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3335 of file random.h.

References `operator()()`, and `param()`.

Referenced by `operator()()`.

**5.378.4.4** `double std::bernoulli_distribution::p ( ) const [inline]`

Returns the  $p$  parameter of the distribution.

Definition at line 3298 of file random.h.

**5.378.4.5** `void std::bernoulli_distribution::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3313 of file `random.h`.

**5.378.4.6** `param_type std::bernoulli_distribution::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 3305 of file `random.h`.

Referenced by `operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.378.4.7** `void std::bernoulli_distribution::reset ( ) [inline]`

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3292 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.379 `std::bernoulli_distribution::param_type` Struct Reference

### Public Types

- typedef `bernoulli_distribution` `distribution_type`

### Public Member Functions

- `param_type` (double `__p=0.5`)
- double `p` ( ) const

## Friends

- `bool operator==(const param\_type &__p1, const param\_type &__p2)`

### 5.379.1 Detailed Description

Parameter type.

Definition at line 3246 of file `random.h`.

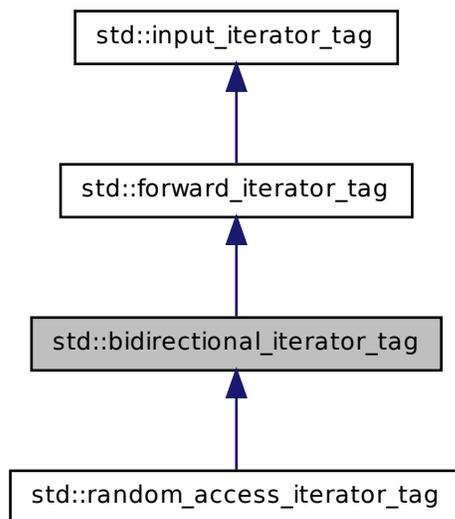
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.380 `std::bidirectional_iterator_tag` Struct Reference

Bidirectional iterators support a superset of forward iterator `///` operations.

Inheritance diagram for `std::bidirectional_iterator_tag`:



## 5.381 `std::binary_function<_Arg1, _Arg2, _Result >` Struct Template Reference

### 5.380.1 Detailed Description

Bidirectional iterators support a superset of forward iterator /// operations.

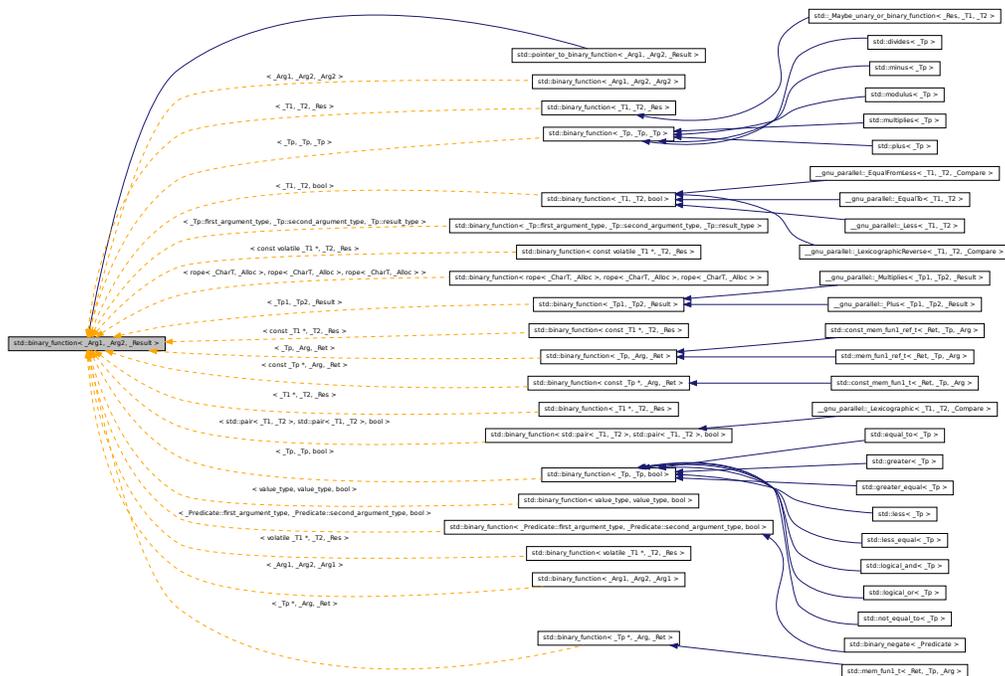
Definition at line 95 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.381 `std::binary_function<_Arg1, _Arg2, _Result >` Struct Template Reference

Inheritance diagram for `std::binary_function<_Arg1, _Arg2, _Result >`:



### Public Types

- typedef `_Arg1` `first_argument_type`
- typedef `_Result` `result_type`

- `typedef _Arg2 second\_argument\_type`

### 5.381.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> struct
std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 112 of file `stl_function.h`.

### 5.381.2 Member Typedef Documentation

**5.381.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`  
`>::first_argument_type`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.381.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`  
`>::result_type`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.381.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`  
`>::second_argument_type`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.382 std::binary\_negate< \_Predicate > Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::binary\_negate< \_Predicate >:



### Public Types

- typedef `_Predicate::first_argument_type` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Predicate::second_argument_type` `second_argument_type`

### Public Member Functions

- `binary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y`) `const`

### Protected Attributes

- `_Predicate _M_pred`

### 5.382.1 Detailed Description

```
template<typename _Predicate> class std::binary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 369 of file `stl_function.h`.

### 5.382.2 Member Typedef Documentation

**5.382.2.1** `typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type` `[inherited]`

the type of the first argument *///* (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.382.2.2** `typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl\_function.h.

**5.382.2.3** `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl\_function.h.

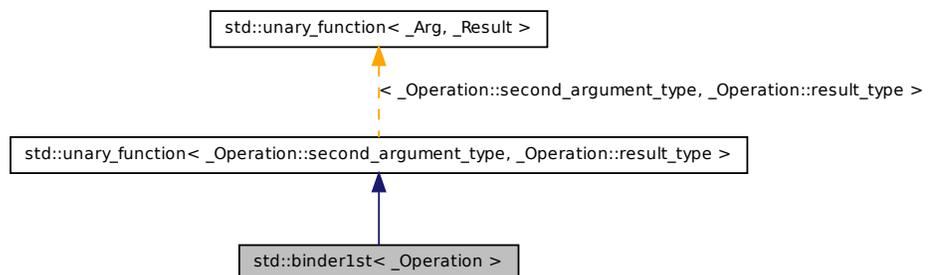
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.383 `std::binder1st< _Operation >` Class Template Reference

One of the [binder functors](#).

Inheritance diagram for `std::binder1st< _Operation >`:



## Public Types

- typedef \_Operation::second\_argument\_type [argument\\_type](#)
- typedef \_Operation::result\_type [result\\_type](#)

## Public Member Functions

- **binder1st** (const \_Operation &\_\_x, const typename \_Operation::first\_argument\_type &\_\_y)
- \_Operation::result\_type **operator()** (typename \_Operation::second\_argument\_type &\_\_x) const
- \_Operation::result\_type **operator()** (const typename \_Operation::second\_argument\_type &\_\_x) const

## Protected Attributes

- \_Operation **op**
- \_Operation::first\_argument\_type **value**

### 5.383.1 Detailed Description

`template<typename _Operation> class std::binder1st< _Operation >`

One of the [binder functors](#).

Definition at line 98 of file binders.h.

### 5.383.2 Member Typedef Documentation

**5.383.2.1** `typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type , _Operation::result_type >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file stl\_function.h.

**5.383.2.2** `typedef _Operation::result_type std::unary_function< _Operation::second_argument_type , _Operation::result_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file stl\_function.h.

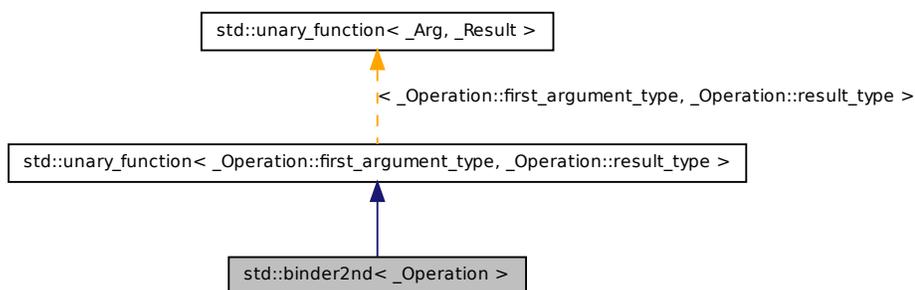
The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.384 `std::binder2nd< _Operation >` Class Template Reference

One of the [binder functors](#).

Inheritance diagram for `std::binder2nd< _Operation >`:



### Public Types

- typedef `_Operation::first_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

### Public Member Functions

- **binder2nd** (`const _Operation &__x`, `const typename _Operation::second_argument_type &__y`)
- `_Operation::result_type` **operator()** (`typename _Operation::first_argument_type &__x`) `const`
- `_Operation::result_type` **operator()** (`const typename _Operation::first_argument_type &__x`) `const`

## Protected Attributes

- `_Operation op`
- `_Operation::second_argument_type value`

### 5.384.1 Detailed Description

```
template<typename _Operation> class std::binder2nd<_Operation >
```

One of the [binder functors](#).

Definition at line 133 of file `binders.h`.

### 5.384.2 Member Typedef Documentation

**5.384.2.1** `typedef _Operation::first_argument_type std::unary_function<_Operation::first_argument_type, _Operation::result_type>::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.384.2.2** `typedef _Operation::result_type std::unary_function<_Operation::first_argument_type, _Operation::result_type>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.385 `std::binomial_distribution<_IntType>` Class Template Reference

A discrete binomial random number distribution.

### Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **binomial\_distribution** (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- **binomial\_distribution** (const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- double **p** () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void [reset](#) ()
- `_IntType` **t** () const

## Friends

- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_ostream](#)< `_CharT`, `_Traits` > & **operator<<** ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &, const [std::binomial\\_distribution](#)< `_IntType1` > &)
- template<typename `_IntType1` >  
bool **operator==** (const [std::binomial\\_distribution](#)< `_IntType1` > &\_\_d1, const [std::binomial\\_distribution](#)< `_IntType1` > &\_\_d2)
- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & **operator>>** ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &, [std::binomial\\_distribution](#)< `_IntType1` > &)

### 5.385.1 Detailed Description

```
template<typename _IntType = int> class std::binomial\_distribution< _IntType >
```

A discrete binomial random number distribution. The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3417 of file `random.h`.

## 5.385.2 Member Typedef Documentation

**5.385.2.1** `template<typename _IntType = int> typedef _IntType  
std::binomial_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 3424 of file `random.h`.

## 5.385.3 Member Function Documentation

**5.385.3.1** `template<typename _IntType = int> result_type  
std::binomial_distribution<_IntType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3527 of file `random.h`.

Referenced by `std::binomial_distribution<_IntType>::operator()`.

**5.385.3.2** `template<typename _IntType = int> result_type  
std::binomial_distribution<_IntType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3520 of file `random.h`.

**5.385.3.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::binomial_distribution<_IntType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3535 of file `random.h`.

References `std::binomial_distribution<_IntType>::operator()`, and `std::binomial_distribution<_IntType>::param()`.

Referenced by `std::binomial_distribution<_IntType>::operator()`.

**5.385.3.4** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > binomial_distribution<  
_IntType >::result_type std::binomial_distribution< _IntType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1390 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::binomial_distribution< _IntType >::max()`.

**5.385.3.5** `template<typename _IntType = int> double  
std::binomial_distribution< _IntType >::p ( ) const [inline]`

Returns the distribution `p` parameter.

Definition at line 3498 of file `random.h`.

**5.385.3.6** `template<typename _IntType = int> void std::binomial_  
distribution< _IntType >::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3513 of file `random.h`.

**5.385.3.7** `template<typename _IntType = int> param_type  
std::binomial_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3505 of file `random.h`.

Referenced by `std::binomial_distribution< _IntType >::operator()`.

**5.385.3.8** `template<typename _IntType = int> void  
std::binomial_distribution<_IntType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 3484 of file `random.h`.

References `std::normal_distribution<_RealType>::reset()`.

**5.385.3.9** `template<typename _IntType = int> _IntType  
std::binomial_distribution<_IntType>::t( ) const [inline]`

Returns the distribution  $t$  parameter.

Definition at line 3491 of file `random.h`.

## 5.385.4 Friends And Related Function Documentation

**5.385.4.1** `template<typename _IntType = int> template<typename _IntType1,  
typename _CharT, typename _Traits> std::basic_ostream<_CharT,  
_Traits>& operator<<( std::basic_ostream<_CharT, _Traits> &  
const std::binomial_distribution<_IntType1> & ) [friend]`

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `binomial_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.385.4.2** `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::binomial_distribution<_IntType1>  
& __d1, const std::binomial_distribution<_IntType1> & __d2 )  
[friend]`

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3550 of file `random.h`.

**5.385.4.3** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & ,  
std::binomial_distribution< _IntType1 > & ) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `binomial_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.386 `std::binomial_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef `binomial_distribution< _IntType > distribution_type`

### Public Member Functions

- `param_type` (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- `double p` () const
- `_IntType t` () const

### Friends

- class `binomial_distribution< _IntType >`
- `bool operator==` (const `param_type` &`__p1`, const `param_type` &`__p2`)

### 5.386.1 Detailed Description

**template<typename \_IntType = int> struct `std::binomial_distribution<_IntType>::param_type`**

Parameter type.

Definition at line 3426 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.387 `std::bitset<_Nb>` Class Template Reference

The `bitset` class represents a *fixed-size* sequence of bits.

Inheritance diagram for `std::bitset<_Nb>`:



### Classes

- class [reference](#)

### Public Member Functions

- [bitset](#) ()
- [bitset](#) (unsigned long long \_\_val)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position=0)
- [bitset](#) (const char \*\_\_str)
- size\_t [Find\\_first](#) () const

- `size_t Find_next (size_t __prev) const`
- `template<class _CharT, class _Traits >`  
`void M_copy_from_ptr (const _CharT *, size_t, size_t, size_t, _CharT, _-`  
`CharT)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`  
`> &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`  
`> &__s, size_t __pos, size_t __n)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &, _-`  
`CharT, _CharT) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`void M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &__s)`  
`const`
- `bool all () const`
- `bool any () const`
- `size_t count () const`
- `bitset<_Nb > & flip ()`
- `bitset<_Nb > & flip (size_t __position)`
- `bool none () const`
- `bitset<_Nb > operator~ () const`
- `bitset<_Nb > & reset ()`
- `bitset<_Nb > & reset (size_t __position)`
- `bitset<_Nb > & set ()`
- `bitset<_Nb > & set (size_t __position, bool __val=true)`
- `size_t size () const`
- `bool test (size_t __position) const`
- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string ()`  
`const`
- `template<class _CharT >`  
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`  
`> > to_string () const`
- `template<class _CharT >`  
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`  
`> > to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string< _CharT, _Traits, _Alloc > to_string () const`
- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string (_-`  
`CharT __zero, _CharT __one=_CharT('1')) const`

- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string<_CharT, _Traits, _Alloc >` **to\_string** (`_CharT __zero, _CharT __one=_CharT('1')`) const
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char> >` **to\_string** (`char __zero, char __one='1'`) const
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char> >` **to\_string** () const
  - unsigned long long **to\_ullong** () const
  - unsigned long **to\_ulong** () const
- 
- `bitset<_Nb >` & **operator&=** (const `bitset<_Nb >` &`__rhs`)
  - `bitset<_Nb >` & **operator|=** (const `bitset<_Nb >` &`__rhs`)
  - `bitset<_Nb >` & **operator^=** (const `bitset<_Nb >` &`__rhs`)
- 
- `bitset<_Nb >` & **operator<<=** (`size_t __position`)
  - `bitset<_Nb >` & **operator>>=** (`size_t __position`)
- 
- `bitset<_Nb >` & **\_Unchecked\_set** (`size_t __pos`)
  - `bitset<_Nb >` & **\_Unchecked\_set** (`size_t __pos, int __val`)
  - `bitset<_Nb >` & **\_Unchecked\_reset** (`size_t __pos`)
  - `bitset<_Nb >` & **\_Unchecked\_flip** (`size_t __pos`)
  - `bool _Unchecked_test` (`size_t __pos`) const
- 
- **reference operator[]** (`size_t __position`)
  - `bool operator[]` (`size_t __position`) const
- 
- `bool operator==` (const `bitset<_Nb >` &`__rhs`) const
  - `bool operator!=` (const `bitset<_Nb >` &`__rhs`) const
- 
- `bitset<_Nb >` **operator<<** (`size_t __position`) const
  - `bitset<_Nb >` **operator>>** (`size_t __position`) const

## Private Types

- typedef unsigned long **\_WordT**

## Private Member Functions

- `size_t _M_are_all_aux` () const
- `void _M_do_and` (const `_Base_bitset<_Nw >` &`__x`)
- `size_t _M_do_count` () const
- `size_t _M_do_find_first` (`size_t __not_found`) const
- `size_t _M_do_find_next` (`size_t __prev, size_t __not_found`) const

- void `_M_do_flip` ()
- void `_M_do_left_shift` (size\_t \_\_shift)
- void `_M_do_or` (const [\\_Base\\_bitset](#)<\_Nw > &\_\_x)
- void `_M_do_reset` ()
- void `_M_do_right_shift` (size\_t \_\_shift)
- void `_M_do_set` ()
- unsigned long long `_M_do_to_ullong` () const
- unsigned long `_M_do_to_ulong` () const
- void `_M_do_xor` (const [\\_Base\\_bitset](#)<\_Nw > &\_\_x)
- const `_WordT * _M_getdata` () const
- `_WordT _M_getword` (size\_t \_\_pos) const
- `_WordT & _M_getword` (size\_t \_\_pos)
- `_WordT & _M_hiword` ()
- `_WordT _M_hiword` () const
- bool `_M_is_any` () const
- bool `_M_is_equal` (const [\\_Base\\_bitset](#)<\_Nw > &\_\_x) const

### Static Private Member Functions

- static `_WordT _S_maskbit` (size\_t \_\_pos)
- static `size_t _S_whichbit` (size\_t \_\_pos)
- static `size_t _S_whichbyte` (size\_t \_\_pos)
- static `size_t _S_whichword` (size\_t \_\_pos)

### Private Attributes

- `_WordT _M_w` [`_Nw`]

### Friends

- class `hash`
- class `reference`

## 5.387.1 Detailed Description

`template<size_t _Nb> class std::bitset<_Nb >`

The `bitset` class represents a *fixed-size* sequence of bits. (Note that `bitset` does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let  $B$  be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage.  $B - NbB$  bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as *a simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index  $Nb-1$  in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long        a = 'a';
    bitset<10>  b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html> for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the `bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 709 of file `bitset`.

## 5.387.2 Constructor & Destructor Documentation

### 5.387.2.1 `template<size_t _Nb> std::bitset<_Nb>::bitset ( ) [inline]`

All bits set to zero.

Definition at line 803 of file `bitset`.

### 5.387.2.2 `template<size_t _Nb> std::bitset<_Nb>::bitset ( unsigned long long __val ) [inline]`

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 808 of file `bitset`.

### 5.387.2.3 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset<_Nb>::bitset ( const std::basic_string<_CharT, _Traits, _Alloc > & __s, size_t __position = 0 ) [inline, explicit]`

Use a subset of a string.

#### Parameters

*s* A string of *0* and *1* characters.

*position* Index of the first character in *s* to use; defaults to zero.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is bigger the size of *s*.

[\*std::invalid\\_argument\*](#) If a character appears in the string which is neither *0* nor *1*.

Definition at line 826 of file `bitset`.

References `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

### 5.387.2.4 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset<_Nb>::bitset ( const std::basic_string<_CharT, _Traits, _Alloc > & __s, size_t __position, size_t __n ) [inline]`

Use a subset of a string.

**Parameters**

- s* A string of *0* and *1* characters.
- position* Index of the first character in *s* to use.
- n* The number of characters to copy.

**Exceptions**

- std::out\_of\_range* If *pos* is bigger the size of *s*.
- std::invalid\_argument* If a character appears in the string which is neither *0* nor *1*.

Definition at line 848 of file `bitset`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.387.2.5** `template<size_t _Nb> std::bitset<_Nb>::bitset ( const char * __str ) [inline, explicit]`

Construct from a string.

**Parameters**

- str* A string of *0* and *1* characters.

**Exceptions**

- std::invalid\_argument* If a character appears in the string which is neither *0* nor *1*.

Definition at line 880 of file `bitset`.

**5.387.3 Member Function Documentation**

**5.387.3.1** `template<size_t _Nb> bool std::bitset<_Nb>::all ( ) const [inline]`

Tests whether all the bits are on.

**Returns**

- True if all the bits are set.

Definition at line 1267 of file `bitset`.

**5.387.3.2** `template<size_t _Nb> bool std::bitset<_Nb>::any ( ) const`  
`[inline]`

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

Definition at line 1275 of file `bitset`.

**5.387.3.3** `template<size_t _Nb> size_t std::bitset<_Nb>::count ( ) const`  
`[inline]`

Returns the number of bits which are set.

Definition at line 1227 of file `bitset`.

**5.387.3.4** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip (`  
`size_t __position ) [inline]`

Toggles a given bit to its opposite value.

**Parameters**

*position* The index of the bit.

**Exceptions**

[\*std::out\\_of\\_range\*](#) If *pos* is bigger the size of the set.

Definition at line 1067 of file `bitset`.

**5.387.3.5** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip ( )`  
`[inline]`

Toggles every bit to its opposite value.

Definition at line 1054 of file `bitset`.

**5.387.3.6** `template<size_t _Nb> bool std::bitset<_Nb>::none ( ) const`  
`[inline]`

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

Definition at line 1283 of file `bitset`.

**5.387.3.7** `template<size_t _Nb> bool std::bitset<_Nb>::operator!=( const bitset<_Nb> & __rhs ) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1242 of file `bitset`.

**5.387.3.8** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator&=( const bitset<_Nb> & __rhs ) [inline]`

Operations on bitsets.

**Parameters**

*rhs* A same-sized bitset.

These should be self-explanatory.

Definition at line 901 of file `bitset`.

**5.387.3.9** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb>::operator<<( size_t __position ) const [inline]`

Self-explanatory.

Definition at line 1289 of file `bitset`.

**5.387.3.10** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator<<=( size_t __position ) [inline]`

Operations on bitsets.

**Parameters**

*position* The number of places to shift.

These should be self-explanatory.

Definition at line 930 of file `bitset`.

**5.387.3.11** `template<size_t _Nb> bool std::bitset<_Nb >::operator==( const bitset<_Nb > & __rhs ) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1238 of file `bitset`.

**5.387.3.12** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb >::operator>>( size_t __position ) const [inline]`

Self-explanatory.

Definition at line 1293 of file `bitset`.

**5.387.3.13** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb >::operator>>=( size_t __position ) [inline]`

Operations on bitsets.

#### Parameters

*position* The number of places to shift.

These should be self-explanatory.

Definition at line 943 of file `bitset`.

**5.387.3.14** `template<size_t _Nb> bool std::bitset<_Nb >::operator[]( size_t __position ) const [inline]`

Array-indexing support.

#### Parameters

*position* Index into the `bitset`.

#### Returns

A `bool` for a *const* `bitset`. For non-const bitsets, an instance of the reference proxy class.

#### Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. `-pme` The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. `-pme`

Definition at line 1099 of file `bitset`.

**5.387.3.15** `template<size_t _Nb> reference std::bitset< _Nb >::operator[] ( size_t __position ) [inline]`

Array-indexing support.

#### Parameters

*position* Index into the `bitset`.

#### Returns

A `bool` for a *const* `bitset`. For non-const `bitsets`, an instance of the reference proxy class.

#### Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. `-pme` The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. `-pme`

Definition at line 1095 of file `bitset`.

**5.387.3.16** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::operator^=( const bitset<_Nb > & __rhs ) [inline]`

Operations on `bitsets`.

#### Parameters

*rhs* A same-sized `bitset`.

These should be self-explanatory.

Definition at line 915 of file `bitset`.

**5.387.3.17** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator|= ( const bitset<_Nb> & __rhs ) [inline]`

Operations on bitsets.

#### Parameters

*rhs* A same-sized bitset.

These should be self-explanatory.

Definition at line 908 of file bitset.

**5.387.3.18** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb>::operator~ ( ) const [inline]`

See the no-argument [flip\(\)](#).

Definition at line 1076 of file bitset.

**5.387.3.19** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset ( ) [inline]`

Sets every bit to false.

Definition at line 1029 of file bitset.

**5.387.3.20** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset ( size_t __position ) [inline]`

Sets a given bit to false.

#### Parameters

*position* The index of the bit.

#### Exceptions

[std::out\\_of\\_range](#) If *pos* is bigger the size of the set.

Same as writing `set (pos, false)`.

Definition at line 1043 of file bitset.

**5.387.3.21** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::set ( )`  
`[inline]`

Sets every bit to true.

Definition at line 1004 of file bitset.

**5.387.3.22** `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb >::set (`  
`size_t __position, bool __val = true ) [inline]`

Sets a given bit to a particular value.

#### Parameters

*position* The index of the bit.

*val* Either true or false, defaults to true.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is bigger the size of the set.

Definition at line 1018 of file bitset.

**5.387.3.23** `template<size_t _Nb> size_t std::bitset< _Nb >::size ( ) const`  
`[inline]`

Returns the total number of bits.

Definition at line 1232 of file bitset.

**5.387.3.24** `template<size_t _Nb> bool std::bitset< _Nb >::test ( size_t`  
`__position ) const [inline]`

Tests the value of a bit.

#### Parameters

*position* The index of a bit.

#### Returns

The value at *pos*.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is bigger the size of the set.

Definition at line 1253 of file bitset.

**5.387.3.25** `template<size_t _Nb> template<class _CharT, class _Traits, class _Alloc > std::basic_string<_CharT, _Traits, _Alloc> std::bitset<_Nb >::to_string ( ) const [inline]`

Returns a character interpretation of the bitset.

#### Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1129 of file bitset.

**5.387.3.26** `template<size_t _Nb> unsigned long std::bitset<_Nb >::to_ulong ( ) const [inline]`

Returns a numerical interpretation of the bitset.

#### Returns

The integral equivalent of the bits.

#### Exceptions

*std::overflow\_error* If there are too many bits to be represented in an unsigned long.

Definition at line 1110 of file bitset.

The documentation for this class was generated from the following file:

- [bitset](#)

## 5.388 `std::bitset<_Nb >::reference` Class Reference

### Public Member Functions

- `reference` (`bitset &__b`, `size_t __pos`)
- `reference & flip ()`
- `operator bool () const`
- `reference & operator= (const reference &__j)`
- `reference & operator= (bool __x)`
- `bool operator~ () const`

## Friends

- class `bitset`

### 5.388.1 Detailed Description

`template<size_t _Nb> class std::bitset<_Nb >::reference`

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 740 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

## 5.389 `std::cauchy_distribution<_RealType >` Class Template Reference

A `cauchy_distribution` random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `cauchy_distribution` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `cauchy_distribution` (const [param\\_type](#) &\_\_p)

- `_RealType a () const`
- `_RealType b () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const  
param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

### 5.389.1 Detailed Description

`template<typename _RealType = double> class std::cauchy_distribution< _RealType >`

A `cauchy_distribution` random number distribution. The formula for the normal probability mass function is  $p(x|a, b) = (\pi b (1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2704 of file random.h.

### 5.389.2 Member Typedef Documentation

**5.389.2.1** `template<typename _RealType = double> typedef _RealType  
std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2711 of file random.h.

### 5.389.3 Member Function Documentation

**5.389.3.1** `template<typename _RealType = double> result_type  
std::cauchy_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2795 of file random.h.

**5.389.3.2** `template<typename _RealType = double> result_type  
std::cauchy_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2788 of file `random.h`.

**5.389.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::cauchy_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2803 of file `random.h`.

References `std::cauchy_distribution<_RealType>::operator()`, and `std::cauchy_distribution<_RealType>::param()`.

Referenced by `std::cauchy_distribution<_RealType>::operator()`.

**5.389.3.4** `template<typename _RealType = double> void  
std::cauchy_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2781 of file `random.h`.

**5.389.3.5** `template<typename _RealType = double> param_type  
std::cauchy_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2773 of file `random.h`.

Referenced by `std::cauchy_distribution<_RealType>::operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.389.3.6** `template<typename _RealType = double> void  
std::cauchy_distribution<_RealType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2755 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.390 `std::cauchy_distribution<_RealType >::param_type` Struct Reference

### Public Types

- typedef `cauchy_distribution<_RealType >` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a=_RealType(0)`, `_RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.390.1 Detailed Description

`template<typename _RealType = double> struct std::cauchy_distribution<_RealType >::param_type`

Parameter type.

Definition at line 2713 of file random.h.

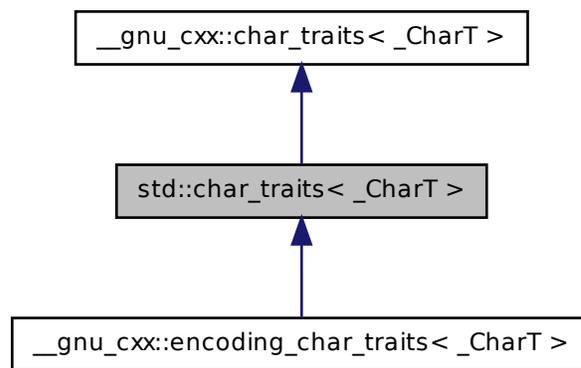
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.391 std::char\_traits< \_CharT > Struct Template Reference

Basis for explicit traits specializations.

Inheritance diagram for std::char\_traits< \_CharT >:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types< _CharT >::int_type` **int\_type**
- typedef `_Char_types< _CharT >::off_type` **off\_type**
- typedef `_Char_types< _CharT >::pos_type` **pos\_type**
- typedef `_Char_types< _CharT >::state_type` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)

- static int\_type **eof** ()
- static bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static std::size\_t **length** (const char\_type \* \_\_s)
- static bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static char\_type **to\_char\_type** (const int\_type &\_\_c)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.391.1 Detailed Description

**template<class \_CharT> struct std::char\_traits< \_CharT >**

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod\\_char\\_traits.h](#).

Definition at line 231 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.392 `std::char_traits< __gnu_cxx::character< V, I, S > >` Struct Template Reference

`char_traits<__gnu_cxx::character>` specialization.

### Public Types

- typedef `__gnu_cxx::character< V, I, S >` **char\_type**

- typedef `char_type::int_type` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `fpos< state_type >` **pos\_type**
- typedef `char_type::state_type` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type &__c1`, const `char_type &__c2`)
- static `char_type * assign` (`char_type *__s`, `size_t __n`, `char_type __a`)
- static int **compare** (const `char_type *__s1`, const `char_type *__s2`, `size_t __n`)
- static `char_type * copy` (`char_type *__s1`, const `char_type *__s2`, `size_t __n`)
- static `int_type eof` ()
- static bool **eq** (const `char_type &__c1`, const `char_type &__c2`)
- static bool **eq\_int\_type** (const `int_type &__c1`, const `int_type &__c2`)
- static const `char_type * find` (const `char_type *__s`, `size_t __n`, const `char_type &__a`)
- static `size_t length` (const `char_type *__s`)
- static bool **lt** (const `char_type &__c1`, const `char_type &__c2`)
- static `char_type * move` (`char_type *__s1`, const `char_type *__s2`, `size_t __n`)
- static `int_type not_eof` (const `int_type &__c`)
- static `char_type to_char_type` (const `int_type &__i`)
- static `int_type to_int_type` (const `char_type &__c`)

#### 5.392.1 Detailed Description

`template<typename V, typename I, typename S> struct std::char_traits< __gnu_cxx::character< V, I, S > >`

`char_traits<__gnu_cxx::character>` specialization.

Definition at line 88 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.393 `std::char_traits< char >` Struct Template Reference

21.1.3.1 `char_traits` specializations

## Public Types

- typedef char **char\_type**
- typedef int **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [streampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

## Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static int\_type **eof** ()
- static bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static char\_type **to\_char\_type** (const int\_type &\_\_c)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

### 5.393.1 Detailed Description

**template<> struct std::char\_traits< char >**

21.1.3.1 [char\\_traits](#) specializations

Definition at line 237 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.394 std::char\_traits< wchar\_t > Struct Template Reference

21.1.3.2 [char\\_traits](#) specializations

## 5.395 `std::chi_squared_distribution<_RealType>` Class Template Reference 2149

### Public Types

- typedef `wchar_t` **char\_type**
- typedef `wint_t` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `wstreampos` **pos\_type**
- typedef `mbstate_t` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **assign** (`char_type` \*\_\_s, `size_t` \_\_n, `char_type` \_\_a)
- static `int` **compare** (const `char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static `char_type` \* **copy** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static `int_type` **eof** ()
- static `bool` **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `bool` **eq\_int\_type** (const `int_type` &\_\_c1, const `int_type` &\_\_c2)
- static const `char_type` \* **find** (const `char_type` \*\_\_s, `size_t` \_\_n, const `char_type` &\_\_a)
- static `size_t` **length** (const `char_type` \*\_\_s)
- static `bool` **lt** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **move** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static `int_type` **not\_eof** (const `int_type` &\_\_c)
- static `char_type` **to\_char\_type** (const `int_type` &\_\_c)
- static `int_type` **to\_int\_type** (const `char_type` &\_\_c)

### 5.394.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

21.1.3.2 `char_traits` specializations

Definition at line 308 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.395 `std::chi_squared_distribution<_RealType>` Class Template Reference

A `chi_squared_distribution` random number distribution.

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [chi\\_squared\\_distribution](#) (`_RealType __n=_RealType(1)`)
- [chi\\_squared\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- `_RealType` [n](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator](#)() (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator](#)() (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type](#) [param](#) () const
- void [reset](#) ()

## Friends

- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_ostream](#)< `_CharT`, `_Traits` > & [operator](#)<< ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &, const [std::chi\\_squared\\_distribution](#)< `_RealType1` > &)
- template<typename `_RealType1` >  
bool [operator](#)== (const [std::chi\\_squared\\_distribution](#)< `_RealType1` > &\_\_d1, const [std::chi\\_squared\\_distribution](#)< `_RealType1` > &\_\_d2)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & [operator](#)>> ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &, [std::chi\\_squared\\_distribution](#)< `_RealType1` > &)

### 5.395.1 Detailed Description

template<typename `_RealType = double`> class [std::chi\\_squared\\_distribution](#)<  
`_RealType` >

A [chi\\_squared\\_distribution](#) random number distribution. The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

## 5.395 std::chi\_squared\_distribution<\_RealType > Class Template Reference 2151

Definition at line 2539 of file random.h.

### 5.395.2 Member Typedef Documentation

**5.395.2.1** `template<typename _RealType = double> typedef _RealType  
std::chi_squared_distribution<_RealType >::result_type`

The type of the range of the distribution.

Definition at line 2546 of file random.h.

### 5.395.3 Member Function Documentation

**5.395.3.1** `template<typename _RealType = double> result_type  
std::chi_squared_distribution<_RealType >::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2619 of file random.h.

**5.395.3.2** `template<typename _RealType = double> result_type  
std::chi_squared_distribution<_RealType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2612 of file random.h.

**5.395.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::chi_squared_distribution<_RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2627 of file random.h.

**5.395.3.4** `template<typename _RealType = double> param_type  
std::chi_squared_distribution<_RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2597 of file random.h.

**5.395.3.5** `template<typename _RealType = double> void  
std::chi_squared_distribution<_RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2605 of file random.h.

**5.395.3.6** `template<typename _RealType = double> void  
std::chi_squared_distribution<_RealType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2583 of file random.h.

References `std::gamma_distribution<_RealType >::reset()`.

## 5.395.4 Friends And Related Function Documentation

**5.395.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits > & , const  
std::chi_squared_distribution<_RealType1 > & ) [friend]`

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `chi_squared_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

## 5.396 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

2153

```
5.395.4.2 template<typename _RealType = double> template<typename _RealType1 > bool operator==( const std::chi_squared_distribution<_RealType1 > & __d1, const std::chi_squared_distribution<_RealType1 > & __d2 ) [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2647 of file `random.h`.

```
5.395.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & , std::chi_squared_distribution<_RealType1 > & ) [friend]
```

Extracts a `chi_squared_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A `chi_squared_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.396 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `chi_squared_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType` `__n`=`_RealType`(1))
- `_RealType` `n` () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.396.1 Detailed Description

`template<typename _RealType = double> struct std::chi_squared_distribution<_RealType >::param_type`

Parameter type.

Definition at line 2548 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.397 std::chrono::duration< \_Rep, \_Period > Struct Template Reference

duration

### Public Types

- typedef `_Period` **period**
- typedef `_Rep` **rep**

### Public Member Functions

- `template<typename _Rep2 , typename = typename enable_if<is_convertible<_Rep2, rep>::value && (treat_as_floating_point<rep>::value || !treat_as_floating_point<_Rep2>::value)>::type>`  
**duration** (const `_Rep2` &\_\_rep)
- **duration** (const [duration](#) &)
- `template<typename _Rep2 , typename _Period2 , typename = typename enable_if<treat_as_floating_point<rep>::value || (ratio_divide<_Period2, period>::type::den == 1 && !treat_as_floating_point<_Rep2>::value)>::type>`  
**duration** (const [duration](#)< `_Rep2`, `_Period2` > &\_\_d)
- `rep` **count** () const
- `template<typename _Rep2 = rep>`  
`enable_if<!treat_as_floating_point<_Rep2 >::value, duration & >::type` **operator%=** (const `rep` &\_\_rhs)

- `template<typename _Rep2 = rep> enable_if<!treat_as_floating_point<_Rep2>::value, duration &>::type operator%=(const duration &__d)`
- `duration & operator*=(const rep &__rhs)`
- `duration operator+ () const`
- `duration & operator++ ()`
- `duration operator++ (int)`
- `duration & operator+=(const duration &__d)`
- `duration operator- () const`
- `duration & operator-- ()`
- `duration operator-- (int)`
- `duration & operator-=(const duration &__d)`
- `duration & operator/=(const rep &__rhs)`
- `duration & operator=(const duration &)`
- `static_assert (!__is_duration<_Rep>::value, "rep cannot be a duration")`
- `static_assert (_Period::num > 0, "period must be positive")`
- `static_assert (__is_ratio<_Period>::value, "period must be a specialization of ratio")`

## Static Public Member Functions

- static const `duration max ()`
- static const `duration min ()`
- static const `duration zero ()`

### 5.397.1 Detailed Description

`template<typename _Rep, typename _Period> struct std::chrono::duration<_Rep, _Period>`

`duration`

Definition at line 201 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.398 `std::chrono::duration_values<_Rep>` Struct Template Reference

[duration\\_values](#)

## Static Public Member Functions

- static const `_Rep` **max** ()
- static const `_Rep` **min** ()
- static const `_Rep` **zero** ()

### 5.398.1 Detailed Description

`template<typename _Rep> struct std::chrono::duration_values< _Rep >`

[duration\\_values](#)

Definition at line 174 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.399 `std::chrono::system_clock` Struct Reference

[system\\_clock](#)

### Public Types

- typedef `chrono::seconds` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef `chrono::time_point< system_clock, duration >` **time\_point**

### Static Public Member Functions

- static `time_point` **from\_time\_t** (std::time\_t \_\_t)
- static `time_point` **now** () throw ()
- static std::time\_t **to\_time\_t** (const `time_point` &\_\_t)

### Static Public Attributes

- static const bool **is\_monotonic**

## 5.400 std::chrono::time\_point< \_Clock, \_Duration > Struct Template Reference

### 5.399.1 Detailed Description

[system\\_clock](#)

Definition at line 628 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.400 std::chrono::time\_point< \_Clock, \_Duration > Struct Template Reference

[time\\_point](#)

### Public Types

- typedef `_Clock` **clock**
- typedef `_Duration` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

### Public Member Functions

- **time\_point** (const duration &\_\_dur)
- template<typename `_Duration2` >  
**time\_point** (const [time\\_point](#)< clock, `_Duration2` > &\_\_t)
- **time\_point** & **operator+=** (const duration &\_\_dur)
- **time\_point** & **operator-=** (const duration &\_\_dur)
- duration **time\_since\_epoch** () const

### Static Public Member Functions

- static const [time\\_point](#) **max** ()
- static const [time\\_point](#) **min** ()

### 5.400.1 Detailed Description

```
template<typename _Clock, typename _Duration> struct std::chrono::time_
point< _Clock, _Duration >
```

[time\\_point](#)

Definition at line 492 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.401 `std::chrono::treat_as_floating_point< _Rep >` Struct Template Reference

[treat\\_as\\_floating\\_point](#)

Inherits `is_floating_point< _Rep >`.

### 5.401.1 Detailed Description

`template<typename _Rep> struct std::chrono::treat_as_floating_point< _Rep >`

[treat\\_as\\_floating\\_point](#)

Definition at line 168 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

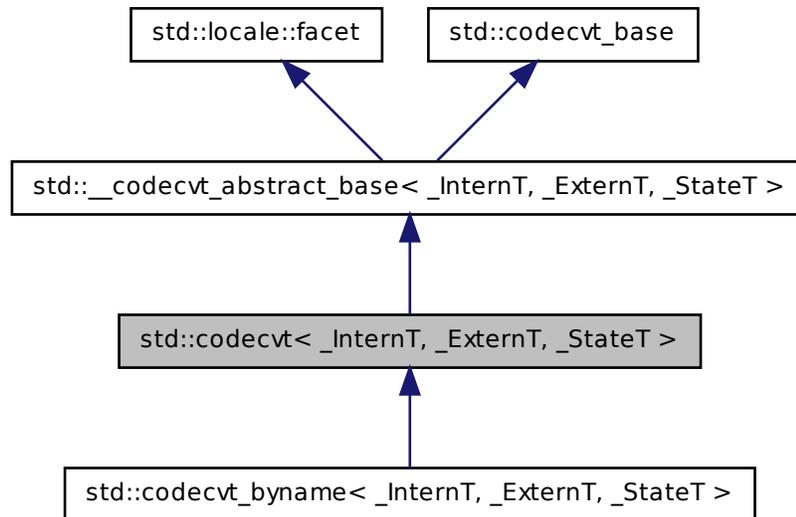
## 5.402 `std::codecvt< _InternT, _ExternT, _StateT >` Class Template Reference

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

## 5.402 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, _StateT >`:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- `codecvt` (`size_t __refs=0`)
- `codecvt` (`__c_locale __cloc, size_t __refs=0`)
- `bool always_noconv` () const throw ()
- `int encoding` () const throw ()
- `result in` (`state_type &__state, const extern_type *__from, const extern_type * __from_end, const extern_type *&__from_next, intern_type *__to, intern_type * __to_end, intern_type *&__to_next`) const

- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_GLIBCXX\_CONST const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## Friends

- class `locale::_Impl`

### 5.402.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT> class
std::codecvt<_InternT, _ExternT, _StateT >
```

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

Definition at line 275 of file `codecvt.h`.

### 5.402.2 Member Function Documentation

**5.402.2.1** `template<typename _InternT, typename _ExternT, typename _StateT > virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [protected, virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >`.

**5.402.2.2** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::underflow()`.

**5.402.2.3** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

## 5.402 std::codecvt<\_InternT, \_ExternT, \_StateT > Class Template Reference 2163

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

codecvt\_base::result.

Definition at line 115 of file codecvt.h.

```
5.402.2.4 template<typename _InternT, typename _ExternT, typename  
_StateT> result std::__codecvt_abstract_base<_InternT, _ExternT,  
_StateT>::unshift ( state_type & __state, extern_type * __to,  
extern_type * __to_end, extern_type *& __to_next ) const  
[inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.403 `std::codecvt< _InternT, _ExternT, encoding_state >` Class Template Reference

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Inheritance diagram for `std::codecvt< _InternT, _ExternT, encoding_state >`:



### Public Types

- typedef `state_type::descriptor_type` **descriptor\_type**

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (state\_type &\_\_enc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const =0
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()

- virtual result `do_out` (`state_type` &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0
- virtual result `do_out` (`state_type` &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (`state_type` &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (`state_type` &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0

### Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` &\_\_cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` &\_\_cloc, const char \*\_\_s, `__c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`__c_locale` &\_\_cloc)
- static `__c_locale` `_S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` \_\_cloc, const char \*\_\_s)

### Friends

- class `locale::Impl`

### 5.403.1 Detailed Description

`template<typename _InternT, typename _ExternT> class std::codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Definition at line 226 of file `codecvt_specializations.h`.

## 5.403.2 Member Function Documentation

**5.403.2.1** `virtual result std::_codecvt_abstract_base<_InternT, _ExternT, encoding_state >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const` [`protected`, `pure virtual`, `inherited`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

### See also

`do_out` for more information.

**5.403.2.2** `result std::_codecvt_abstract_base<_InternT, _ExternT, encoding_state >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const` [`inline`, `inherited`]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

codecvt\_base::result.

Definition at line 195 of file codecvt.h.

**5.403.2.3** `result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

**5.403.2.4** `result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const`  
[`inline`, `inherited`]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 154 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.404 `std::codecvt< char, char, mbstate_t >` Class Template Reference

class `codecvt<char, char, mbstate_t>` specialization.

Inheritance diagram for `std::codecvt< char, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- `codecvt` (size\_t \_\_refs=0)
- `codecvt` (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `bool` `do_always_noconv` () const throw ()
- virtual `int` `do_encoding` () const throw ()
- virtual result `do_in` (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const =0
- virtual result `do_in` (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual `int` `do_length` (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual `int` `do_length` (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual `int` `do_max_length` () const throw ()
- virtual result `do_out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0
- virtual result `do_out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0

## Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static `void` `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static `void` `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale` `_S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

## Friends

- class `locale::_Impl`

### 5.404.1 Detailed Description

`template<> class std::codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

Definition at line 337 of file `codecvt.h`.

### 5.404.2 Member Function Documentation

**5.404.2.1 virtual result `std::__codecvt_abstract_base< char , char , mbstate_t >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [protected, pure virtual, inherited]**

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

`out` for more information.

**5.404.2.2 result `std::__codecvt_abstract_base< char , char , mbstate_t >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const` [inline, inherited]**

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted

character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

```
5.404.2.3 result std::__codecvt_abstract_base< char , char , mbstate_t >::out (
    state_type & __state, const intern_type * __from, const intern_type
    * __from_end, const intern_type *& __from_next, extern_type *
    __to, extern_type * __to_end, extern_type *& __to_next ) const
    [inline, inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

**5.404.2.4** `result std::__codecvt_abstract_base< char , char , mbstate_t >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [inline, inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## 5.405 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference 2175

### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.405 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- `codecvt` (`size_t __refs=0`)
- `codecvt` (`__c_locale __cloc, size_t __refs=0`)
- bool **always\_noconv** () const throw ()

- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const =0
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const =0
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const =0

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_codecvt`

## Friends

- class `locale::_Impl`

### 5.405.1 Detailed Description

`template<> class std::codecvt< wchar_t, char, mbstate_t >`

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Definition at line 395 of file `codecvt.h`.

### 5.405.2 Member Function Documentation

**5.405.2.1** virtual result `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [protected, pure virtual, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

out for more information.

```

5.405.2.2 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t
>::in ( state_type & __state, const extern_type * __from, const
extern_type * __from_end, const extern_type *& __from_next,
intern_type * __to, intern_type * __to_end, intern_type *&
__to_next ) const [inline, inherited]

```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.

*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

```
5.405.2.3 result std::_codecvt_abstract_base< wchar_t, char, mbstate_t
>::out ( state_type & __state, const intern_type * __from, const
intern_type * __from_end, const intern_type *& __from_next,
extern_type * __to, extern_type * __to_end, extern_type *&
__to_next ) const [inline, inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.

*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```
5.405.2.4 result std::_codecvt_abstract_base< wchar_t , char , mbstate_t
>::unshift ( state_type & __state, extern_type * __to, extern_type
* __to_end, extern_type *& __to_next ) const [inline,
inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

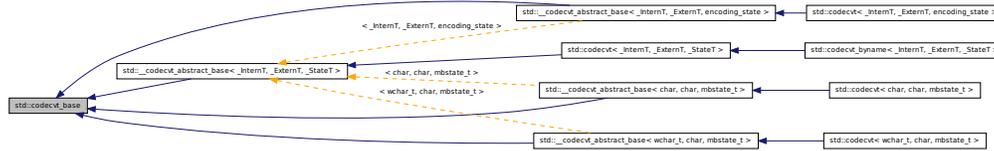
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.406 std::codecvt\_base Class Reference

Empty base class for `codecvt` facet [22.2.1.5].

Inheritance diagram for `std::codecvt_base`:



## Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

### 5.406.1 Detailed Description

Empty base class for `codecvt` facet [22.2.1.5].

Definition at line 45 of file `codecvt.h`.

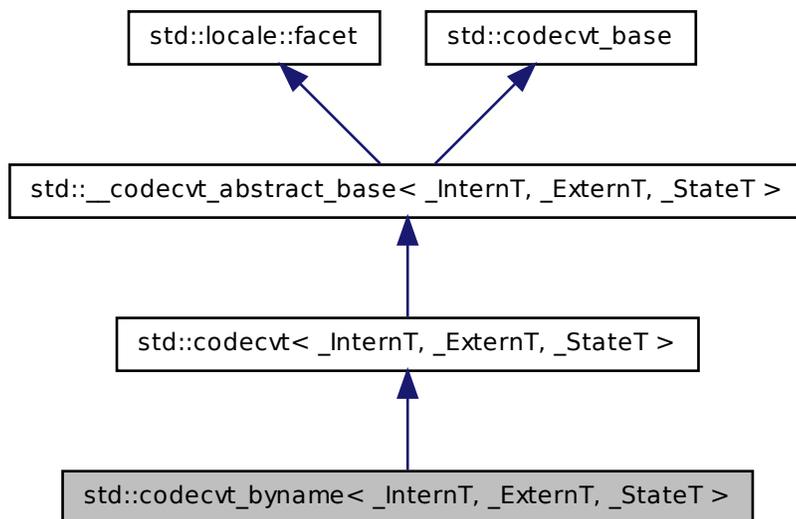
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.407 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference

class `codecvt_byname` [22.2.1.6].

Inheritance diagram for `std::codecvt_byname<_InternT, _ExternT, _StateT >`:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt\_byname** (`const char *__s, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const

- int `max_length` () const throw ()
- result `out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- result `unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual bool `do_always_noconv` () const throw ()
- virtual int `do_encoding` () const throw ()
- virtual result `do_in` (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int `do_length` (state\_type &, const extern\_type \*\_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- virtual int `do_max_length` () const throw ()
- virtual result `do_out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale` `_S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

## Friends

- class `locale::_Impl`

### 5.407.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt_byname< _InternT, _ExternT, _StateT >`

class `codecvt_byname` [22.2.1.6].

Definition at line 455 of file `codecvt.h`.

### 5.407.2 Member Function Documentation

**5.407.2.1** `template<typename _InternT , typename _ExternT , typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [protected, virtual, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`.

**5.407.2.2** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

## 5.407 `std::codecvt_byname< _InternT, _ExternT, _StateT >` Class Template Reference 2185

---

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.407.2.3** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.

*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```
5.407.2.4 template<typename _InternT, typename _ExternT, typename
    _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
    _StateT >::unshift ( state_type & __state, extern_type * __to,
    extern_type * __to_end, extern_type *& __to_next ) const
    [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

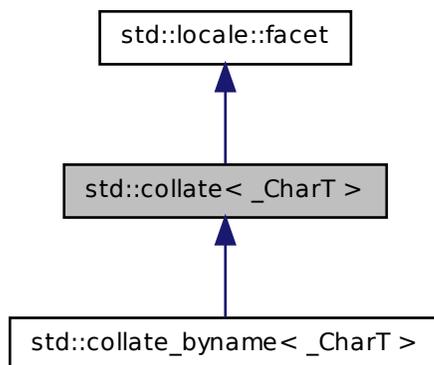
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.408 `std::collate<_CharT >` Class Template Reference

Facet for localized string comparison.

Inheritance diagram for `std::collate<_CharT>`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- [collate](#) (size\_t \_\_refs=0)
- [collate](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- int [compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const

- long `hash` (const `_CharT` \*`__lo`, const `_CharT` \*`__hi`) const
- `string_type transform` (const `_CharT` \*`__lo`, const `_CharT` \*`__hi`) const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~collate` ()
- virtual int `do_compare` (const `_CharT` \*`__lo1`, const `_CharT` \*`__hi1`, const `_CharT` \*`__lo2`, const `_CharT` \*`__hi2`) const
- virtual long `do_hash` (const `_CharT` \*`__lo`, const `_CharT` \*`__hi`) const
- virtual `string_type do_transform` (const `_CharT` \*`__lo`, const `_CharT` \*`__hi`) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale` &`__cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale` &`__cloc`, const char \*`__s`, `__c_locale` `__old`=0)
- static void `_S_destroy_c_locale` (`__c_locale` &`__cloc`)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \*`__s`)

### Protected Attributes

- `__c_locale _M_c_locale_collate`

### Friends

- class `locale::_Impl`

## 5.408.1 Detailed Description

`template<typename _CharT> class std::collate<_CharT>`

Facet for localized string comparison. This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 610 of file locale\_classes.h.

## 5.408.2 Member Typedef Documentation

### 5.408.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::collate\\_byname<\\_CharT>](#).

Definition at line 616 of file locale\_classes.h.

### 5.408.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::collate\\_byname<\\_CharT>](#).

Definition at line 617 of file locale\_classes.h.

## 5.408.3 Constructor & Destructor Documentation

### 5.408.3.1 `template<typename _CharT> std::collate<_CharT>::collate ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 637 of file locale\_classes.h.

### 5.408.3.2 `template<typename _CharT> std::collate<_CharT>::collate ( __c_locale __cloc, size_t __refs = 0 ) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

- cloc* The C locale.  
*refs* Passed to the base facet class.

Definition at line 651 of file locale\_classes.h.

**5.408.3.3** `template<typename _CharT> virtual std::collate<_CharT>  
>::~collate( ) [inline, protected, virtual]`

Destructor.

Definition at line 714 of file locale\_classes.h.

**5.408.4 Member Function Documentation**

**5.408.4.1** `template<typename _CharT> int std::collate<_CharT>::compare  
( const _CharT * __lo1, const _CharT * __hi1, const _CharT *  
__lo2, const _CharT * __hi2 ) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do\\_compare\(\)](#).

**Parameters**

- lo1* Start of string 1.  
*hi1* End of string 1.  
*lo2* Start of string 2.  
*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 668 of file locale\_classes.h.

**5.408.4.2** `template<typename _CharT> int std::collate<_CharT>  
>::~do_compare( const _CharT * __lo1, const _CharT *  
__hi1, const _CharT * __lo2, const _CharT * __hi2 ) const  
[protected, virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See also**

[compare\(\)](#).

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

**5.408.4.3** `template<typename _CharT> long std::collate<_CharT>  
>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const  
[protected, virtual]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

*lo* Start of string.

*hi* End of string.

**Returns**

Hash value.

Definition at line 229 of file locale\_classes.tcc.

**5.408.4.4** `template<typename _CharT> collate<_CharT>::string_type  
std::collate<_CharT>::do_transform ( const _CharT * __lo, const  
_CharT * __hi ) const [protected, virtual]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `std::basic_string<_CharT, _Traits, _Alloc>::length()`, and `std::basic_string<_CharT, _Traits, _Alloc>::push_back()`.

**5.408.4.5** `template<typename _CharT> long std::collate<_CharT>::hash ( const _CharT * __lo, const _CharT * __hi ) const [inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

**Parameters**

*lo* Start of string.

*hi* End of string.

**Returns**

Hash value.

Definition at line 701 of file locale\_classes.h.

**5.408.4.6** `template<typename _CharT> string_type std::collate<_CharT>::transform ( const _CharT * __lo, const _CharT * __hi ) const [inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do\\_transform\(\)](#).

**Parameters**

*lo* Start of string.

*hi* End of string.

**Returns**

Transformed string\_type.

Definition at line 687 of file locale\_classes.h.

## 5.408.5 Member Data Documentation

### 5.408.5.1 `template<typename _CharT> locale::id std::collate<_CharT >::id` `[static]`

Numpunct facet id.

Definition at line 627 of file locale\_classes.h.

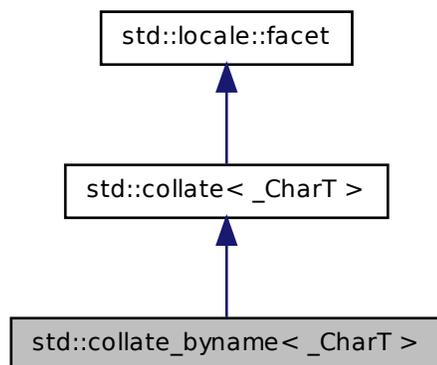
The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 5.409 `std::collate_byname<_CharT >` Class Template Reference

class [collate\\_byname](#) [22.2.4.2].

Inheritance diagram for std::collate\_byname< \_CharT >:



## Public Types

- typedef `_CharT` `char_type`
- typedef `basic_string< _CharT >` `string_type`

## Public Member Functions

- `collate_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- `int _M_compare` (const \_CharT \*, const \_CharT \*) const throw ()
- `template<>`  
`int _M_compare` (const char \*, const char \*) const throw()
- `template<>`  
`int _M_compare` (const wchar\_t \*, const wchar\_t \*) const throw()
- `size_t _M_transform` (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- `template<>`  
`size_t _M_transform` (char \*, const char \*, size\_t) const throw()
- `template<>`  
`size_t _M_transform` (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- `int compare` (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- `long hash` (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- `string_type transform` (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static \_GLIBCXX\_CONST const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- [\\_\\_c\\_locale \\_M\\_c\\_locale\\_collate](#)

## Friends

- class [locale::\\_Impl](#)

### 5.409.1 Detailed Description

`template<typename _CharT> class std::collate_byname< _CharT >`

class [collate\\_byname](#) [22.2.4.2].

Definition at line 786 of file `locale_classes.h`.

## 5.409.2 Member Typedef Documentation

### 5.409.2.1 `template<typename _CharT> typedef _CharT std::collate_byname<_CharT>::char_type`

Public typedefs.

Reimplemented from [std::collate<\\_CharT>](#).

Definition at line 791 of file `locale_classes.h`.

### 5.409.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate_byname<_CharT>::string_type`

Public typedefs.

Reimplemented from [std::collate<\\_CharT>](#).

Definition at line 792 of file `locale_classes.h`.

## 5.409.3 Member Function Documentation

### 5.409.3.1 `template<typename _CharT> int std::collate<_CharT>::compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const [inline, inherited]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do\\_compare\(\)](#).

#### Parameters

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

#### Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 668 of file `locale_classes.h`.

```
5.409.3.2 template<typename _CharT> int std::collate< _CharT
>::do_compare ( const _CharT * __lo1, const _CharT *
__hi1, const _CharT * __lo2, const _CharT * __hi2 ) const
[protected, virtual, inherited]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See also**

[compare\(\)](#).

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.

```
5.409.3.3 template<typename _CharT> long std::collate< _CharT
>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const
[protected, virtual, inherited]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

*lo* Start of string.

*hi* End of string.

**Returns**

Hash value.

Definition at line 229 of file locale\_classes.tcc.

**5.409.3.4** `template<typename _CharT> collate< _CharT >::string_type  
std::collate< _CharT >::do_transform ( const _CharT * __lo, const  
_CharT * __hi ) const [protected, virtual, inherited]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

#### Parameters

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

#### Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

**5.409.3.5** `template<typename _CharT> long std::collate< _CharT >::hash (   
const _CharT * __lo, const _CharT * __hi ) const [inline,   
inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

#### Parameters

*lo* Start of string.

*hi* End of string.

#### Returns

Hash value.

Definition at line 701 of file locale\_classes.h.

**5.409.3.6** `template<typename _CharT> string_type std::collate< _CharT >::transform ( const _CharT * __lo, const _CharT * __hi ) const`  
`[inline, inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do\\_transform\(\)](#).

#### Parameters

- lo* Start of string.
- hi* End of string.

#### Returns

Transformed `string_type`.

Definition at line 687 of file `locale_classes.h`.

### 5.409.4 Member Data Documentation

**5.409.4.1** `template<typename _CharT> locale::id std::collate< _CharT >::id`  
`[static, inherited]`

Numpunct facet id.

Definition at line 627 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

### 5.410 `std::complex< _Tp >` Struct Template Reference

#### Public Types

- `typedef _Tp value_type`

#### Public Member Functions

- `complex (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())`

- template<typename \_Up >  
  **complex** (const **complex**< \_Up > &\_\_z)
- const **complex** & **\_\_rep** () const
- \_Tp **imag** () const
- void **imag** (\_Tp \_\_val)
- **complex**< \_Tp > & **operator\*=** (const \_Tp &)
- template<typename \_Up >  
  **complex**< \_Tp > & **operator\*=** (const **complex**< \_Up > &)
- **complex**< \_Tp > & **operator+=** (const \_Tp &\_\_t)
- template<typename \_Up >  
  **complex**< \_Tp > & **operator+=** (const **complex**< \_Up > &)
- **complex**< \_Tp > & **operator-=** (const \_Tp &\_\_t)
- template<typename \_Up >  
  **complex**< \_Tp > & **operator-=** (const **complex**< \_Up > &)
- template<typename \_Up >  
  **complex**< \_Tp > & **operator/=** (const **complex**< \_Up > &)
- **complex**< \_Tp > & **operator/=** (const \_Tp &)
- template<typename \_Up >  
  **complex**< \_Tp > & **operator=** (const **complex**< \_Up > &)
- **complex**< \_Tp > & **operator=** (const \_Tp &)
- void **real** (\_Tp \_\_val)
- \_Tp **real** () const

### 5.410.1 Detailed Description

**template<typename \_Tp> struct std::complex< \_Tp >**

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

#### Parameters

*Tp* Type of real and imaginary values.

Definition at line 122 of file complex.

### 5.410.2 Member Typedef Documentation

**5.410.2.1 template<typename \_Tp> typedef \_Tp std::complex< \_Tp >::value\_type**

Value typedef.

Definition at line 125 of file complex.

### 5.410.3 Constructor & Destructor Documentation

**5.410.3.1** `template<typename _Tp> std::complex<_Tp>::complex ( const  
_Tp & __r = Tp() , const _Tp & __i = Tp() ) [inline]`

Default constructor. First parameter is x, second parameter is y. /// Unspecified parameters default to 0.

Definition at line 129 of file complex.

**5.410.3.2** `template<typename _Tp> template<typename _Up >  
std::complex<_Tp>::complex ( const complex<_Up> & __z )  
[inline]`

Copy constructor.

Definition at line 136 of file complex.

### 5.410.4 Member Function Documentation

**5.410.4.1** `template<typename _Tp> complex<_Tp>& std::complex<_Tp>  
>::operator+=( const _Tp & __t ) [inline]`

Add *t* to this complex number.

Definition at line 179 of file complex.

**5.410.4.2** `template<typename _Tp> complex<_Tp>& std::complex<_Tp>  
>::operator-=( const _Tp & __t ) [inline]`

Subtract *t* from this complex number.

Definition at line 188 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

## 5.411 `std::condition_variable` Class Reference

[condition\\_variable](#)

## Public Types

- typedef `__native_type * native_handle_type`

## Public Member Functions

- `condition_variable` (const `condition_variable` &)
- `native_handle_type native_handle` ()
- void `notify_all` ()
- void `notify_one` ()
- `condition_variable & operator=` (const `condition_variable` &)
- template<typename `_Predicate` >  
void `wait` (`unique_lock`< `mutex` > &\_\_lock, `_Predicate` \_\_p)
- void `wait` (`unique_lock`< `mutex` > &\_\_lock)
- template<typename `_Rep`, typename `_Period`, typename `_Predicate` >  
bool `wait_for` (`unique_lock`< `mutex` > &\_\_lock, const `chrono::duration`< `_Rep`, `_Period` > &\_\_rtime, `_Predicate` \_\_p)
- template<typename `_Rep`, typename `_Period` >  
`cv_status` `wait_for` (`unique_lock`< `mutex` > &\_\_lock, const `chrono::duration`< `_Rep`, `_Period` > &\_\_rtime)
- template<typename `_Duration` >  
`cv_status` `wait_until` (`unique_lock`< `mutex` > &\_\_lock, const `chrono::time_point`< `__clock_t`, `_Duration` > &\_\_atime)
- template<typename `_Clock`, typename `_Duration` >  
`cv_status` `wait_until` (`unique_lock`< `mutex` > &\_\_lock, const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_atime)
- template<typename `_Clock`, typename `_Duration`, typename `_Predicate` >  
bool `wait_until` (`unique_lock`< `mutex` > &\_\_lock, const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_atime, `_Predicate` \_\_p)

### 5.411.1 Detailed Description

[condition\\_variable](#)

Definition at line 57 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.412 `std::condition_variable_any` Class Reference

[condition\\_variable\\_any](#)

## Public Types

- typedef `condition_variable::native_handle_type` **native\_handle\_type**

## Public Member Functions

- **condition\_variable\_any** (const [condition\\_variable\\_any](#) &)
- `native_handle_type` **native\_handle** ()
- void **notify\_all** ()
- void **notify\_one** ()
- [condition\\_variable\\_any](#) & **operator=** (const [condition\\_variable\\_any](#) &)
- template<typename `_Lock`, typename `_Predicate` >  
void **wait** (`_Lock` &\_\_lock, `_Predicate` \_\_p)
- template<typename `_Lock` >  
void **wait** (`_Lock` &\_\_lock)
- template<typename `_Lock`, typename `_Rep`, typename `_Period`, typename `_Predicate` >  
bool **wait\_for** (`_Lock` &\_\_lock, const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rtime, `_Predicate` \_\_p)
- template<typename `_Lock`, typename `_Rep`, typename `_Period` >  
[cv\\_status](#) **wait\_for** (`_Lock` &\_\_lock, const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rtime)
- template<typename `_Lock`, typename `_Clock`, typename `_Duration` >  
[cv\\_status](#) **wait\_until** (`_Lock` &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime)
- template<typename `_Lock`, typename `_Clock`, typename `_Duration`, typename `_Predicate` >  
bool **wait\_until** (`_Lock` &\_\_lock, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime, `_Predicate` \_\_p)

### 5.412.1 Detailed Description

[condition\\_variable\\_any](#)

Definition at line 166 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

### 5.413 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

`conditional`

## 5.414 `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>` Class Template Reference

### Public Types

- `typedef _Iftrue type`

### 5.413.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct  
std::conditional<_Cond, _Iftrue, _Iffalse >
```

`conditional`

Definition at line 369 of file `type_traits`.

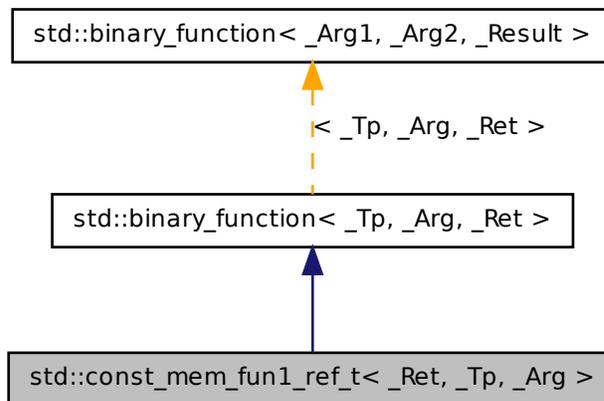
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.414 `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret(_Tp::*_pf)(_Arg)` const)
- `_Ret operator()` (const `_Tp` &`__r`, `_Arg` `__x`) const

### 5.414.1 Detailed Description

template<typename `_Ret`, typename `_Tp`, typename `_Arg`> class `std::const_mem_fun1_ref_t`< `_Ret`, `_Tp`, `_Arg` >

One of the [adaptors for member /// pointers](#).

Definition at line 650 of file `stl_function.h`.

### 5.414.2 Member Typedef Documentation

**5.414.2.1** typedef `_Tp` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`first_argument_type` [`inherited`]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.414.2.2** typedef `_Ret` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.414.2.3** typedef `_Arg` `std::binary_function`< `_Tp`, `_Arg`, `_Ret` >::`second_argument_type` [`inherited`]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this class was generated from the following file:

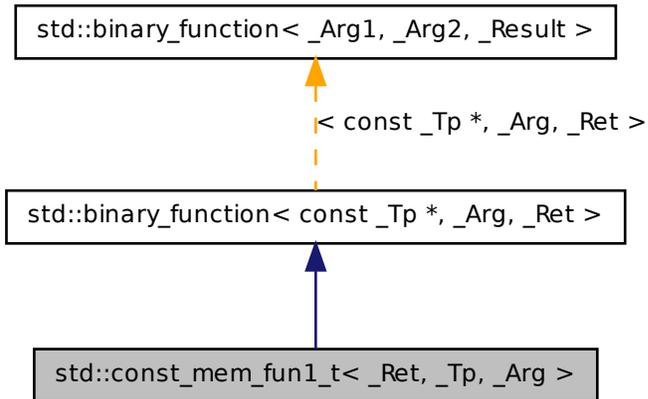
## 5.415 `std::const_mem_fun1_t< _Ret, _Tp, _Arg >` Class Template Reference 2207

- [stl\\_function.h](#)

### 5.415 `std::const_mem_fun1_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`:



#### Public Types

- typedef `const_Tp *` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

#### Public Member Functions

- `const_mem_fun1_t` (`_Ret`(`_Tp::*` `__pf`)(`_Arg`) `const`)
- `_Ret operator()` (`const_Tp *` `__p`, `_Arg` `__x`) `const`

### 5.415.1 Detailed Description

**template<typename \_Ret, typename \_Tp, typename \_Arg> class std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >**

One of the [adaptors for member /// pointers](#).

Definition at line 614 of file `stl_function.h`.

### 5.415.2 Member Typedef Documentation

**5.415.2.1 typedef const \_Tp \* std::binary\_function< const \_Tp \* , \_Arg , \_Ret >::first\_argument\_type [inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.415.2.2 typedef \_Ret std::binary\_function< const \_Tp \* , \_Arg , \_Ret >::result\_type [inherited]**

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.415.2.3 typedef \_Arg std::binary\_function< const \_Tp \* , \_Arg , \_Ret >::second\_argument\_type [inherited]**

the type of the second argument

Definition at line 117 of file `stl_function.h`.

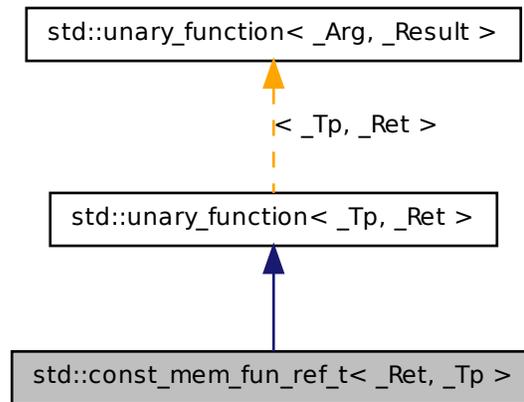
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.416 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_ref_t<_Ret, _Tp>`:



## Public Types

- typedef `_Tp` `argument_type`
- typedef `_Ret` `result_type`

## Public Member Functions

- `const_mem_fun_ref_t` (`_Ret`(`_Tp`::\* `__pf`)() const)
- `_Ret operator()` (const `_Tp` &`__r`) const

### 5.416.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::const_mem_fun_ref_t<_Ret, _Tp>
```

One of the [adaptors for member /// pointers](#).

Definition at line 578 of file `stl_function.h`.

## 5.416.2 Member Typedef Documentation

### 5.416.2.1 `typedef _Tp std::unary_function< _Tp , _Ret >::argument_type` [`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.416.2.2 `typedef _Ret std::unary_function< _Tp , _Ret >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

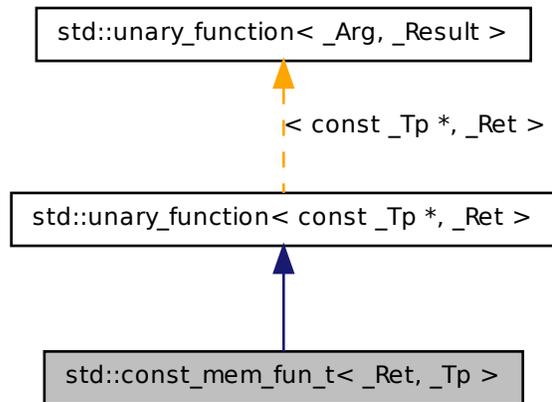
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.417 `std::const_mem_fun_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member `///` pointers](#).

Inheritance diagram for `std::const_mem_fun_t<_Ret, _Tp>`:



## Public Types

- typedef `const_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

## Public Member Functions

- `const_mem_fun_t` (`_Ret`(`_Tp::*_pf`)() const)
- `_Ret operator()` (`const_Tp *_p`) const

### 5.417.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::const_mem_fun_t<_Ret,
_Tp>
```

One of the [adaptors for member /// pointers](#).

Definition at line 542 of file `stl_function.h`.

## 5.417.2 Member Typedef Documentation

### 5.417.2.1 `typedef const _Tp * std::unary_function< const _Tp * , _Ret >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.417.2.2 `typedef _Ret std::unary_function< const _Tp * , _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

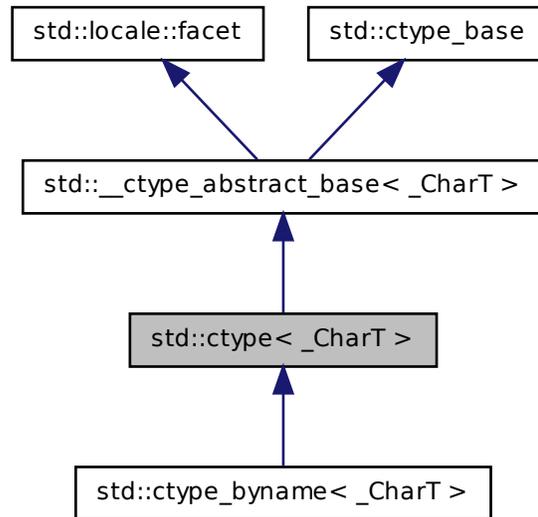
- [stl\\_function.h](#)

## 5.418 `std::ctype< _CharT >` Class Template Reference

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

Inheritance diagram for `std::ctype<_CharT>`:



## Public Types

- typedef `const int * __to_type`
- typedef `_CharT char_type`
- typedef `__ctype_abstract_base<_CharT>::mask mask`

## Public Member Functions

- `ctype` (`size_t __refs=0`)
- `bool is` (`mask __m, char_type __c`) `const`
- `const char_type * is` (`const char_type *__lo, const char_type *__hi, mask *__-vec`) `const`
- `const char_type * narrow` (`const char_type *__lo, const char_type *__hi, char __default, char *__to`) `const`
- `char narrow` (`char_type __c, char __default`) `const`
- `const char_type * scan_is` (`mask __m, const char_type *__lo, const char_type *__hi`) `const`

- const `char_type` \* `scan_not` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` `tolower` (`char_type` `__c`) const
- const `char_type` \* `tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- const `char_type` \* `toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` `toupper` (`char_type` `__c`) const
- const `char` \* `widen` (const `char` \* `__lo`, const `char` \* `__hi`, `char_type` \* `__to`) const
- `char_type` `widen` (`char` `__c`) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id` `id`
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool `do_is` (mask `__m`, `char_type` `__c`) const
- virtual const `char_type` \* `do_is` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, mask \* `__vec`) const
- virtual const `char_type` \* `do_narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, `char` `__dfault`, `char` \* `__dest`) const
- virtual `char` `do_narrow` (`char_type`, `char` `__dfault`) const
- virtual const `char_type` \* `do_scan_is` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- virtual const `char_type` \* `do_scan_not` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- virtual `char_type` `do_tolower` (`char_type` `__c`) const
- virtual const `char_type` \* `do_tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- virtual const `char_type` \* `do_toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- virtual `char_type` `do_toupper` (`char_type` `__c`) const

- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const
- virtual [char\\_type do\\_widen](#) (char \_\_c) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static [\\_GLIBCXX\\_CONST](#) const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class [locale::\\_Impl](#)

## 5.418.1 Detailed Description

**template<typename \_CharT> class std::ctype< \_CharT >**

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations. This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in [\\_\\_ctype\\_abstract\\_base](#), to allow for implementation flexibility. See [ctype<wchar\\_t>](#) for an example. The functions are documented in [\\_\\_ctype\\_abstract\\_base](#).

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 604 of file locale\_facets.h.

## 5.418.2 Member Typedef Documentation

**5.418.2.1** **template<typename \_CharT> typedef \_CharT std::ctype< \_CharT >::char\_type**

Typedef for the template parameter.

Reimplemented from [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

Definition at line 608 of file locale\_facets.h.

### 5.418.3 Member Function Documentation

**5.418.3.1** `template<typename _CharT> virtual const char_type* std::ctype<_CharT >::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [protected, virtual]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

**5.418.3.2** `template<typename _CharT> virtual bool std::ctype<_CharT >::do_is ( mask __m, char_type __c ) const [protected, virtual]`

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

*c* The char\_type to find the mask of.

*m* The mask to compare against.

**Returns**

$(M \& m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.3** `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type, char __dfault ) const [protected, virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The `char_type` to convert.

*dfault* Char to return if conversion fails.

**Returns**

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.4** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __dest ) const [protected, virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.  
*dfault* Char to use if conversion fails.  
*to* Pointer to the destination array.

### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.5** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

### Parameters

*m* The mask to compare against.  
*lo* Pointer to start of range.  
*hi* Pointer to end of range.

### Returns

Pointer to a matching `char_type` if found, else *hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

**Parameters**

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns**

Pointer to a non-matching `char_type` if found, else *hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

```
5.418.3.7 template<typename _CharT> virtual const char_type* std::ctype<
    _CharT>::do_tolower ( char_type * __lo, const char_type * __hi )
    const [protected, virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

```
5.418.3.8 template<typename _CharT> virtual char_type std::ctype<_CharT
    >::do_tolower ( char_type ) const [protected, virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

- c* The `char_type` to convert.

**Returns**

The lowercase `char_type` if convertible, else `c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.9** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.10** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type ) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The `char_type` to convert.

**Returns**

The uppercase `char_type` if convertible, else `c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.418.3.11** `template<typename _CharT> virtual char_type std::ctype<_CharT >::do_widen( char ) const [protected, virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.418.3.12** `template<typename _CharT> virtual const char* std::ctype<_CharT >::do_widen( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*lo* Pointer to start range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.418.3.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT >::is ( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

*c* The char\_type to compare the mask of.

*m* The mask to compare against.

#### Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale\_facets.h.

Referenced by `std::basic_istream<_CharT, _Traits >::sentry::sentry()`.

**5.418.3.14** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 178 of file locale\_facets.h.

**5.418.3.15** `template<typename _CharT> char std::_ctype_abstract_base<_CharT >::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 323 of file locale\_facets.h.

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**5.418.3.16** `template<typename _CharT> const char_type* std::_ctype_abstract_base<_CharT >::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline, inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 345 of file locale\_facets.h.

```
5.418.3.17 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT >::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char\\_type>::do\\_scan\\_is\(\)](#).

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to matching char\_type if found, else *hi*.

Definition at line 194 of file locale\_facets.h.

```
5.418.3.18 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT >::scan_not ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char\\_type>::do\\_scan\\_not\(\)](#).

#### Parameters

- m* The mask to compare against.
- lo* Pointer to first char in range.
- hi* Pointer to end of range.

#### Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale\_facets.h.

---

**5.418.3.19** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::tolower ( char_type *  
__lo, const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 268 of file `locale_facets.h`.

**5.418.3.20** `template<typename _CharT> char_type std::__ctype_abstract_  
base<_CharT>::tolower ( char_type __c ) const [inline,  
inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper(c)`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

**5.418.3.21** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::toupper ( char_type *  
__lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

*hi*.

Definition at line 239 of file `locale_facets.h`.

**5.418.3.22** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

### Parameters

*c* The `char_type` to convert.

### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

**5.418.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type.

Definition at line 285 of file locale\_facets.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::money\_put< \_CharT, \_OutIter >::do\_put(), and std::operator<<().

**5.418.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base< _CharT >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 304 of file locale\_facets.h.

**5.418.4 Member Data Documentation**

**5.418.4.1** `template<typename _CharT> locale::id std::ctype< _CharT >::id [static]`

The facet id for ctype<char\_type>

Definition at line 612 of file locale\_facets.h.

The documentation for this class was generated from the following file:

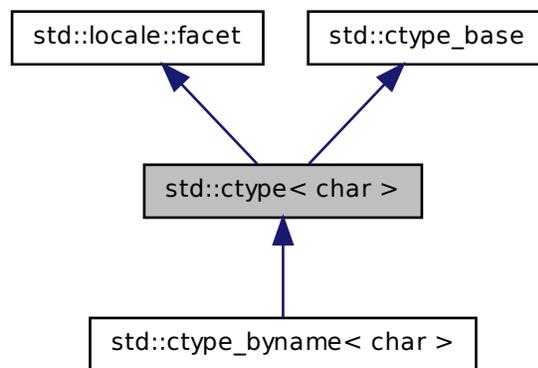
- [locale\\_facets.h](#)

## 5.419 `std::ctype< char >` Class Template Reference

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

Inheritance diagram for `std::ctype< char >`:



### Public Types

- typedef `const int * __to_type`
- typedef `char char_type`
- typedef `unsigned short mask`

### Public Member Functions

- `ctype` (`const mask * __table=0`, `bool __del=false`, `size_t __refs=0`)
- `ctype` (`__c_locale __cloc`, `const mask * __table=0`, `bool __del=false`, `size_t __refs=0`)
- `const char * is` (`const char * __lo`, `const char * __hi`, `mask * __vec`) `const`
- `bool is` (`mask __m`, `char __c`) `const`
- `char narrow` (`char_type __c`, `char __dfault`) `const`

- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* [scan\\_is](#) (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* [scan\\_not](#) (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* [table](#) () const throw ()
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

### Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) [id](#)
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char, char \* \_\_dest) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#)) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#)) const

- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi)` const
- virtual `char_type do_widen (char __c)` const
- virtual const `char * do_widen (const char *__lo, const char *__hi, char_type *__dest)` const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__c_locale _M_c_locale_ctype`
- `bool _M_del`
- `char _M_narrow [1+static_cast< unsigned char >(-1)]`
- `char _M_narrow_ok`
- `const mask * _M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- `char _M_widen [1+static_cast< unsigned char >(-1)]`
- `char _M_widen_ok`

### Friends

- class `locale::_Impl`

### 5.419.1 Detailed Description

`template<> class std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 673 of file `locale_facets.h`.

## 5.419.2 Member Typedef Documentation

### 5.419.2.1 typedef char std::ctype< char >::char\_type

Typedef for the template parameter char.

Definition at line 678 of file locale\_facets.h.

## 5.419.3 Constructor & Destructor Documentation

### 5.419.3.1 std::ctype< char >::ctype ( const mask \* \_\_table = 0, bool \_\_del = false, size\_t \_\_refs = 0 ) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*table* If non-zero, table is used as the per-char mask. Else [classic\\_table\(\)](#) is used.

*del* If true, passes ownership of table to this facet.

*refs* Passed to the base facet class.

### 5.419.3.2 std::ctype< char >::ctype ( \_\_c\_locale \_\_cloc, const mask \* \_\_table = 0, bool \_\_del = false, size\_t \_\_refs = 0 ) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

#### Parameters

*cloc* Handle to C locale data.

*table* If non-zero, table is used as the per-char mask.

*del* If true, passes ownership of table to this facet.

*refs* Passed to the base facet class.

### 5.419.3.3 virtual std::ctype< char >::~~ctype ( ) [protected, virtual]

Destructor.

This function deletes [table\(\)](#) if *del* was true in the constructor.

## 5.419.4 Member Function Documentation

**5.419.4.1** `static const mask* std::ctype< char >::classic_table ( ) throw ()`  
`[static]`

Returns a pointer to the C locale mask table.

**5.419.4.2** `virtual char std::ctype< char >::do_narrow ( char_type __c, char`  
`) const [inline, protected, virtual]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

### Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

**5.419.4.3** `virtual const char_type* std::ctype< char >::do_narrow ( const`  
`char_type * __lo, const char_type * __hi, char, char * __dest )`  
`const [inline, protected, virtual]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 1149 of file locale\_facets.h.

**5.419.4.4** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do\\_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do\\_tolower\(\)](#) must always return the same result for the same input.

**Parameters**

- lo* Pointer to first char in range.
- hi* Pointer to end of range.

**Returns**

*hi*.

**5.419.4.5** `virtual char_type std::ctype< char >::do_tolower ( char_type ) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do\\_tolower\(\)](#) must always return the same result for the same input.

**Parameters**

- c* The char to convert.

**Returns**

The lowercase char if convertible, else *c*.

**5.419.4.6** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

**5.419.4.7** `virtual char_type std::ctype< char >::do_toupper ( char_type ) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The char to convert.

**Returns**

The uppercase char if convertible, else *c*.

**5.419.4.8** virtual char\_type std::ctype< char >::do\_widen ( char \_\_c ) const  
[inline, protected, virtual]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted character.

Definition at line 1074 of file [locale\\_facets.h](#).

**5.419.4.9** virtual const char\* std::ctype< char >::do\_widen ( const char \*  
\_\_lo, const char \* \_\_hi, char\_type \* \_\_dest ) const [inline,  
protected, virtual]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be copied unchanged.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1097 of file [locale\\_facets.h](#).

**5.419.4.10** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 46 of file ctype\_inline.h.

**5.419.4.11** `bool std::ctype< char >::is ( mask __m, char __c ) const [inline]`

Test char classification.

This function compares the mask table[c] to *m*.

#### Parameters

- c* The char to compare the mask of.
- m* The mask to compare against.

#### Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype\_inline.h.

**5.419.4.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault ) const [inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted character.

Definition at line 922 of file `locale_facets.h`.

**5.419.4.13** `const char_type* std::ctype< char >::narrow ( const char_type *  
__lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 955 of file `locale_facets.h`.

**5.419.4.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const [inline]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype\_inline.h.

**5.419.4.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const [inline]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype\_inline.h.

**5.419.4.16** `const mask* std::ctype< char >::table ( ) const throw () [inline]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic\\_table\(\)](#) if none was provided.

Definition at line 973 of file locale\_facets.h.

**5.419.4.17** `char_type std::ctype< char >::tolower ( char_type __c ) const`  
`[inline]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(c)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

*c* The char to convert.

#### Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale\_facets.h.

**5.419.4.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo,`  
`const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(lo, hi)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 844 of file locale\_facets.h.

**5.419.4.19** `const char_type* std::ctype< char >::toupper ( char_type * __lo,`  
`const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(lo, hi)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Definition at line 811 of file locale\_facets.h.

**5.419.4.20 char\_type std::ctype< char >::toupper ( char\_type \_\_c ) const [inline]**

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

*c* The char to convert.

**Returns**

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale\_facets.h.

**5.419.4.21 char\_type std::ctype< char >::widen ( char \_\_c ) const [inline]**

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted character.

Definition at line 864 of file locale\_facets.h.

**5.419.4.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 891 of file locale\_facets.h.

**5.419.5 Member Data Documentation**

**5.419.5.1** `locale::id std::ctype< char >::id [static]`

The facet id for `ctype<char>`

Definition at line 695 of file locale\_facets.h.

**5.419.5.2** `const size_t std::ctype< char >::table_size [static]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file locale\_facets.h.

The documentation for this class was generated from the following files:

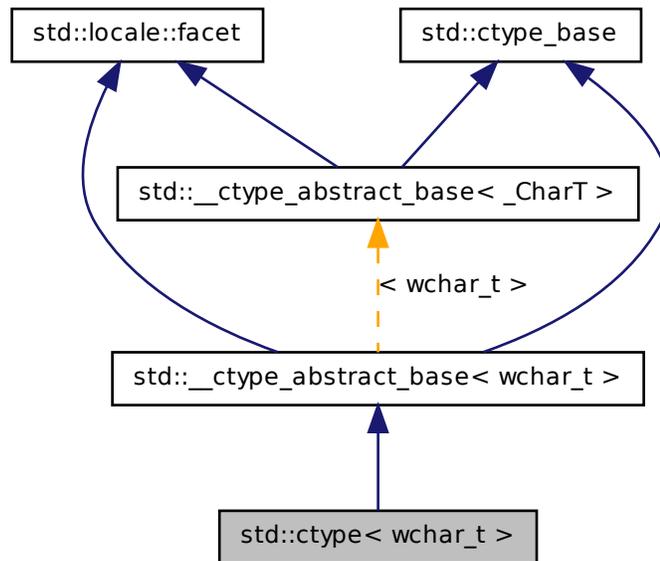
- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

## 5.420 `std::ctype< wchar_t >` Class Template Reference

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

Inheritance diagram for `std::ctype< wchar_t >`:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- **ctype** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- char **narrow** (**char\_type** \_\_c, char \_\_dfault) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **scan\_not** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **tolower** (**char\_type** \_\_c) const
- const **char\_type** \* **tolower** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **toupper** (**char\_type** \_\_c) const
- const **char\_type** \* **toupper** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, **char\_type** \* \_\_to) const
- **char\_type** **widen** (char \_\_c) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static **locale::id** **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual `~ctype ()`
- `__wmask_type _M_convert_to_wmask (const mask __m) const` throw ()
- `void _M_initialize_ctype ()` throw ()
- virtual const `char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const`
- virtual `bool do_is (mask __m, char_type __c) const` =0
- virtual const `char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const` =0
- virtual `bool do_is (mask __m, char_type __c) const`
- virtual `char do_narrow (char_type, char __dfault) const`
- virtual `char do_narrow (char_type, char __dfault) const` =0
- virtual const `char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__dest) const` =0
- virtual const `char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__dest) const`
- virtual const `char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const` =0
- virtual const `char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const`
- virtual const `char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const`
- virtual const `char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const` =0
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_tolower (char_type) const` =0
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const` =0
- virtual `char_type do_tolower (char_type) const`
- virtual `char_type do_toupper (char_type) const`
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_toupper (char_type) const` =0
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const` =0
- virtual const `char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const` =0
- virtual `char_type do_widen (char) const`
- virtual const `char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `mask _M_bit` [16]
- `__c_locale _M_c_locale_ctype`
- `char _M_narrow` [128]
- `bool _M_narrow_ok`
- `wint_t _M_widen` [1+static\_cast< unsigned char >(-1)]
- `__wmask_type _M_wmask` [16]

## Friends

- class `locale::_Impl`

### 5.420.1 Detailed Description

`template<> class std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well. `ctype<wchar_t>` inherits its public methods from `__ctype_abstract_base<wchar_t>`.

Definition at line 1174 of file `locale_facets.h`.

### 5.420.2 Member Typedef Documentation

#### 5.420.2.1 `typedef wchar_t std::ctype< wchar_t >::char_type`

Typedef for the template parameter `wchar_t`.

Reimplemented from `std::__ctype_abstract_base< wchar_t >`.

Definition at line 1179 of file `locale_facets.h`.

### 5.420.3 Constructor & Destructor Documentation

#### 5.420.3.1 `std::ctype< wchar_t >::ctype ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

*refs* Passed to the base facet class.

#### 5.420.3.2 `std::ctype< wchar_t >::ctype ( __c_locale __cloc, size_t __refs = 0 ) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

##### Parameters

*cloc* Handle to C locale data.

*refs* Passed to the base facet class.

#### 5.420.3.3 `virtual std::ctype< wchar_t >::~ctype ( ) [protected, virtual]`

Destructor.

### 5.420.4 Member Function Documentation

#### 5.420.4.1 `virtual bool std::ctype< wchar_t >::do_is ( mask __m, char_type __c ) const [protected, virtual]`

Test `wchar_t` classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

*c* The `wchar_t` to find the mask of.

*m* The mask to compare against.

**Returns**

(M & m) != 0.

**5.420.4.2** virtual bool std::\_\_ctype\_abstract\_base< wchar\_t >::do\_is ( mask \_\_m, char\_type \_\_c ) const [protected, pure virtual, inherited]

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

*c* The char\_type to find the mask of.

*m* The mask to compare against.

**Returns**

(M & m) != 0.

**5.420.4.3** virtual const char\_type\* std::ctype< wchar\_t >::do\_is ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec ) const [protected, virtual]

Return a mask array.

This function finds the mask for each wchar\_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*vec* Pointer to an array of mask storage.

**Returns**

*hi*.

**5.420.4.4** `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_is( const char_type * __lo, const char_type * __hi, mask * __vec ) const` [protected, pure virtual, inherited]

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*vec* Pointer to an array of mask storage.

#### Returns

*hi*.

**5.420.4.5** `virtual char std::__ctype_abstract_base< wchar_t >::do_narrow( char_type, char __dfault ) const` [protected, pure virtual, inherited]

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

*c* The char\_type to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted char.

**5.420.4.6** virtual const char\_type\* std::\_\_ctype\_abstract\_base< wchar\_t >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_dest ) const [protected, pure virtual, inherited]

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

**5.420.4.7** virtual char std::ctype< wchar\_t >::do\_narrow ( char\_type, char \_\_dfault ) const [protected, virtual]

Narrow wchar\_t to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an undervived [ctype<wchar\\_t>](#) facet, *c* will be cast to char and returned.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- c* The wchar\_t to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted char.

**5.420.4.8** `virtual const char_type* std::ctype< wchar_t >::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __dest ) const [protected, virtual]`

Narrow `wchar_t` array to `char` array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, `dfault` is used instead. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to `char`.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

**5.420.4.9** `virtual const char_type* std::ctype< wchar_t >::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Find `wchar_t` matching mask.

This function searches for and returns the first `wchar_t` `c` in `[lo,hi)` for which `is(m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a matching `wchar_t` if found, else *hi*.

**5.420.4.10** `virtual const char_type* std::_ctype_abstract_base< wchar_t >::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const` [`protected`, `pure virtual`, `inherited`]

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a matching `char_type` if found, else *hi*.

**5.420.4.11** `virtual const char_type* std::ctype< wchar_t >::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const` [`protected`, `virtual`]

Find `wchar_t` not matching mask.

This function searches for and returns a pointer to the first `wchar_t` `c` of `[lo,hi)` for which `is(m,c)` is false.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a non-matching `wchar_t` if found, else *hi*.

**5.420.4.12** `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, pure virtual, inherited]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

**5.420.4.13** `virtual char_type std::ctype< wchar_t >::do_tolower ( char_type ) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

- c* The `wchar_t` to convert.

#### Returns

The lowercase `wchar_t` if convertible, else *c*.

**5.420.4.14** `virtual char_type std::__ctype_abstract_base< wchar_t >::do_tolower ( char_type ) const [protected, pure virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

*c* The char\_type to convert.

#### Returns

The lowercase char\_type if convertible, else *c*.

**5.420.4.15** virtual const char\_type\* std::\_ctype\_abstract\_base< wchar\_t >::do\_tolower ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected, pure virtual, inherited]

Convert array to lowercase.

This virtual function converts each char\_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

**5.420.4.16** virtual const char\_type\* std::ctype< wchar\_t >::do\_tolower ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected, virtual]

Convert array to lowercase.

This virtual function converts each wchar\_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

**5.420.4.17** `virtual char_type std::__ctype_abstract_base< wchar_t >::do_toupper ( char_type ) const [protected, pure virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

*c* The `char_type` to convert.

**Returns**

The uppercase `char_type` if convertible, else *c*.

**5.420.4.18** `virtual char_type std::ctype< wchar_t >::do_toupper ( char_type ) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The `wchar_t` to convert.

**Returns**

The uppercase `wchar_t` if convertible, else *c*.

**5.420.4.19** virtual const char\_type\* std::\_\_ctype\_abstract\_base< wchar\_t >::do\_toupper ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected, pure virtual, inherited]

Convert array to uppercase.

This virtual function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

**5.420.4.20** virtual const char\_type\* std::ctype< wchar\_t >::do\_toupper ( char\_type \* \_\_lo, const char\_type \* \_\_hi ) const [protected, virtual]

Convert array to uppercase.

This virtual function converts each wchar\_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

**5.420.4.21** virtual char\_type std::ctype< wchar\_t >::do\_widen ( char ) const [protected, virtual]

Widen char to wchar\_t.

This virtual function converts the char to wchar\_t using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be cast to wchar\_t.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted wchar\_t.

Implements `std::__ctype_abstract_base< wchar_t >`.

**5.420.4.22** `virtual const char* std::ctype< wchar_t >::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual]`

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to wchar\_t.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*lo* Pointer to start range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

#### Returns

*hi*.

**5.420.4.23** `virtual const char* std::__ctype_abstract_base< wchar_t >::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [protected, pure virtual, inherited]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

**5.420.4.24** `const char_type* std::_ctype_abstract_base< wchar_t >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 178 of file locale\_facets.h.

**5.420.4.25** `bool std::_ctype_abstract_base< wchar_t >::is ( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask M for c and compares it to mask m. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

- c* The char\_type to compare the mask of.
- m* The mask to compare against.

**Returns**

(M & m) != 0.

Definition at line 161 of file locale\_facets.h.

**5.420.4.26** `const char_type* std::__ctype_abstract_base< wchar_t >::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline, inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char\_type>::do\_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 345 of file locale\_facets.h.

**5.420.4.27** `char std::__ctype_abstract_base< wchar_t >::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. It does so by returning ctype<char\_type>::do\_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

- c* The char\_type to convert.  
*dfault* Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 323 of file locale\_facets.h.

**5.420.4.28** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

- m* The mask to compare against.  
*lo* Pointer to start of range.  
*hi* Pointer to end of range.

**Returns**

Pointer to matching char\_type if found, else *hi*.

Definition at line 194 of file locale\_facets.h.

**5.420.4.29** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

- m* The mask to compare against.  
*lo* Pointer to first char in range.  
*hi* Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale\_facets.h.

**5.420.4.30** `const char_type* std::__ctype_abstract_base< wchar_t >::tolower ( char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each char\_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Definition at line 268 of file locale\_facets.h.

**5.420.4.31** `char_type std::__ctype_abstract_base< wchar_t >::tolower ( char_type __c ) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

*c* The char\_type to convert.

**Returns**

The lowercase char\_type if convertible, else *c*.

Definition at line 253 of file locale\_facets.h.

---

**5.420.4.32** `const char_type* std::_ctype_abstract_base< wchar_t >::toupper ( char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 239 of file `locale_facets.h`.

**5.420.4.33** `char_type std::_ctype_abstract_base< wchar_t >::toupper ( char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

**5.420.4.34** `char_type std::_ctype_abstract_base< wchar_t >::widen ( char __c ) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type.

Definition at line 285 of file locale\_facets.h.

**5.420.4.35** `const char* std::__ctype_abstract_base< wchar_t >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 304 of file locale\_facets.h.

## 5.420.5 Member Data Documentation

**5.420.5.1** `locale::id std::ctype< wchar_t >::id [static]`

The facet id for [ctype<wchar\\_t>](#)

Definition at line 1197 of file locale\_facets.h.

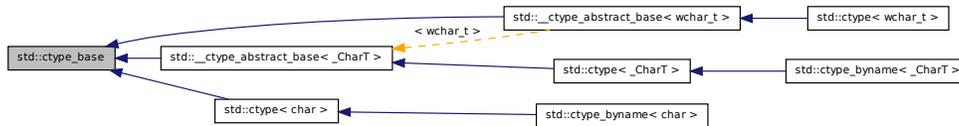
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.421 std::ctype\_base Struct Reference

Base class for ctype.

Inheritance diagram for std::ctype\_base:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

#### 5.421.1 Detailed Description

Base class for ctype.

Definition at line 40 of file ctype\_base.h.

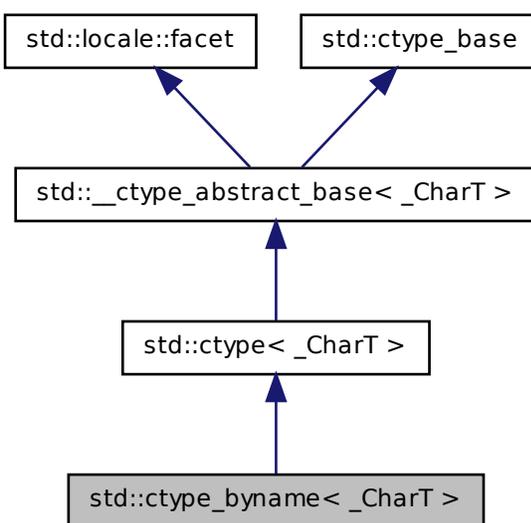
The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 5.422 `std::ctype_byname< _CharT >` Class Template Reference

class `ctype_byname` [22.2.1.2].

Inheritance diagram for `std::ctype_byname< _CharT >`:



### Public Types

- typedef `const int * __to_type`
- typedef `_CharT char_type`
- typedef `ctype< _CharT >::mask mask`

### Public Member Functions

- `ctype_byname` (`const char * __s`, `size_t __refs=0`)
- `bool is` (`mask __m`, `char_type __c`) `const`
- `const char_type * is` (`const char_type * __lo`, `const char_type * __hi`, `mask * __vec`) `const`

- `const char_type * narrow` (`const char_type *__lo`, `const char_type *__hi`, `char __dfault`, `char *__to`) `const`
- `char narrow` (`char_type __c`, `char __dfault`) `const`
- `const char_type * scan_is` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- `const char_type * scan_not` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type *__lo`, `const char_type *__hi`) `const`
- `const char_type * toupper` (`char_type *__lo`, `const char_type *__hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char * widen` (`const char *__lo`, `const char *__hi`, `char_type *__to`) `const`
- `char_type widen` (`char __c`) `const`

### Static Public Attributes

- static `const mask` **alnum**
- static `const mask` **alpha**
- static `const mask` **cntrl**
- static `const mask` **digit**
- static `const mask` **graph**
- static `locale::id` **id**
- static `const mask` **lower**
- static `const mask` **print**
- static `const mask` **punct**
- static `const mask` **space**
- static `const mask` **upper**
- static `const mask` **xdigit**

### Protected Member Functions

- virtual `bool do_is` (`mask __m`, `char_type __c`) `const`
- virtual `const char_type * do_is` (`const char_type *__lo`, `const char_type *__hi`, `mask *__vec`) `const`
- virtual `const char_type * do_narrow` (`const char_type *__lo`, `const char_type *__hi`, `char __dfault`, `char *__dest`) `const`
- virtual `char do_narrow` (`char_type`, `char __dfault`) `const`
- virtual `const char_type * do_scan_is` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- virtual `const char_type * do_scan_not` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- virtual `char_type do_tolower` (`char_type __c`) `const`

- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale](#) [\\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale](#) [\\_S\\_get\\_c\\_locale](#) ()
- static [\\_GLIBCXX\\_CONST](#) const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale](#) [\\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

### Friends

- class [locale::Impl](#)

### 5.422.1 Detailed Description

**template<typename \_CharT> class std::ctype\_byname< \_CharT >**

class [ctype\\_byname](#) [22.2.1.2].

Definition at line 1466 of file [locale\\_facets.h](#).

### 5.422.2 Member Typedef Documentation

**5.422.2.1 template<typename \_CharT> typedef \_CharT std::ctype< \_CharT >::char\_type [inherited]**

Typedef for the template parameter.

Reimplemented from [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Definition at line 608 of file [locale\\_facets.h](#).

### 5.422.3 Member Function Documentation

**5.422.3.1** `template<typename _CharT> virtual const char_type* std::ctype<_CharT >::do_is( const char_type * __lo, const char_type * __hi, mask * __vec ) const [protected, virtual, inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

**5.422.3.2** `template<typename _CharT> virtual bool std::ctype<_CharT >::do_is( mask __m, char_type __c ) const [protected, virtual, inherited]`

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

- c* The char\_type to find the mask of.
- m* The mask to compare against.

#### Returns

(M & m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

**5.422.3.3** `template<typename _CharT> virtual char std::ctype< _CharT >::do_narrow ( char_type, char __dfault ) const [protected, virtual, inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted `char`.

Implements `std::__ctype_abstract_base< _CharT >`.

**5.422.3.4** `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __dest ) const [protected, virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

```
5.422.3.5 template<typename _CharT> virtual const char_type* std::ctype<
    _CharT >::do_scan_is ( mask __m, const char_type * __lo,
    const char_type * __hi ) const [protected, virtual,
    inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

**Parameters**

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

Pointer to a matching char\_type if found, else *hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

```
5.422.3.6 template<typename _CharT> virtual const char_type* std::ctype<
    _CharT >::do_scan_not ( mask __m, const char_type * __lo,
    const char_type * __hi ) const [protected, virtual,
    inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

**Parameters**

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.422.3.7** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

*hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.422.3.8** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type ) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

### Parameters

*c* The `char_type` to convert.

**Returns**

The lowercase `char_type` if convertible, else `c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.422.3.9** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.422.3.10** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type ) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

*c* The `char_type` to convert.

**Returns**

The uppercase `char_type` if convertible, else `c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.422.3.11** `template<typename _CharT> virtual char_type std::ctype<_CharT >::do_widen( char ) const [protected, virtual, inherited]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

**5.422.3.12** `template<typename _CharT> virtual const char* std::ctype<_CharT >::do_widen( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual, inherited]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*lo* Pointer to start range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

**5.422.3.13** `template<typename _CharT> bool std::_ctype_abstract_base<_CharT >::is( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

- c* The char\_type to compare the mask of.
- m* The mask to compare against.

#### Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale\_facets.h.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.422.3.14** `template<typename _CharT> const char_type* std::_ctype_abstract_base<_CharT >::is( const char_type * __lo, const char_type * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 178 of file locale\_facets.h.

**5.422.3.15** `template<typename _CharT> char std::_ctype_abstract_base<_CharT>::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 323 of file locale\_facets.h.

Referenced by std::time\_put<\_CharT, \_OutIter>::put().

**5.422.3.16** `template<typename _CharT> const char_type* std::_ctype_abstract_base<_CharT>::narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline, inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 345 of file locale\_facets.h.

```
5.422.3.17 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char\\_type>::do\\_scan\\_is\(\)](#).

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to matching char\_type if found, else *hi*.

Definition at line 194 of file locale\_facets.h.

```
5.422.3.18 template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::scan_not ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char\\_type>::do\\_scan\\_not\(\)](#).

#### Parameters

*m* The mask to compare against.

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale\_facets.h.

**5.422.3.19** `template<typename _CharT> const char_type*  
std::_ctype_abstract_base<_CharT >::tolower ( char_type *  
__lo, const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 268 of file `locale_facets.h`.

**5.422.3.20** `template<typename _CharT> char_type std::_ctype_abstract_-  
base<_CharT >::tolower ( char_type __c ) const [inline,  
inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper(c)`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

**5.422.3.21** `template<typename _CharT> const char_type*  
std::_ctype_abstract_base<_CharT >::toupper ( char_type *  
__lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 239 of file `locale_facets.h`.

**5.422.3.22** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

**5.422.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline, inherited]`

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type.

Definition at line 285 of file locale\_facets.h.

Referenced by std::money\_get<\_CharT, \_InIter >::do\_get(), std::time\_put<\_CharT, \_OutIter >::do\_put(), std::money\_put<\_CharT, \_OutIter >::do\_put(), and std::operator<<().

**5.422.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 304 of file locale\_facets.h.

**5.422.4 Member Data Documentation**

**5.422.4.1** `template<typename _CharT> locale::id std::ctype<_CharT >::id [static, inherited]`

The facet id for ctype<char\_type>

Definition at line 612 of file locale\_facets.h.

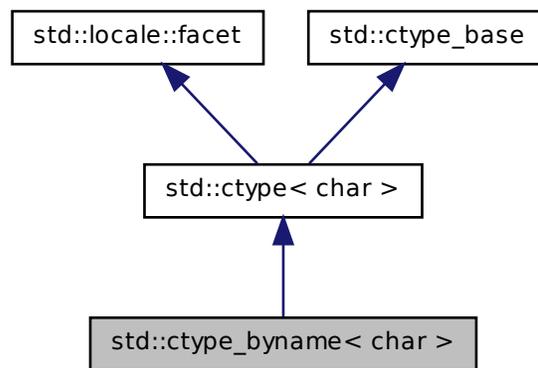
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.423 std::ctype\_byname< char > Class Template Reference

22.2.1.4 Class `ctype_byname` specializations.

Inheritance diagram for `std::ctype_byname< char >`:



### Public Types

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef unsigned short `mask`

### Public Member Functions

- `ctype_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- bool `is` (mask \_\_m, char \_\_c) const
- const char \* `is` (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- const `char_type` \* `narrow` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, char \_\_default, char \* \_\_to) const
- char `narrow` (`char_type` \_\_c, char \_\_default) const
- const char \* `scan_is` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* `scan_not` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const

- const mask \* [table](#) () const throw ()
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- const char \* [widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_to) const
- [char\\_type](#) [widen](#) (char \_\_c) const

## Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi, char, char \*\_\_dest) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#)) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#)) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_ctype`
- `bool _M_del`
- `char _M_narrow [1+static_cast< unsigned char >(-1)]`
- `char _M_narrow_ok`
- `const mask * _M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- `char _M_widen [1+static_cast< unsigned char >(-1)]`
- `char _M_widen_ok`

## Friends

- class `locale::_Impl`

### 5.423.1 Detailed Description

`template<> class std::ctype_byname< char >`

22.2.1.4 Class `ctype_byname` specializations.

Definition at line 1481 of file `locale_facets.h`.

### 5.423.2 Member Typedef Documentation

#### 5.423.2.1 `typedef char std::ctype< char >::char_type` [inherited]

Typedef for the template parameter `char`.

Definition at line 678 of file `locale_facets.h`.

### 5.423.3 Member Function Documentation

**5.423.3.1** `static const mask* std::ctype< char >::classic_table ( ) throw ()`  
`[static, inherited]`

Returns a pointer to the C locale mask table.

**5.423.3.2** `virtual char std::ctype< char >::do_narrow ( char_type __c, char`  
`) const [inline, protected, virtual, inherited]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

**5.423.3.3** `virtual const char_type* std::ctype< char >::do_narrow ( const`  
`char_type * __lo, const char_type * __hi, char, char * __dest )`  
`const [inline, protected, virtual, inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 1149 of file locale\_facets.h.

**5.423.3.4** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

- lo* Pointer to first char in range.
- hi* Pointer to end of range.

**Returns**

*hi*.

**5.423.3.5** `virtual char_type std::ctype< char >::do_tolower ( char_type ) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

- c* The char to convert.

**Returns**

The lowercase char if convertible, else *c*.

**5.423.3.6** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

**5.423.3.7** `virtual char_type std::ctype< char >::do_toupper ( char_type ) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The char to convert.

**Returns**

The uppercase char if convertible, else *c*.

**5.423.3.8** virtual char\_type std::ctype< char >::do\_widen ( char \_\_c ) const  
[inline, protected, virtual, inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted character.

Definition at line 1074 of file [locale\\_facets.h](#).

**5.423.3.9** virtual const char\* std::ctype< char >::do\_widen ( const char \*  
\_\_lo, const char \* \_\_hi, char\_type \* \_\_dest ) const [inline,  
protected, virtual, inherited]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be copied unchanged.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1097 of file [locale\\_facets.h](#).

**5.423.3.10** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 46 of file ctype\_inline.h.

**5.423.3.11** `bool std::ctype< char >::is ( mask __m, char __c ) const [inline, inherited]`

Test char classification.

This function compares the mask table[c] to *m*.

#### Parameters

- c* The char to compare the mask of.
- m* The mask to compare against.

#### Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype\_inline.h.

**5.423.3.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted character.

Definition at line 922 of file `locale_facets.h`.

**5.423.3.13** `const char_type* std::ctype< char >::narrow ( const char_type *  
__lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline, inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 955 of file `locale_facets.h`.

**5.423.3.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const [inline, inherited]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype\_inline.h.

**5.423.3.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const [inline, inherited]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype\_inline.h.

**5.423.3.16** `const mask* std::ctype< char >::table ( ) const throw () [inline, inherited]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic\\_table\(\)](#) if none was provided.

Definition at line 973 of file locale\_facets.h.

**5.423.3.17** `char_type std::ctype< char >::tolower ( char_type __c ) const`  
`[inline, inherited]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(c)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

*c* The char to convert.

#### Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale\_facets.h.

**5.423.3.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo,`  
`const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(lo, hi)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 844 of file locale\_facets.h.

**5.423.3.19** `const char_type* std::ctype< char >::toupper ( char_type * __lo,`  
`const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(lo, hi)`. `do_toupper()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 811 of file locale\_facets.h.

**5.423.3.20** `char_type std::ctype< char >::toupper ( char_type __c ) const`  
`[inline, inherited]`

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

#### Parameters

*c* The char to convert.

#### Returns

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale\_facets.h.

**5.423.3.21** `char_type std::ctype< char >::widen ( char __c ) const`  
`[inline, inherited]`

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted character.

Definition at line 864 of file locale\_facets.h.

**5.423.3.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 891 of file locale\_facets.h.

**5.423.4 Member Data Documentation**

**5.423.4.1** `locale::id std::ctype< char >::id [static, inherited]`

The facet id for `ctype<char>`

Definition at line 695 of file locale\_facets.h.

**5.423.4.2** `const size_t std::ctype< char >::table_size [static, inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.424 `std::decay< _Tp >` Class Template Reference

decay

### Public Types

- `typedef __decay_selector< __remove_type >::__type type`

### 5.424.1 Detailed Description

`template<typename _Tp> class std::decay< _Tp >`

decay

Definition at line 400 of file type\_traits.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 5.425 `std::decimal::decimal128` Class Reference

3.2.4 Class [decimal128](#).

### Public Types

- `typedef float __decfloat128 __attribute__((mode(TD)))`

### Public Member Functions

- `decimal128` ([decimal32](#) d32)
- `decimal128` (float \_\_r)
- `decimal128` (unsigned int \_\_z)
- `decimal128` (long \_\_z)

- **decimal128** (double \_\_r)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (long long \_\_z)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (long double \_\_r)
- **decimal128** (unsigned long long \_\_z)
- [decimal128](#) (\_\_decfloat128 \_\_z)
- **decimal128** (int \_\_z)
- \_\_decfloat128 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat128 \_\_x)
- [decimal128](#) & **operator\*=([decimal32](#) \_\_rhs)**
- [decimal128](#) & **operator\*=([decimal64](#) \_\_rhs)**
- [decimal128](#) & **operator\*=([decimal128](#) \_\_rhs)**
- [decimal128](#) & **operator\*=(int \_\_rhs)**
- [decimal128](#) & **operator\*=(unsigned int \_\_rhs)**
- [decimal128](#) & **operator\*=(long \_\_rhs)**
- [decimal128](#) & **operator\*=(unsigned long \_\_rhs)**
- [decimal128](#) & **operator\*=(unsigned long long \_\_rhs)**
- [decimal128](#) & **operator\*=(long long \_\_rhs)**
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=(int \_\_rhs)**
- [decimal128](#) & **operator+=([decimal32](#) \_\_rhs)**
- [decimal128](#) & **operator+=([decimal64](#) \_\_rhs)**
- [decimal128](#) & **operator+=([decimal128](#) \_\_rhs)**
- [decimal128](#) & **operator+=(unsigned int \_\_rhs)**
- [decimal128](#) & **operator+=(long \_\_rhs)**
- [decimal128](#) & **operator+=(unsigned long \_\_rhs)**
- [decimal128](#) & **operator+=(long long \_\_rhs)**
- [decimal128](#) & **operator+=(unsigned long long \_\_rhs)**
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=(long long \_\_rhs)**
- [decimal128](#) & **operator-=(long \_\_rhs)**
- [decimal128](#) & **operator-=(int \_\_rhs)**
- [decimal128](#) & **operator-=([decimal32](#) \_\_rhs)**
- [decimal128](#) & **operator-=(unsigned long \_\_rhs)**
- [decimal128](#) & **operator-=([decimal128](#) \_\_rhs)**
- [decimal128](#) & **operator-=(unsigned long long \_\_rhs)**
- [decimal128](#) & **operator-=(unsigned int \_\_rhs)**
- [decimal128](#) & **operator-=([decimal64](#) \_\_rhs)**
- [decimal128](#) & **operator/=([decimal64](#) \_\_rhs)**
- [decimal128](#) & **operator/=(long \_\_rhs)**

- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)

### 5.425.1 Detailed Description

3.2.4 Class [decimal128](#).

Definition at line 391 of file decimal.

### 5.425.2 Constructor & Destructor Documentation

#### 5.425.2.1 `std::decimal::decimal128::decimal128 ( __decfloat128 __z )` `[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 416 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.426 `std::decimal::decimal32` Class Reference

3.2.2 Class [decimal32](#).

### Public Types

- typedef float `__decfloat32` **\_\_attribute\_\_** ((mode(SD)))

### Public Member Functions

- `decimal32` ([decimal64](#) \_\_d64)
- `decimal32` (float \_\_r)
- `decimal32` (unsigned int \_\_z)
- `decimal32` (long \_\_z)

- **decimal32** (double \_\_r)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** (long double \_\_r)
- **decimal32** (unsigned long long \_\_z)
- [decimal32](#) (\_\_decfloat32 \_\_z)
- **decimal32** (int \_\_z)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- [decimal32](#) & **operator\***= ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator\***= ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\***= ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\***= (int \_\_rhs)
- [decimal32](#) & **operator\***= (unsigned int \_\_rhs)
- [decimal32](#) & **operator\***= (long \_\_rhs)
- [decimal32](#) & **operator\***= (unsigned long \_\_rhs)
- [decimal32](#) & **operator\***= (unsigned long long \_\_rhs)
- [decimal32](#) & **operator\***= (long long \_\_rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** (int \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (long long \_\_rhs)
- [decimal32](#) & **operator-=** (long \_\_rhs)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)

- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)

### 5.426.1 Detailed Description

3.2.2 Class [decimal32](#).

Definition at line 225 of file decimal.

### 5.426.2 Constructor & Destructor Documentation

#### 5.426.2.1 `std::decimal::decimal32::decimal32 ( __decfloat32 __z )` `[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 249 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.427 `std::decimal::decimal64` Class Reference

3.2.3 Class [decimal64](#).

### Public Types

- typedef float `__decfloat64` **\_\_attribute\_\_** ((mode(DD)))

### Public Member Functions

- [decimal64](#) ([decimal32](#) d32)
- [decimal64](#) (float \_\_r)
- [decimal64](#) (unsigned int \_\_z)
- [decimal64](#) (long \_\_z)

- **decimal64** (double \_\_r)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (long double \_\_r)
- **decimal64** (unsigned long long \_\_z)
- [decimal64](#) (\_\_decfloat64 \_\_z)
- **decimal64** (int \_\_z)
- [\\_\\_decfloat64](#) **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- [decimal64](#) & **operator\*=([decimal32](#) \_\_rhs)**
- [decimal64](#) & **operator\*=([decimal64](#) \_\_rhs)**
- [decimal64](#) & **operator\*=([decimal128](#) \_\_rhs)**
- [decimal64](#) & **operator\*=(int \_\_rhs)**
- [decimal64](#) & **operator\*=(unsigned int \_\_rhs)**
- [decimal64](#) & **operator\*=(long \_\_rhs)**
- [decimal64](#) & **operator\*=(unsigned long \_\_rhs)**
- [decimal64](#) & **operator\*=(unsigned long long \_\_rhs)**
- [decimal64](#) & **operator\*=(long long \_\_rhs)**
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=(int \_\_rhs)**
- [decimal64](#) & **operator+=([decimal32](#) \_\_rhs)**
- [decimal64](#) & **operator+=([decimal64](#) \_\_rhs)**
- [decimal64](#) & **operator+=([decimal128](#) \_\_rhs)**
- [decimal64](#) & **operator+=(unsigned int \_\_rhs)**
- [decimal64](#) & **operator+=(long \_\_rhs)**
- [decimal64](#) & **operator+=(unsigned long \_\_rhs)**
- [decimal64](#) & **operator+=(long long \_\_rhs)**
- [decimal64](#) & **operator+=(unsigned long long \_\_rhs)**
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=(long long \_\_rhs)**
- [decimal64](#) & **operator-=(long \_\_rhs)**
- [decimal64](#) & **operator-=(int \_\_rhs)**
- [decimal64](#) & **operator-=([decimal32](#) \_\_rhs)**
- [decimal64](#) & **operator-=(unsigned long \_\_rhs)**
- [decimal64](#) & **operator-=([decimal128](#) \_\_rhs)**
- [decimal64](#) & **operator-=(unsigned long long \_\_rhs)**
- [decimal64](#) & **operator-=(unsigned int \_\_rhs)**
- [decimal64](#) & **operator-=([decimal64](#) \_\_rhs)**
- [decimal64](#) & **operator/=([decimal64](#) \_\_rhs)**
- [decimal64](#) & **operator/=(long \_\_rhs)**

- [decimal64](#) & **operator/=** (long long \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator/=** (int \_\_rhs)

### 5.427.1 Detailed Description

3.2.3 Class [decimal64](#).

Definition at line 308 of file decimal.

### 5.427.2 Constructor & Destructor Documentation

#### 5.427.2.1 `std::decimal::decimal64::decimal64 ( __decfloat64 __z )` `[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 332 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.428 `std::default_delete<_Tp>` Struct Template Reference

Primary template, [default\\_delete](#).

### Public Member Functions

- `template<typename _Up >`  
`default_delete` (const `default_delete<_Up >` &)
- `void operator() (_Tp *__ptr) const`

### 5.428.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp >`

Primary template, [default\\_delete](#).

Definition at line 48 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.429 `std::default_delete< _Tp[]>` Struct Template Reference

Specialization, [default\\_delete](#).

### Public Member Functions

- `void operator() (_Tp * __ptr) const`

### 5.429.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp[]>`

Specialization, [default\\_delete](#).

Definition at line 68 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.430 `std::defer_lock_t` Struct Reference

Do not acquire ownership of the mutex.

### 5.430.1 Detailed Description

Do not acquire ownership of the mutex.

Definition at line 376 of file `mutex`.

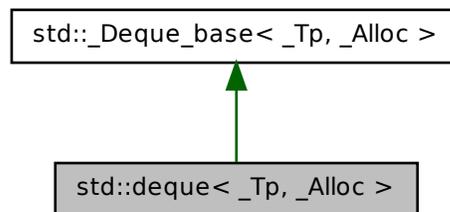
The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.431 `std::deque< _Tp, _Alloc >` Class Template Reference

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Inheritance diagram for `std::deque< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [deque](#) ()
- [deque](#) (const allocator\_type &\_\_a)
- [deque](#) (const [deque](#) &\_\_x)
- [deque](#) ([deque](#) &&\_\_x)
- [deque](#) (size\_type \_\_n, const value\_type &\_\_value=value\_type(), const allocator\_type &\_\_a=allocator\_type())
- [deque](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[deque](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- [~deque](#) ()
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) ()
- const\_reference [back](#) () const
- [iterator begin](#) ()
- [const\\_iterator begin](#) () const
- [const\\_iterator cbegin](#) () const
- [const\\_iterator cend](#) () const
- void [clear](#) ()
- [const\\_reverse\\_iterator crbegin](#) () const
- [const\\_reverse\\_iterator crend](#) () const
- template<typename... \_Args >  
[iterator emplace](#) ([iterator](#) \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args >  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- template<typename... \_Args >  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- [iterator end](#) ()
- [const\\_iterator end](#) () const
- [iterator erase](#) ([iterator](#) \_\_position)
- [iterator erase](#) ([iterator](#) \_\_first, [iterator](#) \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const

- `template<typename _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `iterator insert (iterator __position, value_type &&__x)`
- `void insert (iterator __position, size_type __n, const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `size_type max_size () const`
- `deque & operator= (const deque &__x)`
- `deque & operator= (deque &&__x)`
- `deque & operator= (initializer_list< value_type > __l)`
- `reference operator[ ] (size_type __n)`
- `const_reference operator[ ] (size_type __n) const`
- `void pop_back ()`
- `void pop_front ()`
- `void push_back (value_type &&__x)`
- `void push_back (const value_type &__x)`
- `void push_front (value_type &&__x)`
- `void push_front (const value_type &__x)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void resize (size_type __new_size, value_type __x=value_type())`
- `void shrink_to_fit ()`
- `size_type size () const`
- `void swap (deque &__x)`

## Protected Types

- `enum { _S_initial_map_size }`
- `typedef _Alloc::template rebind< _Tp * >::other _Map_alloc_type`
- `typedef pointer * _Map_pointer`

## Protected Member Functions

- `_Tp ** _M_allocate_map (size_t __n)`
- `_Tp * _M_allocate_node ()`
- `template<typename _ForwardIterator >`  
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`

- `template<typename _Integer >`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_deallocate_map (_Tp ** __p, size_t __n)`
- `void _M_deallocate_node (_Tp * __p)`
- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator<_Tp > &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type & __val)`
- `void _M_fill_initialize (const value_type & __value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type & __x)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type & __x)`
- `template<typename _ForwardIterator >`  
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `template<typename... _Args >`  
`iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`  
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`

- `template<typename _ForwardIterator >`  
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _-`  
`ForwardIterator __last, std::forward_iterator_tag)`
  
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last,`  
`std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward_iterator_tag)`
  
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&...__args)`
- `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&...__args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
  
- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`
  
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

## Static Protected Member Functions

- `static size_t _S_buffer_size ()`

## Protected Attributes

- `_Deque_impl _M_impl`

### 5.431.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::deque<_Tp, _Alloc >
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of `deque`, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each `deque` has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of- `Tp`. If  `Tp` is very large, there will be one  `Tp` element per node (i.e., an *array* of one). For non-huge  `Tp`'s, node size is inversely related to  `Tp` size: the larger the  `Tp`, the fewer  `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different  `Tp`'s, to improve allocator efficiency.

Not every pointer in the `map` array will point to a node. If the initial number of elements in the `deque` is small, the /middle/ `map` pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the `map` grows: available `map` pointers, if any, will be on the ends. As new nodes are created, only a subset of the `map`'s pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator  `i`:
  - `i.node` points to a member of the `map` array. (Yes, you read that correctly:  `i.node` does not actually point to a node.) The member of the `map` array is what actually points to the node.
  - `i.first == *(i.node)` (This points to the node (first  `Tp` element).)
  - `i.last == i.first + node_size`
  - `i.cur` is a pointer in the range  `[i.first, i.last)`. NOTE: the implication of this is that  `i.cur` is always a dereferenceable pointer, even if  `i` is a past-the-end iterator.
- `Start` and  `Finish` are always nonsingular iterators. NOTE: this means that an empty `deque` must have one node, a `deque` with  `<N` elements (where  `N` is the node buffer size) must have one node, a `deque` with  `N` through  `(2N-1)` elements must have two nodes, etc.
- For every node other than  `start.node` and  `finish.node`, every element in the node is an initialized object. If  `start.node == finish.node`, then  `[start.cur, finish.cur)` are

initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.

- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, \_Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 713 of file stl\_deque.h.

## 5.431.2 Constructor & Destructor Documentation

**5.431.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::deque ( ) [inline]`

Default constructor creates no elements.

Definition at line 765 of file stl\_deque.h.

**5.431.2.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::deque ( const allocator_type & __a )  
[inline, explicit]`

Creates a deque with no elements.

### Parameters

*a* An allocator object.

Definition at line 773 of file stl\_deque.h.

```
5.431.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque ( size_type __n, const value_type
& __value = value_type(), const allocator_type & __a =
allocator_type() ) [inline, explicit]
```

Creates a deque with copies of an exemplar element.

#### Parameters

*n* The number of elements to initially create.

*value* An element to copy.

*a* An allocator.

This constructor fills the deque with *n* copies of *value*.

Definition at line 785 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::\_M\_fill\_initialize().

```
5.431.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque ( const deque<_Tp, _Alloc> &
__x ) [inline]
```

Deque copy constructor.

#### Parameters

*x* A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocation object used by *x*.

Definition at line 797 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::begin(), and std::deque< \_Tp, \_Alloc >::end().

```
5.431.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque ( deque<_Tp, _Alloc> && __x
) [inline]
```

Deque move constructor.

#### Parameters

*x* A deque of identical element and allocator types.

The newly-created deque contains the exact contents of *x*. The contents of *x* are a valid, but unspecified deque.

Definition at line 811 of file `stl_deque.h`.

```
5.431.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque ( initializer_list< value_type
> __l, const allocator_type & __a = allocator_type() )
[inline]
```

Builds a deque from an initializer list.

#### Parameters

- l* An `initializer_list`.
- a* An allocator object.

Create a deque consisting of copies of the elements in the `initializer_list` *l*.

This will call the element type's copy constructor *N* times (where *N* is `l.size()`) and do no memory reallocation.

Definition at line 825 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_range_initialize()`.

```
5.431.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > std::deque<_Tp, _Alloc
>::deque ( _InputIterator __first, _InputIterator __last, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a deque from a range.

#### Parameters

- first* An input iterator.
- last* An input iterator.
- a* An allocator object.

Create a deque consisting of copies of the elements from `[first, last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most *2N* calls to the copy constructor, and *logN* memory reallocations.

Definition at line 850 of file `stl_deque.h`.

**5.431.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::~~deque( ) [inline]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 864 of file stl\_deque.h.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

### 5.431.3 Member Function Documentation

**5.431.3.1** `template<typename _Tp, typename _Alloc > void  
deque::_M_fill_initialize( const value_type & __value )  
[protected]`

Fills the deque with copies of value.

#### Parameters

*value* Initial value.

#### Returns

Nothing.

#### Precondition

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 277 of file deque.tcc.

References `std::_Destroy()`.

Referenced by `std::deque<_Tp, _Alloc>::deque()`.

**5.431.3.2** `template<typename _Tp, typename _Alloc > void  
std::_Deque_base<_Tp, _Alloc>::_M_initialize_map( size_t  
__num_elements ) [protected, inherited]`

Layout storage.

**Parameters**

*num\_elements* The count of T's for which to allocate space at first.

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 568 of file stl\_deque.h.

References std::max().

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_range\_initialize().

**5.431.3.3** `template<typename _Tp , typename _Alloc > void  
deque::_M_new_elements_at_back ( size_type __new_elements )  
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 771 of file deque.tcc.

References std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_back(), std::deque< \_Tp, \_Alloc >::max\_size(), and std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_elements\_at\_back().

**5.431.3.4** `template<typename _Tp , typename _Alloc > void  
deque::_M_new_elements_at_front ( size_type __new_elements )  
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 746 of file deque.tcc.

References std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_front(), std::deque< \_Tp, \_Alloc >::max\_size(), and std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_elements\_at\_front().

**5.431.3.5** `template<typename _Tp , typename _Alloc > void  
deque::_M_pop_back_aux ( ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 427 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_back().

**5.431.3.6** `template<typename _Tp, typename _Alloc > void  
deque::_M_pop_front_aux ( ) [protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 442 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_front().

**5.431.3.7** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> void deque::_M_push_back_aux ( _Args &&... __args )  
[protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 361 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_back().

**5.431.3.8** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> void deque::_M_push_front_aux ( _Args &&... __args )  
[protected]`

Helper functions for push\_\* and pop\_\*.

Definition at line 395 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_front().

**5.431.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::_M_range_check ( size_type __n )  
const [inline, protected]`

Safety check used only from [at\(\)](#).

Definition at line 1159 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::at().

**5.431.3.10** `template<typename _Tp, typename _Alloc > template<typename  
_InputIterator > void deque::_M_range_initialize ( _InputIterator  
__first, _InputIterator __last, std::input_iterator_tag )  
[protected]`

Fills the deque with whatever is in [first,last).

**Parameters**

*first* An input iterator.

*last* An input iterator.

**Returns**

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 303 of file `deque.tcc`.

References `std::_Deque_base<_Tp, _Alloc>::_M_initialize_map()`, `std::deque<_Tp, _Alloc>::clear()`, and `std::deque<_Tp, _Alloc>::push_back()`.

Referenced by `std::deque<_Tp, _Alloc>::deque()`.

**5.431.3.11** `template<typename _Tp, typename _Alloc > template<typename _ForwardIterator > void deque::_M_range_initialize ( _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag ) [protected]`

Fills the deque with whatever is in `[first,last)`.

**Parameters**

*first* An input iterator.

*last* An input iterator.

**Returns**

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 323 of file `deque.tcc`.

References `std::_Destroy()`, `std::_Deque_base<_Tp, _Alloc>::_M_initialize_map()`, `std::advance()`, and `std::distance()`.

**5.431.3.12** `template<typename _Tp, typename _Alloc > void deque::_M_reallocate_map ( size_type __nodes_to_add, bool __add_at_front ) [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 796 of file `deque.tcc`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, and `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`.

**5.431.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back  
( size_type __n ) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1768 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**5.431.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front  
( size_type __n ) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1758 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**5.431.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (   
size_type __nodes_to_add = 1 ) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1794 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**5.431.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (   
size_type __nodes_to_add = 1 ) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1802 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**5.431.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::assign ( size_type __n, const  
value_type & __val ) [inline]`

Assigns a given value to a deque.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 925 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::operator=()`.

**5.431.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > void std::deque< _Tp, _Alloc  
>::assign ( _InputIterator __first, _InputIterator __last )  
[inline]`

Assigns a range to a deque.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a deque with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 942 of file stl\_deque.h.

```
5.431.3.19  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::deque<_Tp, _Alloc >::assign ( initializer_list<
             value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a deque with copies of the elements in the [initializer\\_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 961 of file stl\_deque.h.

References `std::deque<_Tp, _Alloc >::assign()`.

Referenced by `std::deque<_Tp, _Alloc >::assign()`.

```
5.431.3.20  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             reference std::deque<_Tp, _Alloc >::at ( size_type __n )
             [inline]
```

Provides access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read/write reference to data.

#### Exceptions

[std::out\\_of\\_range](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws [out\\_of\\_range](#) if the check fails.

Definition at line 1178 of file stl\_deque.h.

References `std::deque<_Tp, _Alloc >::_M_range_check()`.

**5.431.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque<_Tp, _Alloc>::at ( size_type __n )  
const [inline]`

Provides access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read-only (constant) reference to data.

#### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out\_of\_range* if the check fails.

Definition at line 1196 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_check()`.

**5.431.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque<_Tp, _Alloc>::back ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1235 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::end()`.

**5.431.3.23** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::deque<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1223 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::end()`.

**5.431.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 976 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc >::clear()`, `std::deque<_Tp, _Alloc >::deque()`, `std::deque<_Tp, _Alloc >::erase()`, `std::deque<_Tp, _Alloc >::front()`, `std::operator==( )`, and `std::deque<_Tp, _Alloc >::~~deque()`.

**5.431.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`const_iterator std::deque<_Tp, _Alloc >::begin ( ) const`  
`[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 984 of file `stl_deque.h`.

**5.431.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`const_iterator std::deque<_Tp, _Alloc >::cbegin ( ) const`  
`[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1047 of file `stl_deque.h`.

**5.431.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`const_iterator std::deque<_Tp, _Alloc >::cend ( ) const`  
`[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1056 of file `stl_deque.h`.

**5.431.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`void std::deque<_Tp, _Alloc >::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1501 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::begin()`.

Referenced by `std::deque<_Tp, _Alloc >::_M_range_initialize()`, `std::deque<_Tp, _Alloc >::erase()`, and `std::deque<_Tp, _Alloc >::operator=( )`.

**5.431.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque<_Tp, _Alloc>::crbegin ( )  
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1065 of file `stl_deque.h`.

**5.431.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque<_Tp, _Alloc>::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1074 of file `stl_deque.h`.

**5.431.3.31** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> deque<_Tp, _Alloc>::iterator deque::emplace ( iterator  
_position, _Args &&... _args )`

Inserts an object in deque before specified iterator.

#### Parameters

*position* An iterator into the deque.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...)`  before the specified location.

Definition at line 144 of file `deque.tcc`.

References `std::deque<_Tp, _Alloc>::push_back()`, and `std::deque<_Tp, _Alloc>::push_front()`.

Referenced by `std::deque<_Tp, _Alloc>::insert()`.

**5.431.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::deque<_Tp, _Alloc>::empty ( ) const [inline]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1122 of file stl\_deque.h.

**5.431.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc>::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 993 of file stl\_deque.h.

Referenced by `std::deque<_Tp, _Alloc>::back()`, `std::deque<_Tp, _Alloc>::deque()`, `std::deque<_Tp, _Alloc>::erase()`, `std::operator==( )`, and `std::deque<_Tp, _Alloc>::~~deque()`.

**5.431.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::deque<_Tp, _Alloc>::end ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1002 of file stl\_deque.h.

**5.431.3.35** `template<typename _Tp, typename _Alloc > deque<_Tp, _Alloc  
>::iterator deque::erase ( iterator __position )`

Remove element at given position.

#### Parameters

*position* Iterator pointing to element to be erased.

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 166 of file deque.tcc.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `std::deque<_Tp, _Alloc>::pop_back()`, `std::deque<_Tp, _Alloc>::pop_front()`, and `std::deque<_Tp, _Alloc>::size()`.

**5.431.3.36** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase ( iterator __first, iterator __last )`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 189 of file `deque.tcc`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

**5.431.3.37** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1215 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

**5.431.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1207 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

**5.431.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
allocator_type std::deque<_Tp, _Alloc >::get_allocator ( ) const  
[inline]`

Get a copy of the memory allocation object.

Reimplemented from [std::\\_Deque\\_base<\\_Tp, \\_Alloc >](#).

Definition at line 967 of file `stl_deque.h`.

**5.431.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc >::insert ( iterator __position,  
size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the deque.

#### Parameters

*position* An iterator into the deque.

*n* Number of elements to be inserted.

*x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Definition at line 1413 of file `stl_deque.h`.

**5.431.3.41** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc >::insert ( iterator __p,  
initializer_list<value_type > __l ) [inline]`

Inserts an initializer list into the deque.

#### Parameters

*p* An iterator into the deque.

*l* An [initializer\\_list](#).

This function will insert copies of the data in the [initializer\\_list](#) *l* into the deque before the location specified by *p*. This is known as *list insert*.

Definition at line 1399 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::insert()`.

Referenced by `std::deque<_Tp, _Alloc >::insert()`.

**5.431.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc >::insert ( iterator __position,  
value_type && __x ) [inline]`

Inserts given rvalue into deque before specified iterator.

#### Parameters

*position* An iterator into the deque.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1386 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::emplace()`.

**5.431.3.43** `template<typename _Tp, typename _Alloc > deque<_Tp, _Alloc  
>::iterator deque::insert ( iterator __position, const value_type &  
__x )`

Inserts given value into deque before specified iterator.

#### Parameters

*position* An iterator into the deque.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 121 of file `deque.tcc`.

References `std::deque<_Tp, _Alloc >::push_back()`, and `std::deque<_Tp, _Alloc >::push_front()`.

Referenced by `std::deque<_Tp, _Alloc >::resize()`.

```
5.431.3.44 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::deque< _Tp,
_Alloc >::insert ( iterator __position, _InputIterator __first,
_InputIterator __last ) [inline]
```

Inserts a range into the deque.

#### Parameters

*position* An iterator into the deque.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range [first,last) into the deque before the location specified by *pos*. This is known as *range insert*.

Definition at line 1428 of file stl\_deque.h.

```
5.431.3.45 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size ( ) const
[inline]
```

Returns the [size\(\)](#) of the largest possible deque.

Definition at line 1086 of file stl\_deque.h.

Referenced by `std::deque< _Tp, _Alloc >::M_new_elements_at_back()`, and `std::deque< _Tp, _Alloc >::M_new_elements_at_front()`.

```
5.431.3.46 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= ( initializer_list<
value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a deque with copies of the elements in the [initializer\\_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 907 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::assign()`.

**5.431.3.47** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc > & deque::operator= ( const deque< _Tp, _Alloc > & __x )`

Deque assignment operator.

#### Parameters

*x* A deque of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 65 of file deque.tcc.

**5.431.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= ( deque< _Tp, _Alloc > && __x ) [inline]`

Deque move assignment operator.

#### Parameters

*x* A deque of identical element and allocator types.

The contents of *x* are moved into this deque (without copying). *x* is a valid, but unspecified deque.

Definition at line 886 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::clear()`, and `std::deque< _Tp, _Alloc >::swap()`.

**5.431.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::operator[] ( size_type __n ) const [inline]`

Subscript access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1153 of file `stl_deque.h`.

```
5.431.3.50 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::deque<_Tp, _Alloc >::operator[] ( size_type __n )  
[inline]
```

Subscript access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1138 of file `stl_deque.h`.

```
5.431.3.51 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc >::pop_back ( ) [inline]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1336 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::_M_pop_back_aux()`.

Referenced by `std::deque<_Tp, _Alloc >::erase()`.

```
5.431.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc >::pop_front ( ) [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1315 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_front_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::erase()`.

**5.431.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::push_back ( const value_type &  
__x ) [inline]`

Add data to the end of the deque.

#### Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1284 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_push_back_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::_M_range_initialize()`, `std::deque<_Tp, _Alloc>::emplace()`, and `std::deque<_Tp, _Alloc>::insert()`.

**5.431.3.54** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::push_front ( const value_type &  
__x ) [inline]`

Add data to the front of the deque.

#### Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1253 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_push_front_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::emplace()`, and `std::deque<_Tp, _Alloc>::insert()`.

**5.431.3.55** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque<_Tp, _Alloc>::rbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1020 of file `stl_deque.h`.

**5.431.3.56** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::deque<_Tp, _Alloc>::rbegin ( )  
[inline]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1011 of file `stl_deque.h`.

**5.431.3.57** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque<_Tp, _Alloc>::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1038 of file `stl_deque.h`.

**5.431.3.58** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::deque<_Tp, _Alloc>::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1029 of file `stl_deque.h`.

**5.431.3.59** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::resize ( size_type __new_size,  
value_type __x = value_type() ) [inline]`

Resizes the deque to the specified number of elements.

#### Parameters

*new\_size* Number of elements the deque should contain.

*x* Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1101 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::insert()`, and `std::deque< _Tp, _Alloc >::size()`.

**5.431.3.60** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce memory use.

Definition at line 1113 of file `stl_deque.h`.

**5.431.3.61** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::deque< _Tp, _Alloc >::size ( ) const [inline]`

Returns the number of elements in the deque.

Definition at line 1081 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`, `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`, `std::deque< _Tp, _Alloc >::_M_range_check()`, `std::deque< _Tp, _Alloc >::erase()`, `std::operator==(,)`, and `std::deque< _Tp, _Alloc >::resize()`.

**5.431.3.62** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::swap ( deque< _Tp, _Alloc > &  
_x ) [inline]`

Swaps data with another deque.

### Parameters

**x** A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1481 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::operator=(,)`, and `std::swap()`.

The documentation for this class was generated from the following files:

**5.432 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference** **2329**

---

- [stl\\_deque.h](#)
- [deque.tcc](#)

**5.432 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference**

**Public Types**

- typedef \_RandomNumberEngine::result\_type [result\\_type](#)

**Public Member Functions**

- [discard\\_block\\_engine](#) ()
- [discard\\_block\\_engine](#) (const \_RandomNumberEngine &\_\_rne)
- [discard\\_block\\_engine](#) (result\_type \_\_s)
- template<typename \_Sseq , typename = typename std::enable\_if<!std::is\_same<\_Sseq, discard\_block\_engine>::value && !std::is\_same<\_Sseq, \_RandomNumberEngine>::value >::type> [discard\\_block\\_engine](#) (\_Sseq &\_\_q)
- [discard\\_block\\_engine](#) (\_RandomNumberEngine &&\_\_rne)
- const \_RandomNumberEngine & [base](#) () const
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- [result\\_type operator](#)() ()
- void [seed](#) (result\_type \_\_s)
- template<typename \_Sseq > void [seed](#) (\_Sseq &\_\_q)
- void [seed](#) ()

**Static Public Attributes**

- static const size\_t **block\_size**
- static const size\_t **used\_block**

**Friends**

- template<typename \_RandomNumberEngine1 , size\_t \_\_p1, size\_t \_\_r1, typename \_CharT , typename \_Traits >

```

std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &)

```

- bool operator== (const discard\_block\_engine &\_\_lhs, const discard\_block\_engine &\_\_rhs)
- template<typename \_RandomNumberEngine1 , size\_t \_\_p1, size\_t \_\_r1, typename \_CharT , typename \_Traits >

```

std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &)

```

### 5.432.1 Detailed Description

```

template<typename _RandomNumberEngine, size_t __p, size_t __r> class
std::discard_block_engine< _RandomNumberEngine, __p, __r >

```

Produces random numbers from some base engine by discarding blocks of data.

0 <= \_\_r <= \_\_p

Definition at line 784 of file random.h.

### 5.432.2 Member Typedef Documentation

5.432.2.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::result_type`

The type of the generated random value.

Definition at line 791 of file random.h.

### 5.432.3 Constructor & Destructor Documentation

5.432.3.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine ( ) [inline]`

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

Definition at line 802 of file random.h.

**5.432 std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r > Class**  
**Template Reference** **2331**

---

**5.432.3.2** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::discard_block_engine ( const _RandomNumberEngine & __rne )  
[inline, explicit]`

Copy constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

**Parameters**

*rng* An existing (base class) engine object.

Definition at line 812 of file random.h.

**5.432.3.3** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::discard_block_engine ( _RandomNumberEngine && __rne )  
[inline, explicit]`

Move constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

**Parameters**

*rng* An existing (base class) engine object.

Definition at line 822 of file random.h.

**5.432.3.4** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p,  
__r >::discard_block_engine ( result_type __s ) [inline,  
explicit]`

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

**Parameters**

*\_\_s* A seed value for the base class engine.

Definition at line 832 of file random.h.

```
5.432.3.5 template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _Sseq, typename = typename
std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value
&& !std::is_same<_Sseq, _RandomNumberEngine>::value>
::type> std::discard_block_engine< _RandomNumberEngine,
__p, __r >::discard_block_engine ( _Sseq & __q ) [inline,
explicit]
```

Generator construct a discard\_block\_engine engine.

#### Parameters

`__q` A seed sequence.

Definition at line 845 of file random.h.

### 5.432.4 Member Function Documentation

```
5.432.4.1 template<typename _RandomNumberEngine, size_t __p, size_t
__r> const _RandomNumberEngine& std::discard_block_engine<
_RandomNumberEngine, __p, __r >::base ( ) const [inline]
```

Gets a const reference to the underlying generator engine object.

Definition at line 889 of file random.h.

```
5.432.4.2 template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r
>::discard ( unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

#### Todo

Look for a faster way to do discard.

Definition at line 916 of file random.h.

```
5.432.4.3 template<typename _RandomNumberEngine, size_t __p, size_t __r>
result_type std::discard_block_engine< _RandomNumberEngine,
__p, __r >::max ( ) const [inline]
```

Gets the maximum value in the generated random number range.

**Todo**

This should be constexpr.

Definition at line 907 of file random.h.

**5.432.4.4** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
result_type std::discard_block_engine<_RandomNumberEngine,  
__p, __r >::min ( ) const [inline]`

Gets the minimum value in the generated random number range.

**Todo**

This should be constexpr.

Definition at line 898 of file random.h.

**5.432.4.5** `template<typename _RandomNumberEngine , size_t __p, size_t  
__r> discard_block_engine<_RandomNumberEngine, __p, __r  
>::result_type std::discard_block_engine<_RandomNumberEngine,  
__p, __r >::operator() ( )`

Gets the next value in the generated random number sequence.

Definition at line 662 of file random.tcc.

**5.432.4.6** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> template<typename _Sseq > void std::discard_block_engine<  
_RandomNumberEngine, __p, __r >::seed ( _Sseq & __q )  
[inline]`

Reseeds the `discard_block_engine` object with the given seed sequence.

**Parameters**

`__q` A seed generator function.

Definition at line 878 of file random.h.

**5.432.4.7** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
void std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::seed ( result_type __s ) [inline]`

Reseeds the `discard_block_engine` object with the default seed for the underlying base class generator engine.

Definition at line 865 of file random.h.

**5.432.4.8** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
void std::discard_block_engine< _RandomNumberEngine, __p, __r  
>::seed ( ) [inline]`

Reseeds the `discard_block_engine` object with the default seed for the underlying base class generator engine.

Definition at line 854 of file random.h.

## 5.432.5 Friends And Related Function Documentation

**5.432.5.1** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> template<typename _RandomNumberEngine1 ,  
size_t __p1, size_t __r1, typename _CharT , typename _Traits  
> std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream< _CharT, _Traits > & , const  
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 >  
& ) [friend]`

Inserts the current state of a `discard_block_engine` random number generator engine `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `discard_block_engine` random number generator engine.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.432.5.2** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> bool operator==( const discard_block_engine<  
_RandomNumberEngine, __p, __r > & __lhs, const  
discard_block_engine< _RandomNumberEngine, __p, __r > &  
__rhs ) [friend]`

Compares two `discard_block_engine` random number generator objects of the same type for equality.

### Parameters

`__lhs` A `discard_block_engine` random number generator object.

`__rhs` Another `discard_block_engine` random number generator object.

### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 940 of file `random.h`.

```
5.432.5.3 template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _RandomNumberEngine1
, size_t __p1, size_t __r1, typename _CharT, typename
_Traits > std::basic_istream<_CharT, _Traits>&
operator>> ( std::basic_istream<_CharT, _Traits > & ,
std::discard_block_engine<_RandomNumberEngine1, __p1, __r1 >
& ) [friend]
```

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A `discard_block_engine` random number generator engine.

### Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.433 `std::discrete_distribution<_IntType>` Class Template Reference

A [discrete\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- `template<typename _InputIterator >`  
**discrete\_distribution** (`_InputIterator __wbegin, _InputIterator __wend`)
- `template<typename _Func >`  
**discrete\_distribution** (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- **discrete\_distribution** (`const param_type &__p`)
- **discrete\_distribution** (`initializer_list< double > __wl`)
- [result\\_type](#) `max` () const
- [result\\_type](#) `min` () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **operator**() (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **operator**() (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (`const param_type &__param`)
- [param\\_type](#) `param` () const
- `std::vector< double >` [probabilities](#) () const
- void [reset](#) ()

## Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > &` **operator<<** (`std::basic_ostream< _CharT, _Traits > &, const std::discrete_distribution< _IntType1 > &`)
- `template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_istream< _CharT, _Traits > &` **operator>>** (`std::basic_istream< _CharT, _Traits > &, std::discrete_distribution< _IntType1 > &`)

### 5.433.1 Detailed Description

```
template<typename _IntType = int> class std::discrete_distribution< _IntType >
```

A [discrete\\_distribution](#) random number distribution. The formula for the discrete probability mass function is

Definition at line 4682 of file random.h.

## 5.433.2 Member Typedef Documentation

### 5.433.2.1 `template<typename _IntType = int> typedef _IntType std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4689 of file `random.h`.

## 5.433.3 Member Function Documentation

### 5.433.3.1 `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4795 of file `random.h`.

References `std::vector< _Tp, _Alloc >::size()`.

### 5.433.3.2 `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4788 of file `random.h`.

### 5.433.3.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type std::discrete_distribution< _IntType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4803 of file `random.h`.

References `std::discrete_distribution< _IntType >::operator()`, and `std::discrete_distribution< _IntType >::param()`.

Referenced by `std::discrete_distribution< _IntType >::operator()`.

### 5.433.3.4 `template<typename _IntType = int> void std::discrete_distribution< _IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

**Parameters**

`__param` The new parameter set of the distribution.

Definition at line 4781 of file random.h.

**5.433.3.5** `template<typename _IntType = int> param_type  
std::discrete_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4773 of file random.h.

Referenced by `std::discrete_distribution< _IntType >::operator()()`, and `std::operator==( )`.

**5.433.3.6** `template<typename _IntType = int> std::vector<double>  
std::discrete_distribution< _IntType >::probabilities ( ) const  
[inline]`

Returns the probabilities of the distribution.

Definition at line 4766 of file random.h.

**5.433.3.7** `template<typename _IntType = int> void std::discrete_distribution<  
_IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4759 of file random.h.

**5.433.4 Friends And Related Function Documentation**

**5.433.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & ,  
const std::discrete_distribution< _IntType1 > & ) [friend]`

Inserts a `discrete_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

`__os` An output stream.

## 5.434 std::discrete\_distribution< \_IntType >::param\_type Struct Reference 2339

`__x` A discrete\_distribution random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.433.4.2** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & ,  
std::discrete_distribution< _IntType1 > & ) [friend]`

Extracts a discrete\_distribution random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A discrete\_distribution random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## **5.434 std::discrete\_distribution< \_IntType >::param\_type Struct Reference**

### Public Types

- typedef `discrete_distribution< _IntType >` `distribution_type`

### Public Member Functions

- `template<typename _InputIterator >`  
`param_type` (`_InputIterator __wbegin, _InputIterator __wend`)
- `template<typename _Func >`  
`param_type` (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- `param_type` (`initializer_list< double > __wil`)
- `std::vector< double >` `probabilities` () const

## Friends

- class `discrete_distribution<_IntType>`
- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.434.1 Detailed Description

`template<typename _IntType = int> struct std::discrete_distribution<_IntType>::param_type`

Parameter type.

Definition at line 4691 of file `random.h`.

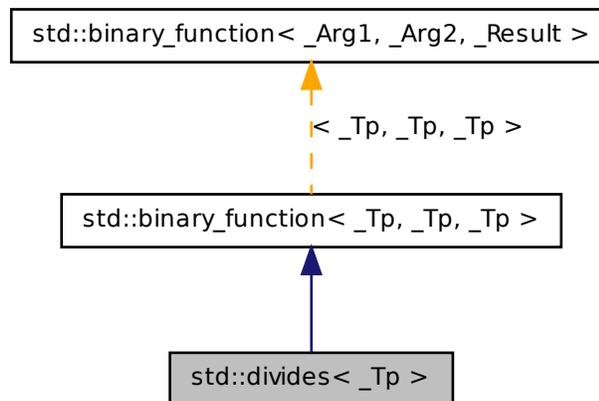
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

### 5.435 `std::divides<_Tp>` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::divides<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

### 5.435.1 Detailed Description

`template<typename _Tp> struct std::divides<_Tp>`

One of the [math functors](#).

Definition at line 162 of file `stl_function.h`.

### 5.435.2 Member Typedef Documentation

**5.435.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` `[inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.435.2.2** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` `[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.435.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` `[inherited]`

the type of the second argument

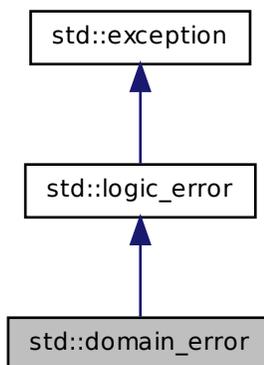
Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.436 std::domain\_error Class Reference

Inheritance diagram for std::domain\_error:



### Public Member Functions

- **domain\_error** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const throw ()

#### 5.436.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 73 of file `stdexcept`.

#### 5.436.2 Member Function Documentation

##### 5.436.2.1 virtual const char\* std::logic\_error::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

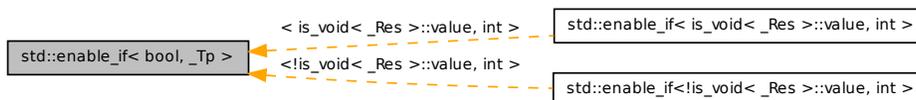
The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.437 `std::enable_if< bool, _Tp >` Struct Template Reference

[enable\\_if](#)

Inheritance diagram for `std::enable_if< bool, _Tp >`:



### 5.437.1 Detailed Description

```
template<bool, typename _Tp = void> struct std::enable_if< bool, _Tp >
```

[enable\\_if](#)

Definition at line 356 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.438 `std::enable_shared_from_this< _Tp >` Class Template Reference

Base class allowing use of member function `shared_from_this`.

## Public Member Functions

- [shared\\_ptr](#)< \_Tp > **shared\_from\_this** ()
- [shared\\_ptr](#)< const \_Tp > **shared\_from\_this** () const

## Protected Member Functions

- **enable\_shared\_from\_this** (const [enable\\_shared\\_from\\_this](#) &)
- [enable\\_shared\\_from\\_this](#) & **operator=** (const [enable\\_shared\\_from\\_this](#) &)

## Friends

- `template<typename _Tp1 >`  
void **\_\_enable\_shared\_from\_this\_helper** (const \_\_shared\_count<> &\_\_pn,  
const [enable\\_shared\\_from\\_this](#) \*\_\_pe, const \_Tp1 \*\_\_px)

### 5.438.1 Detailed Description

`template<typename _Tp> class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

Definition at line 398 of file `shared_ptr.h`.

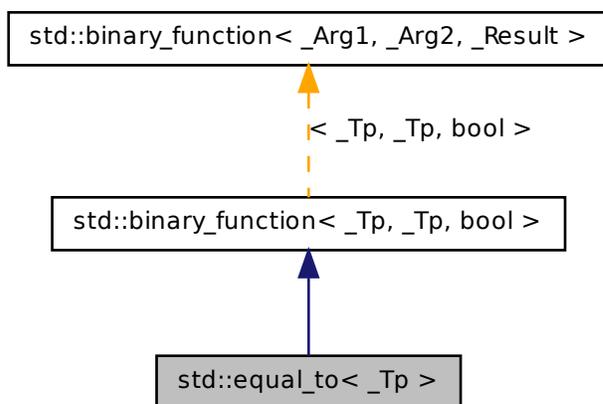
The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

### 5.439 `std::equal_to`< \_Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::equal\_to< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.439.1 Detailed Description

```
template<typename _Tp> struct std::equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 199 of file `stl_function.h`.

## 5.439.2 Member Typedef Documentation

**5.439.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.439.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.439.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.440 `std::error_category` Class Reference

[error\\_category](#)

### Public Member Functions

- `error_category` (const [error\\_category](#) &)
- virtual `error_condition default_error_condition` (int \_\_i) const
- virtual bool `equivalent` (const [error\\_code](#) &\_\_code, int \_\_i) const
- virtual bool `equivalent` (int \_\_i, const [error\\_condition](#) &\_\_cond) const
- virtual `string message` (int) const =0
- virtual const char \* `name` () const =0
- bool `operator!=` (const [error\\_category](#) &\_\_other) const
- bool `operator<` (const [error\\_category](#) &\_\_other) const
- [error\\_category](#) & `operator=` (const [error\\_category](#) &)
- bool `operator==` (const [error\\_category](#) &\_\_other) const

### 5.440.1 Detailed Description

[error\\_category](#)

Definition at line 64 of file system\_error.

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.441 std::error\_code Struct Reference

[error\\_code](#)

### Public Member Functions

- **error\_code** (int \_\_v, const [error\\_category](#) &\_\_cat)
- template<typename \_ErrorCodeEnum >  
**error\_code** (\_ErrorCodeEnum \_\_e, typename [enable\\_if](#)< [is\\_error\\_code\\_enum](#)< \_ErrorCodeEnum >::value >::type \*=0)
- void **assign** (int \_\_v, const [error\\_category](#) &\_\_cat)
- const [error\\_category](#) & **category** () const
- void **clear** ()
- [error\\_condition](#) **default\_error\_condition** () const
- [string](#) **message** () const
- **operator bool** () const
- template<typename \_ErrorCodeEnum >  
[enable\\_if](#)< [is\\_error\\_code\\_enum](#)< \_ErrorCodeEnum >::value, [error\\_code](#) & >::type **operator=** (\_ErrorCodeEnum \_\_e)
- int **value** () const

### Friends

- class [hash](#)< [error\\_code](#) >

### 5.441.1 Detailed Description

[error\\_code](#)

Definition at line 116 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.442 `std::error_condition` Struct Reference

[error\\_condition](#)

### Public Member Functions

- **error\_condition** (int \_\_v, const [error\\_category](#) &\_\_cat)
- `template<typename _ErrorConditionEnum >`  
**error\_condition** (\_ErrorConditionEnum \_\_e, typename [enable\\_if](#)< [is\\_error\\_ - condition\\_enum](#)< \_ErrorConditionEnum >::value >::type !=0)
- void **assign** (int \_\_v, const [error\\_category](#) &\_\_cat)
- const [error\\_category](#) & **category** () const
- void **clear** ()
- [string](#) **message** () const
- **operator bool** () const
- `template<typename _ErrorConditionEnum >`  
[enable\\_if](#)< [is\\_error\\_condition\\_enum](#)< \_ErrorConditionEnum >::value, [error\\_ - condition](#) & >::type **operator=** (\_ErrorConditionEnum \_\_e)
- int **value** () const

### 5.442.1 Detailed Description

[error\\_condition](#)

Definition at line 193 of file `system_error`.

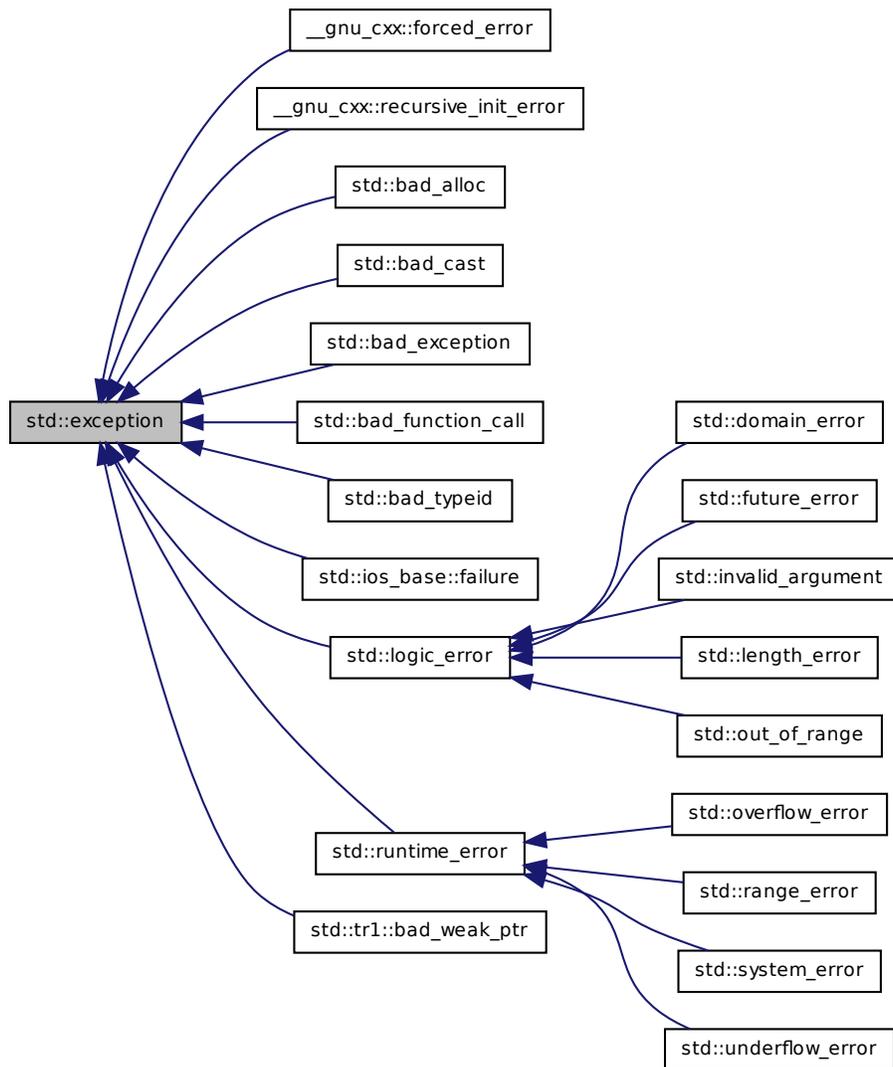
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.443 `std::exception` Class Reference

Base class for all library exceptions.

Inheritance diagram for std::exception:



## Public Member Functions

- virtual const char \* [what](#) () const throw ()

### 5.443.1 Detailed Description

Base class for all library exceptions. This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 61 of file `exception`.

### 5.443.2 Member Function Documentation

#### 5.443.2.1 virtual const char\* `std::exception::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_exception](#), [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_typeid](#), [std::future\\_error](#), [std::logic\\_error](#), [std::runtime\\_error](#), [std::tr1::bad\\_weak\\_ptr](#), and [std::ios\\_base::failure](#).

The documentation for this class was generated from the following file:

- [exception](#)

## 5.444 `std::exponential_distribution< _RealType >` Class Template Reference

An exponential continuous distribution for random numbers.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## 5.444 `std::exponential_distribution<_RealType>` Class Template Reference 2351

### Public Member Functions

- `exponential_distribution` (const `result_type` &\_\_lambda=`result_type`(1))
- `exponential_distribution` (const `param_type` &\_\_p)
- `_RealType lambda` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (\_UniformRandomNumberGenerator &\_\_urng)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (\_UniformRandomNumberGenerator &\_\_urng, const `param_type` &\_\_p)
- `param_type param` () const
- `void param` (const `param_type` &\_\_param)
- `void reset` ()

### 5.444.1 Detailed Description

`template<typename _RealType = double> class std::exponential_distribution<_RealType >`

An exponential continuous distribution for random numbers. The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Table 5.1: Distribution Statistics

Definition at line 4161 of file `random.h`.

### 5.444.2 Member Typedef Documentation

**5.444.2.1** `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType >::result_type`

The type of the range of the distribution.

Definition at line 4168 of file `random.h`.

### 5.444.3 Constructor & Destructor Documentation

**5.444.3.1** `template<typename _RealType = double> std::exponential_distribution< _RealType >::exponential_distribution ( const result_type & lambda = result_type(1) ) [inline, explicit]`

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4199 of file random.h.

### 5.444.4 Member Function Documentation

**5.444.4.1** `template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda ( ) const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4220 of file random.h.

**5.444.4.2** `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4249 of file random.h.

**5.444.4.3** `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4242 of file random.h.

**5.444.4.4** `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::exponential_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & urng ) [inline]`

Generating functions.

Definition at line 4257 of file random.h.

## **5.444 std::exponential\_distribution<\_RealType> Class Template Reference**

References `std::exponential_distribution<_RealType>::operator()`, and `std::exponential_distribution<_RealType>::param()`.

Referenced by `std::exponential_distribution<_RealType>::operator()`.

**5.444.4.5** `template<typename _RealType = double> param_type  
std::exponential_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4227 of file `random.h`.

Referenced by `std::exponential_distribution<_RealType>::operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.444.4.6** `template<typename _RealType = double> void  
std::exponential_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

### **Parameters**

`__param` The new parameter set of the distribution.

Definition at line 4235 of file `random.h`.

**5.444.4.7** `template<typename _RealType = double> void  
std::exponential_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4214 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.445 `std::exponential_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef [exponential\\_distribution< \\_RealType >](#) `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __lambda=_RealType(1)`)
- `_RealType lambda` () const

### Friends

- bool `operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.445.1 Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution< _RealType >::param_type`

Parameter type.

Definition at line 4170 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.446 `std::extreme_value_distribution< _RealType >` Class Template Reference

A [extreme\\_value\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` `result_type`

## Public Member Functions

- `extreme_value_distribution` (`_RealType __a=_RealType(0)`, `_RealType __b=_RealType(1)`)
- `extreme_value_distribution` (const `param_type` &\_\_p)
- `_RealType a` () const
- `_RealType b` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator`>  
`result_type operator`() (`_UniformRandomNumberGenerator` &\_\_urng, const `param_type` &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
`result_type operator`() (`_UniformRandomNumberGenerator` &\_\_urng)
- void `param` (const `param_type` &\_\_param)
- `param_type param` () const
- void `reset` ()

### 5.446.1 Detailed Description

`template<typename _RealType = double> class std::extreme_value_distribution<_RealType>`

A `extreme_value_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 4510 of file `random.h`.

### 5.446.2 Member Typedef Documentation

**5.446.2.1** `template<typename _RealType = double> typedef _RealType std::extreme_value_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4517 of file `random.h`.

### 5.446.3 Member Function Documentation

**5.446.3.1** `template<typename _RealType = double> _RealType  
std::extreme_value_distribution< _RealType >::a ( ) const  
[inline]`

Return the  $a$  parameter of the distribution.

Definition at line 4568 of file random.h.

**5.446.3.2** `template<typename _RealType = double> _RealType  
std::extreme_value_distribution< _RealType >::b ( ) const  
[inline]`

Return the  $b$  parameter of the distribution.

Definition at line 4575 of file random.h.

**5.446.3.3** `template<typename _RealType = double> result_type  
std::extreme_value_distribution< _RealType >::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4604 of file random.h.

**5.446.3.4** `template<typename _RealType = double> result_type  
std::extreme_value_distribution< _RealType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4597 of file random.h.

**5.446.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::extreme_value_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4612 of file random.h.

References `std::extreme_value_distribution< _RealType >::operator()`, and `std::extreme_value_distribution< _RealType >::param()`.

## 5.447 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference 2357

---

Referenced by `std::extreme_value_distribution<_RealType>::operator()`.

**5.446.3.6** `template<typename _RealType = double> void  
std::extreme_value_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

### Parameters

`__param` The new parameter set of the distribution.

Definition at line 4590 of file `random.h`.

**5.446.3.7** `template<typename _RealType = double> param_type  
std::extreme_value_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4582 of file `random.h`.

Referenced by `std::extreme_value_distribution<_RealType>::operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.446.3.8** `template<typename _RealType = double> void  
std::extreme_value_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 4561 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.447 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `extreme_value_distribution<_RealType>` `distribution_type`

## Public Member Functions

- **param\_type** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.447.1 Detailed Description

**template<typename \_RealType = double> struct std::extreme\_value\_distribution< \_RealType >::param\_type**

Parameter type.

Definition at line 4519 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.448 std::fisher\_f\_distribution< \_RealType > Class Template Reference

A [fisher\\_f\\_distribution](#) random number distribution.

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **fisher\_f\_distribution** (\_RealType \_\_m=\_RealType(1), \_RealType \_\_n=\_RealType(1))
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)

- `_RealType m () const`
- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng, const`  
`param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

## Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`  
`CharT, _Traits > &, const std::fisher_f_distribution< _RealType1 > &)`
- `template<typename _RealType1 >`  
`bool operator==(const std::fisher_f_distribution< _RealType1 > &__d1, const`  
`std::fisher_f_distribution< _RealType1 > &__d2)`
- `template<typename _RealType1 , typename _CharT , typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`  
`CharT, _Traits > &, std::fisher_f_distribution< _RealType1 > &)`

### 5.448.1 Detailed Description

`template<typename _RealType = double> class std::fisher_f_distribution< _-`  
`RealType >`

A `fisher_f_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 2877 of file `random.h`.

### 5.448.2 Member Typedef Documentation

**5.448.2.1** `template<typename _RealType = double> typedef _RealType`  
`std::fisher_f_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2884 of file random.h.

### 5.448.3 Member Function Documentation

**5.448.3.1** `template<typename _RealType = double> result_type  
std::fisher_f_distribution<_RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2971 of file random.h.

**5.448.3.2** `template<typename _RealType = double> result_type  
std::fisher_f_distribution<_RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2964 of file random.h.

**5.448.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::fisher_f_distribution<_RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2979 of file random.h.

**5.448.3.4** `template<typename _RealType = double> void  
std::fisher_f_distribution<_RealType >::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2957 of file random.h.

**5.448.3.5** `template<typename _RealType = double> param_type  
std::fisher_f_distribution<_RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2949 of file `random.h`.

**5.448.3.6** `template<typename _RealType = double> void  
std::fisher_f_distribution<_RealType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 2928 of file `random.h`.

References `std::gamma_distribution<_RealType>::reset()`.

## 5.448.4 Friends And Related Function Documentation

**5.448.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> &, const  
std::fisher_f_distribution<_RealType1> & ) [friend]`

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `fisher_f_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.448.4.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::fisher_f_distribution<  
_RealType1 > & __d1, const std::fisher_f_distribution<_RealType1  
> & __d2 ) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3000 of file `random.h`.

```

5.448.4.3 template<typename _RealType = double> template<typename
          _RealType1 , typename _CharT , typename _Traits >
          std::basic_istream<_CharT, _Traits>& operator>>
          ( std::basic_istream< _CharT, _Traits > & ,
            std::fisher_f_distribution< _RealType1 > & ) [friend]

```

Extracts a fisher\_f\_distribution random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A fisher\_f\_distribution random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.449 std::fisher\_f\_distribution< \_RealType >::param\_type Struct Reference

### Public Types

- typedef [fisher\\_f\\_distribution](#)< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_m=\_RealType(1), \_RealType \_\_n=\_RealType(1))
- **\_RealType m** () const
- **\_RealType n** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.449.1 Detailed Description

```
template<typename _RealType = double> struct std::fisher_f_distribution< _  
RealType >::param_type
```

Parameter type.

Definition at line 2886 of file random.h.

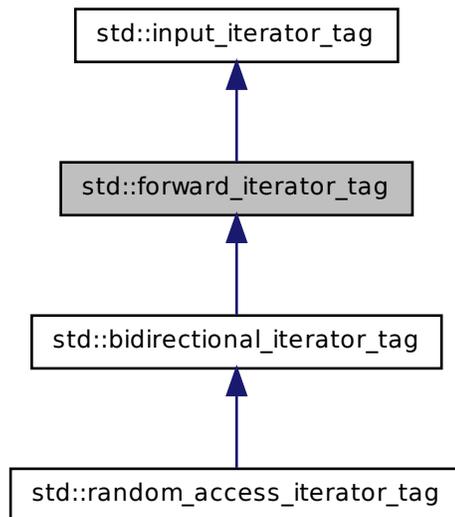
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.450 std::forward\_iterator\_tag Struct Reference

Forward iterators support a superset of input iterator operations.

Inheritance diagram for std::forward\_iterator\_tag:



### 5.450.1 Detailed Description

Forward iterators support a superset of input iterator operations.

Definition at line 91 of file `stl_iterator_base_types.h`.

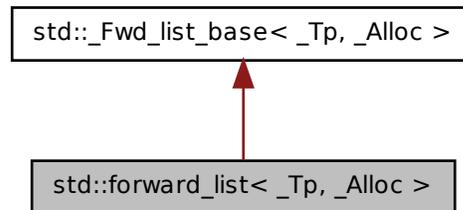
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.451 `std::forward_list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::forward_list< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Fwd_list_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**

- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [forward\\_list](#) (const `_Alloc` &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list, const `_Alloc` &\_\_al)
- [forward\\_list](#) (size\_type \_\_n)
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list)
- [forward\\_list](#) ([std::initializer\\_list](#)< `_Tp` > \_\_il, const `_Alloc` &\_\_al=\_Alloc())
- [forward\\_list](#) (size\_type \_\_n, const `_Tp` &\_\_value, const `_Alloc` &\_\_al=\_Alloc())
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list, const `_Alloc` &\_\_al)
- template<typename `_InputIterator` >  
[forward\\_list](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Alloc` &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list)
- [~forward\\_list](#) ()
- void [assign](#) ([std::initializer\\_list](#)< `_Tp` > \_\_il)
- template<typename `_InputIterator` >  
void [assign](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void [assign](#) (size\_type \_\_n, const `_Tp` &\_\_val)
- [iterator before\\_begin](#) ()
- [const\\_iterator before\\_begin](#) () const
- [iterator begin](#) ()
- [const\\_iterator begin](#) () const
- [const\\_iterator cbefore\\_begin](#) () const
- [const\\_iterator cbegin](#) () const
- [const\\_iterator cend](#) () const
- void [clear](#) ()
- template<typename... `_Args`>  
[iterator emplace\\_after](#) ([const\\_iterator](#) \_\_pos, `_Args` &&...\_\_args)
- template<typename... `_Args`>  
void [emplace\\_front](#) (`_Args` &&...\_\_args)
- bool [empty](#) () const
- [const\\_iterator end](#) () const
- [iterator end](#) ()
- void [erase\\_after](#) ([const\\_iterator](#) \_\_pos)
- void [erase\\_after](#) ([const\\_iterator](#) \_\_pos, [const\\_iterator](#) \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const

- `template<typename _InputIterator >`  
`iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator __last)`
- `iterator insert_after (const_iterator __pos, const _Tp &__val)`
- `iterator insert_after (const_iterator __pos, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, size_type __n, const _Tp &__val)`
- `iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)`
- `size_type max_size () const`
- `template<typename _Comp >`  
`void merge (forward_list &&__list, _Comp __comp)`
- `void merge (forward_list &&__list)`
- `forward_list & operator= (std::initializer_list< _Tp > __il)`
- `forward_list & operator= (const forward_list &__list)`
- `forward_list & operator= (forward_list &&__list)`
- `void pop_front ()`
- `void push_front (const _Tp &__val)`
- `void push_front (_Tp &&__val)`
- `void remove (const _Tp &__val)`
- `template<typename _Pred >`  
`void remove_if (_Pred __pred)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, value_type __val)`
- `void reverse ()`
- `template<typename _Comp >`  
`void sort (_Comp __comp)`
- `void sort ()`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __before, const_iterator __last)`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i)`
- `void splice_after (const_iterator __pos, forward_list &&__list)`
- `void swap (forward_list &__list)`
- `template<typename _BinPred >`  
`void unique (_BinPred __binary_pred)`
- `void unique ()`

## Private Types

- `typedef _Alloc::template rebind< _Fwd_list_node< _Tp > >::other _Node_alloc_type`

## Private Member Functions

- `template<typename... _Args>`  
`_Node * _M_create_node (_Args &&...__args)`
- `void _M_erase_after (_Fwd_list_node_base * __pos)`
- `void _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `const _Node_alloc_type & _M_get_Node_allocator () const`
- `_Node_alloc_type & _M_get_Node_allocator ()`
- `template<typename... _Args>`  
`_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void _M_put_node (_Node * __p)`

## Private Attributes

- `_Fwd_list_impl _M_impl`

### 5.451.1 Detailed Description

`template<typename _Tp, typename _Alloc = allocator<_Tp>> class std::forward_list<_Tp, _Alloc >`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `forward_list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `Fwd_list_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `forward_list<X, Alloc1 >` are spliced into `forward_list<X, Alloc2 >`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Definition at line 405 of file `forward_list.h`.

## 5.451.2 Constructor & Destructor Documentation

**5.451.2.1** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( const _Alloc & __al  
= _Alloc() ) [inline, explicit]`

Creates a forward\_list with no elements.

### Parameters

*al* An allocator object.

Definition at line 434 of file forward\_list.h.

**5.451.2.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( const forward_list<  
_Tp, _Alloc > & __list, const _Alloc & __al ) [inline]`

Copy constructor with allocator argument.

### Parameters

*list* Input list to copy.

*al* An allocator object.

Definition at line 443 of file forward\_list.h.

**5.451.2.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( forward_list<_Tp,  
_Alloc > && __list, const _Alloc & __al ) [inline]`

Move constructor with allocator argument.

### Parameters

*list* Input list to move.

*al* An allocator object.

Definition at line 452 of file forward\_list.h.

**5.451.2.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( size_type __n )  
[inline, explicit]`

Creates a forward\_list with default constructed elements.

**Parameters**

*n* The number of elements to initially create.

This constructor creates the forward\_list with *n* default constructed elements.

Definition at line 464 of file forward\_list.h.

```
5.451.2.5 template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n,
const _Tp & __value, const _Alloc & __al = _Alloc() )
[inline]
```

Creates a forward\_list with copies of an exemplar element.

**Parameters**

*n* The number of elements to initially create.

*value* An element to copy.

*al* An allocator object.

This constructor fills the forward\_list with *n* copies of *value*.

Definition at line 477 of file forward\_list.h.

```
5.451.2.6 template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator > std::forward_list< _Tp, _Alloc
>::forward_list ( _InputIterator __first, _InputIterator __last,
const _Alloc & __al = _Alloc() ) [inline]
```

Builds a forward\_list from a range.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*al* An allocator object.

Create a forward\_list consisting of copies of the elements from [*first,last*). This is linear in N (where N is distance(*first,last*)).

Definition at line 493 of file forward\_list.h.

**5.451.2.7** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( const forward_list<  
_Tp, _Alloc > & __list ) [inline]`

The `forward_list` copy constructor.

#### Parameters

*list* A `forward_list` of identical element and allocator types.

The newly-created `forward_list` uses a copy of the allocation object used by *list*.

Definition at line 510 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc >::begin()`, and `std::forward_list<_Tp, _Alloc >::end()`.

**5.451.2.8** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( forward_list<_Tp,  
_Alloc > && __list ) [inline]`

The `forward_list` move constructor.

#### Parameters

*list* A `forward_list` of identical element and allocator types.

The newly-created `forward_list` contains the exact contents of *forward\_list*. The contents of *list* are a valid, but unspecified `forward_list`.

Definition at line 523 of file `forward_list.h`.

**5.451.2.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc >::forward_list ( std::initializer_list<  
_Tp > __il, const _Alloc & __al = _Alloc() ) [inline]`

Builds a `forward_list` from an `initializer_list`.

#### Parameters

*il* An `initializer_list` of value\_type.

*al* An allocator object.

Create a `forward_list` consisting of copies of the elements in the `initializer_list` *il*. This is linear in `il.size()`.

Definition at line 534 of file `forward_list.h`.

**5.451.2.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list< _Tp, _Alloc >::~~forward_list ( ) [inline]`

The `forward_list` dtor.

Definition at line 542 of file `forward_list.h`.

### 5.451.3 Member Function Documentation

**5.451.3.1** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
template<typename _InputIterator > void std::forward_list< _Tp,  
_Alloc >::assign ( _InputIterator __first, _InputIterator __last )  
[inline]`

Assigns a range to a `forward_list`.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a `forward_list` with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 604 of file `forward_list.h`.

**5.451.3.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::assign ( size_type __n, const _Tp  
& __val ) [inline]`

Assigns a given value to a `forward_list`.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a `forward_list` with *n* copies of the given value. Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 621 of file `forward_list.h`.

**5.451.3.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::assign ( std::initializer_list<_Tp  
> il ) [inline]`

Assigns an [initializer\\_list](#) to a `forward_list`.

#### Parameters

*il* An [initializer\\_list](#) of `value_type`.

Replace the contents of the `forward_list` with copies of the elements in the [initializer\\_list](#) *il*. This is linear in `il.size()`.

Definition at line 636 of file `forward_list.h`.

**5.451.3.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list<_Tp, _Alloc>::before_begin ( )  
[inline]`

Returns a read/write iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 654 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::resize()`.

**5.451.3.5** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list<_Tp, _Alloc>::before_begin ( )  
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 663 of file `forward_list.h`.

**5.451.3.6** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list<_Tp, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 671 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::unique()`.

**5.451.3.7** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list<_Tp, _Alloc>::begin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 680 of file `forward_list.h`.

**5.451.3.8** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list<_Tp, _Alloc>::cbefore_begin ( )  
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 716 of file `forward_list.h`.

**5.451.3.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list<_Tp, _Alloc>::cbegin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 707 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::operator==( )`.

**5.451.3.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list<_Tp, _Alloc>::cend ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 725 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::operator==( )`.

**5.451.3.11** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::clear ( ) [inline]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves

are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1014 of file `forward_list.h`.

```
5.451.3.12 template<typename _Tp, typename _Alloc = allocator<_Tp>>
             template<typename... _Args> iterator std::forward_list<_Tp,
             _Alloc >::emplace_after ( const_iterator __pos, _Args &&...
             __args ) [inline]
```

Constructs object in `forward_list` after the specified iterator.

#### Parameters

*pos* A `const_iterator` into the `forward_list`.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...)`  after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 838 of file `forward_list.h`.

```
5.451.3.13 template<typename _Tp, typename _Alloc = allocator<_Tp>>
             template<typename... _Args> void std::forward_list<_Tp, _Alloc
             >::emplace_front ( _Args &&... __args ) [inline]
```

Constructs object in `forward_list` at the front of the list.

#### Parameters

*args* Arguments.

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...)`  at the front of the list. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 782 of file `forward_list.h`.

**5.451.3.14** `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool  
std::forward_list< _Tp, _Alloc >::empty ( ) const [inline]`

Returns true if the forward\_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 733 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

**5.451.3.15** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 689 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::resize()`, and `std::forward_list< _Tp, _Alloc >::unique()`.

**5.451.3.16** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::end ( ) const  
[inline]`

Returns a read-only iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 698 of file forward\_list.h.

**5.451.3.17** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::erase_after ( const_iterator  
__pos, const_iterator __last ) [inline]`

Remove a range of elements.

#### Parameters

*pos* Iterator pointing before the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

This function will erase the elements in the range (pos,last) and shorten the forward\_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only

erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 958 of file `forward_list.h`.

**5.451.3.18** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc >::erase_after ( const_iterator  
_pos ) [inline]`

Removes the element pointed to by the iterator following `pos`.

#### Parameters

*pos* Iterator pointing before element to be erased.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 937 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc >::operator=()`, `std::forward_list<_Tp, _Alloc >::resize()`, and `std::forward_list<_Tp, _Alloc >::unique()`.

**5.451.3.19** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
reference std::forward_list<_Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the `forward_list`.

Definition at line 750 of file `forward_list.h`.

**5.451.3.20** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_reference std::forward_list<_Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

Definition at line 761 of file `forward_list.h`.

```
5.451.3.21 template<typename _Tp, typename _Alloc = allocator<_Tp>>
        allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ( )
        const [inline]
```

Get a copy of the memory allocation object.

Definition at line 644 of file forward\_list.h.

```
5.451.3.22 template<typename _Tp, typename _Alloc = allocator<_Tp>>
        iterator std::forward_list< _Tp, _Alloc >::insert_after (
        const_iterator __pos, const _Tp & __val ) [inline]
```

Inserts given value into forward\_list after specified iterator.

#### Parameters

*pos* An iterator into the forward\_list.

*val* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 855 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::forward\_list< \_Tp, \_Alloc >::resize().

```
5.451.3.23 template<typename _Tp, typename _Alloc > forward_list< _Tp,
        _Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (
        const_iterator __pos, size_type __n, const _Tp & __val )
```

Inserts a number of copies of given data into the forward\_list.

#### Parameters

*pos* An iterator into the forward\_list.

*n* Number of elements to be inserted.

*val* Data to be inserted.

#### Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 244 of file forward\_list.tcc.

**5.451.3.24** `template<typename _Tp, typename _Alloc > template<typename  
_InputIterator > forward_list< _Tp, _Alloc >::iterator  
std::forward_list< _Tp, _Alloc >::insert_after ( const_iterator  
__pos, _InputIterator __first, _InputIterator __last )`

Inserts a range into the forward\_list.

#### Parameters

*position* An iterator into the forward\_list.

*first* An input iterator.

*last* An input iterator.

#### Returns

An iterator pointing to the last inserted element or *pos* if *first* == *last*.

This function will insert copies of the data in the range [*first,last*) into the forward\_list after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 259 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::empty().

**5.451.3.25** `template<typename _Tp, typename _Alloc > forward_list< _Tp,  
_Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (   
const_iterator __pos, std::initializer_list< _Tp > __il )`

Inserts the contents of an [initializer\\_list](#) into forward\_list after the specified iterator.

#### Parameters

*pos* An iterator into the forward\_list.

*il* An [initializer\\_list](#) of value\_type.

#### Returns

An iterator pointing to the last inserted element or *pos* if *il* is empty.

This function will insert copies of the data in the `initializer_list` *il* into the `forward_list` before the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 272 of file `forward_list.tcc`.

**5.451.3.26** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const  
[inline]`

Returns the largest possible size of `forward_list`.

Definition at line 740 of file `forward_list.h`.

**5.451.3.27** `template<typename _Tp , typename _Alloc > template<typename  
_Comp > void std::forward_list< _Tp, _Alloc >::merge (   
forward_list< _Tp, _Alloc > && __list, _Comp __comp )`

Merge sorted lists according to comparison function.

#### Parameters

*list* Sorted list to merge.

*comp* Comparison function defining sort order.

Assumes that both *list* and this list are sorted according to *comp*. Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equivalent according to *comp*().

Definition at line 339 of file `forward_list.tcc`.

**5.451.3.28** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::merge ( forward_list< _Tp,  
_Alloc > && __list ) [inline]`

Merge sorted lists.

#### Parameters

*list* Sorted list to merge.

Assumes that both *list* and this list are sorted according to operator<(). Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equal.

Definition at line 1145 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::merge().

Referenced by std::forward\_list< \_Tp, \_Alloc >::merge().

**5.451.3.29** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
forward_list& std::forward_list< _Tp, _Alloc >::operator= (  
forward_list< _Tp, _Alloc > && __list ) [inline]`

The forward\_list move assignment operator.

#### Parameters

*list* A forward\_list of identical element and allocator types.

The contents of *list* are moved into this forward\_list (without copying). *list* is a valid, but unspecified forward\_list

Definition at line 566 of file forward\_list.h.

References std::swap().

**5.451.3.30** `template<typename _Tp, typename _Alloc > forward_list< _Tp,  
_Alloc > & std::forward_list< _Tp, _Alloc >::operator= ( const  
forward_list< _Tp, _Alloc > & __list )`

The forward\_list assignment operator.

#### Parameters

*list* A forward\_list of identical element and allocator types.

All the elements of *list* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 140 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), std::forward\_list< \_Tp, \_Alloc >::begin(), std::forward\_list< \_Tp, \_Alloc >::cbegin(), std::forward\_list< \_Tp, \_Alloc >::cend(), std::forward\_list< \_Tp, \_Alloc >::end(), std::forward\_list< \_Tp, \_Alloc >::erase\_after(), and std::forward\_list< \_Tp, \_Alloc >::insert\_after().

**5.451.3.31** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
forward_list& std::forward_list< _Tp, _Alloc >::operator= (  
std::initializer_list<_Tp > __il ) [inline]`

The forward\_list initializer list assignment operator.

**Parameters**

*il* An [initializer\\_list](#) of value\_type.

Replace the contents of the forward\_list with copies of the elements in the [initializer\\_list](#) *il*. This is linear in *il.size()*.

Definition at line 584 of file forward\_list.h.

**5.451.3.32** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward\_list by one. Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop\\_front\(\)](#) is called.

Definition at line 820 of file forward\_list.h.

**5.451.3.33** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::push_front ( const _Tp & __val  
) [inline]`

Add data to the front of the forward\_list.

**Parameters**

*val* Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 797 of file forward\_list.h.

**5.451.3.34** `template<typename _Tp, typename _Alloc > void  
std::forward_list< _Tp, _Alloc >::remove ( const _Tp & __val )`

Remove all elements equal to value.

**Parameters**

*val* The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 286 of file forward\_list.tcc.

**5.451.3.35** `template<typename _Tp , typename _Alloc > template<typename  
_Pred > void std::forward_list< _Tp, _Alloc >::remove_if ( _Pred  
_pred )`

Remove all elements satisfying a predicate.

#### Parameters

*pred* Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 302 of file forward\_list.tcc.

**5.451.3.36** `template<typename _Tp , typename _Alloc > void  
std::forward_list< _Tp, _Alloc >::resize ( size_type __sz )`

Resizes the forward\_list to the specified number of elements.

#### Parameters

*sz* Number of elements the forward\_list should contain.

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current size the forward\_list is truncated, otherwise the forward\_list is extended and the new elements are default constructed.

Definition at line 185 of file forward\_list.tcc.

References `std::forward_list< _Tp, _Alloc >::before_begin()`, `std::forward_list< _Tp, _Alloc >::end()`, and `std::forward_list< _Tp, _Alloc >::erase_after()`.

**5.451.3.37** `template<typename _Tp , typename _Alloc > void  
std::forward_list< _Tp, _Alloc >::resize ( size_type __sz,  
value_type __val )`

Resizes the forward\_list to the specified number of elements.

**Parameters**

*sz* Number of elements the `forward_list` should contain.

*val* Data with which new elements should be populated.

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current size the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

Definition at line 204 of file `forward_list.tcc`.

References `std::forward_list<_Tp, _Alloc >::before_begin()`, `std::forward_list<_Tp, _Alloc >::end()`, `std::forward_list<_Tp, _Alloc >::erase_after()`, and `std::forward_list<_Tp, _Alloc >::insert_after()`.

**5.451.338** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc >::reverse ( ) [inline]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1189 of file `forward_list.h`.

**5.451.339** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc >::sort ( ) [inline]`

Sort the elements of the list.

Sorts the elements of this list in  $N\log N$  time. Equivalent elements remain in list order.

Definition at line 1170 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc >::sort()`.

Referenced by `std::forward_list<_Tp, _Alloc >::sort()`.

**5.451.340** `template<typename _Tp, class _Alloc > template<typename _Comp > void std::forward_list<_Tp, _Alloc >::sort ( _Comp __comp )`

Sort the `forward_list` using a comparison function.

Sorts the elements of this list in  $N\log N$  time. Equivalent elements remain in list order.

Definition at line 385 of file `forward_list.tcc`.

**5.451.3.41** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::splice_after ( const_iterator  
__pos, forward_list<_Tp, _Alloc> && __list ) [inline]`

Insert contents of another forward\_list.

#### Parameters

*pos* Iterator referencing the element to insert after.

*list* Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*.  
*list* becomes an empty list.

Requires this != *x*.

Definition at line 1031 of file forward\_list.h.

**5.451.3.42** `template<typename _Tp, typename _Alloc > void  
std::forward_list<_Tp, _Alloc>::splice_after ( const_iterator  
__pos, forward_list<_Tp, _Alloc> && __list, const_iterator  
__before, const_iterator __last )`

Insert range from another forward\_list.

#### Parameters

*pos* Iterator referencing the element to insert after.

*list* Source list.

*before* Iterator referencing before the start of range in list.

*last* Iterator referencing the end of range in list.

Removes elements in the range (before,last) and inserts them after *pos* in constant time.

Undefined if *pos* is in (before,last).

Definition at line 233 of file forward\_list.tcc.

**5.451.3.43** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::splice_after ( const_iterator  
__pos, forward_list<_Tp, _Alloc> && __list, const_iterator __i  
) [inline]`

Insert element from another forward\_list.

**Parameters**

*pos* Iterator referencing the element to insert after.

*list* Source list.

*i* Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 1048 of file `forward_list.h`.

**5.451.3.44** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
void std::forward_list<_Tp, _Alloc >::swap ( forward_list<_Tp,  
_Alloc > & __list ) [inline]`

Swaps data with another `forward_list`.

**Parameters**

*list* A `forward_list` of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 973 of file `forward_list.h`.

References `std::swap()`.

Referenced by `std::swap()`.

**5.451.3.45** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc >::unique ( ) [inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1116 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc >::unique()`.

Referenced by `std::forward_list<_Tp, _Alloc >::unique()`.

**5.451.3.46** `template<typename _Tp, typename _Alloc > template<typename  
_BinPred > void std::forward_list< _Tp, _Alloc >::unique (   
_BinPred __binary_pred )`

Remove consecutive elements satisfying a predicate.

#### Parameters

*binary\_pred* Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 318 of file forward\_list.tcc.

References `std::forward_list< _Tp, _Alloc >::begin()`, `std::forward_list< _Tp, _Alloc >::end()`, and `std::forward_list< _Tp, _Alloc >::erase_after()`.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.452 `std::fpos< _StateT >` Class Template Reference

Class representing stream positions.

#### Public Member Functions

- `fpos (streamoff __off)`
- `operator streamoff () const`
- `fpos operator+ (streamoff __off) const`
- `fpos & operator+= (streamoff __off)`
- `fpos operator- (streamoff __off) const`
- `streamoff operator- (const fpos &__other) const`
- `fpos & operator-= (streamoff __off)`
- `void state (_StateT __st)`
- `_StateT state () const`

### 5.452.1 Detailed Description

```
template<typename _StateT> class std::fpos<_StateT >
```

Class representing stream positions. The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

#### Parameters

*StateT* Type passed to and returned from `state()`.

Definition at line 112 of file `postypes.h`.

### 5.452.2 Constructor & Destructor Documentation

```
5.452.2.1 template<typename _StateT> std::fpos<_StateT >::fpos (
    streamoff __off ) [inline]
```

Construct position from offset.

Definition at line 133 of file `postypes.h`.

### 5.452.3 Member Function Documentation

```
5.452.3.1 template<typename _StateT> std::fpos<_StateT >::operator
    streamoff ( ) const [inline]
```

Convert to `streamoff`.

Definition at line 137 of file `postypes.h`.

```
5.452.3.2 template<typename _StateT> fpos std::fpos<_StateT >::operator+
    ( streamoff __off ) const [inline]
```

Add position and offset.

Definition at line 178 of file `postypes.h`.

```
5.452.3.3 template<typename _StateT> fpos& std::fpos<_StateT
    >::operator+=( streamoff __off ) [inline]
```

Add offset to this position.

Definition at line 154 of file postypes.h.

**5.452.3.4** `template<typename _StateT> streamoff std::fpos< _StateT >::operator-( const fpos< _StateT > & __other ) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

**5.452.3.5** `template<typename _StateT> fpos std::fpos< _StateT >::operator-( streamoff __off ) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

**5.452.3.6** `template<typename _StateT> fpos& std::fpos< _StateT >::operator=( streamoff __off ) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.

**5.452.3.7** `template<typename _StateT> _StateT std::fpos< _StateT >::state ( ) const [inline]`

Return the last set value of *st*.

Definition at line 146 of file postypes.h.

**5.452.3.8** `template<typename _StateT> void std::fpos< _StateT >::state ( _StateT __st ) [inline]`

Remember the value of *st*.

Definition at line 141 of file postypes.h.

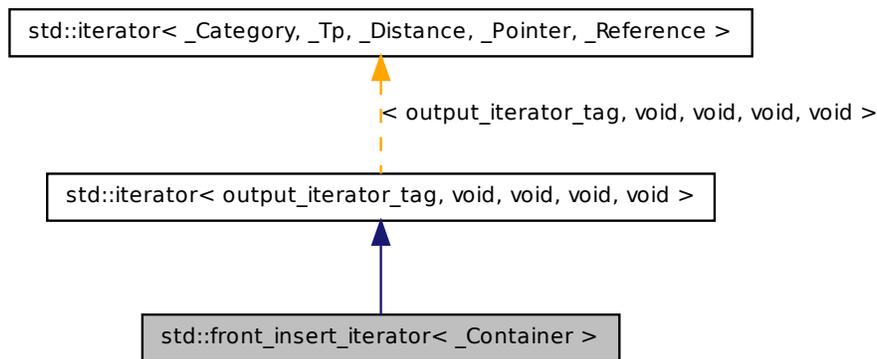
The documentation for this class was generated from the following file:

- [postypes.h](#)

## 5.453 std::front\_insert\_iterator< \_Container > Class Template Reference

Turns assignment into insertion.

Inheritance diagram for std::front\_insert\_iterator< \_Container >:



### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- `front_insert_iterator` (`_Container &__x`)
- `front_insert_iterator` & `operator*` ()
- `front_insert_iterator` & `operator++` ()
- `front_insert_iterator` `operator++` (int)
- `front_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)

- [front\\_insert\\_iterator](#) & **operator=** (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container * container`

### 5.453.1 Detailed Description

**template<typename `_Container`> class `std::front_insert_iterator`< `_Container` >**

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 485 of file `stl_iterator.h`.

### 5.453.2 Member Typedef Documentation

**5.453.2.1 `template<typename _Container > typedef _Container std::front_insert_iterator< _Container >::container_type`**

A nested typedef for the type of whatever container you used.

Definition at line 493 of file `stl_iterator.h`.

**5.453.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`**

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.453.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`**

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

## **5.453 std::front\_insert\_iterator< \_Container > Class Template Reference 2391**

**5.453.2.4** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file stl\_iterator\_base\_types.h.

**5.453.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

**5.453.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file stl\_iterator\_base\_types.h.

### **5.453.3 Constructor & Destructor Documentation**

**5.453.3.1** `template<typename _Container > std::front_insert_iterator< _Container >::front_insert_iterator ( _Container & __x ) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 496 of file stl\_iterator.h.

### **5.453.4 Member Function Documentation**

**5.453.4.1** `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* ( ) [inline]`

Simply returns \*this.

Definition at line 534 of file stl\_iterator.h.

**5.453.4.2** `template<typename _Container > front_insert_iterator  
std::front_insert_iterator< _Container >::operator++ ( int )  
[inline]`

Simply returns \*this. (This iterator does not *move*).

Definition at line 544 of file stl\_iterator.h.

**5.453.4.3** `template<typename _Container > front_insert_iterator&  
std::front_insert_iterator< _Container >::operator++ ( )  
[inline]`

Simply returns \*this. (This iterator does not *move*).

Definition at line 539 of file stl\_iterator.h.

**5.453.4.4** `template<typename _Container > front_insert_iterator&  
std::front_insert_iterator< _Container >::operator= ( const  
typename _Container::value_type & __value ) [inline]`

#### Parameters

*value* An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 518 of file stl\_iterator.h.

The documentation for this class was generated from the following file:

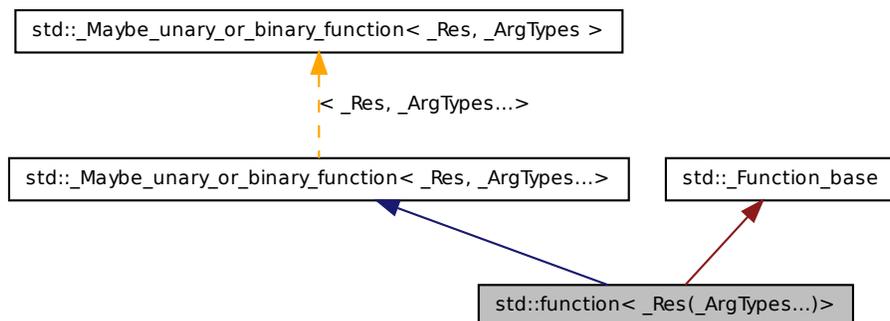
- [stl\\_iterator.h](#)

## 5.454 `std::function< _Res(_ArgTypes...)>` Class Template Reference

Primary class template for `std::function`.

Polymorphic function wrapper.

Inheritance diagram for std::function< \_Res(\_ArgTypes...)>:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- [function](#) ()
- [function](#) (\_M\_clear\_type \*)
- [function](#) (function &&\_\_x)
- [template](#)<typename \_Function >  
[function](#) (\_Function \_\_f, typename [enable\\_if](#)< !is\_integral< \_Function >::value, \_Useless >::type=\_Useless())
- [function](#) (const function &\_\_x)
- [operator bool](#) () const
- [template](#)<typename \_Res2 , typename... \_ArgTypes2>  
void **operator!=** (const function< \_Res2(\_ArgTypes2...)> &) const
- `_Res` [operator\(\)](#) (\_ArgTypes...\_\_args) const
- [function](#) & [operator=](#) (\_M\_clear\_type \*)
- [template](#)<typename \_Function >  
[enable\\_if](#)<!is\_integral< \_Function >::value, function & >::type [operator=](#) (\_Function &&\_\_f)
- [template](#)<typename \_Function >  
[enable\\_if](#)<!is\_integral< \_Function >::value, function & >::type [operator=](#) ([reference\\_wrapper](#)< \_Function > \_\_f)

- function & [operator=](#) (const function &\_\_x)
- function & [operator=](#) (function &&\_\_x)
- template<typename \_Res2, typename... \_ArgTypes2>  
void [operator==](#) (const function< \_Res2(\_ArgTypes2...)> &) const
- void [swap](#) (function &\_\_x)
- template<typename \_Functor >  
\_Functor \* [target](#) ()
- template<typename \_Functor >  
const \_Functor \* [target](#) () const
- const [type\\_info](#) & [target\\_type](#) () const

### Private Types

- typedef bool(\* [\\_Manager\\_type](#) )(\_Any\_data &, const \_Any\_data &, \_-  
Manager\_operation)

### Private Member Functions

- bool [\\_M\\_empty](#) () const

### Private Attributes

- [\\_Any\\_data](#) [\\_M\\_functor](#)
- [\\_Manager\\_type](#) [\\_M\\_manager](#)

### Static Private Attributes

- static const std::size\_t [\\_M\\_max\\_align](#)
- static const std::size\_t [\\_M\\_max\\_size](#)

#### 5.454.1 Detailed Description

template<typename \_Res, typename... \_ArgTypes> class std::function< \_Res(\_-  
ArgTypes...)>

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 1812 of file functional.

## 5.454.2 Constructor & Destructor Documentation

**5.454.2.1** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( ) [inline, explicit]`

Default construct creates an empty function call wrapper.

### Postcondition

!(bool)\*this

Definition at line 1830 of file functional.

**5.454.2.2** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( _M_clear_type * ) [inline]`

Default construct creates an empty function call wrapper.

### Postcondition

!(bool)\*this

Definition at line 1836 of file functional.

**5.454.2.3** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( const function< _Res(_ArgTypes...)> & __x )`

Function copy constructor.

### Parameters

*x* A function object with identical call signature.

### Postcondition

(bool)\*this == (bool)*x*

The newly-created function contains a copy of the target of *x* (if it has one).

Definition at line 2080 of file functional.

**5.454.2.4** `template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( function< _Res(_ArgTypes...)> && __x ) [inline]`

Function move constructor.

**Parameters**

*x* A function object rvalue with identical call signature.

The newly-created function contains the target of *x* (if it has one).

Definition at line 1855 of file functional.

```
5.454.2.5 template<typename _Res , typename... _ArgTypes>
template<typename _Function > std::function<
_Res(_ArgTypes...)>::function ( _Function _f, typename enable_if<
!is_integral< _Function >::value, _Useless >::type = _Useless () )
```

Builds a function that targets a copy of the incoming function object.

**Parameters**

*f* A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to *Res*.

The newly-created function object will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2094 of file functional.

**5.454.3 Member Function Documentation**

```
5.454.3.1 template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::operator bool ( ) const [inline,
explicit]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 2023 of file functional.

```
5.454.3.2 template<typename _Res , typename... _ArgTypes> _Res
std::function< _Res(_ArgTypes...)>::operator() ( _ArgTypes...
__args ) const
```

Invokes the function targeted by `*this`.

#### Returns

the result of the target.

#### Exceptions

*bad\_function\_call* when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.

Definition at line 2112 of file functional.

```
5.454.3.3 template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= ( function<
_Res(_ArgTypes...)> && __x ) [inline]
```

Function move-assignment operator.

#### Parameters

`x` A function rvalue with identical call signature.

#### Returns

`*this`

The target of `x` is moved to `*this`. If `x` has no target, then `*this` will be empty.

If `x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1915 of file functional.

```
5.454.3.4 template<typename _Res , typename... _ArgTypes>
template<typename _Functor > enable_if<!is_integral<_
Functor>::value, function&>::type std::function<
_Res(_ArgTypes...)>::operator= ( _Functor && __f ) [inline]
```

Function assignment to a new target.

**Parameters**

*f* A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res.

**Returns**

\*this

This function object wrapper will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 1958 of file functional.

```
5.454.3.5 template<typename _Res , typename... _ArgTypes>
template<typename _Functor > enable_if<!is_integral<_
Functor>::value, function&>::type std::function<
_Res(_ArgTypes...)>::operator= ( reference_wrapper< _Functor >
_f ) [inline]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 1967 of file functional.

```
5.454.3.6 template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= ( const function<
_Res(_ArgTypes...)> & _x ) [inline]
```

Function assignment operator.

**Parameters**

*x* A function with identical call signature.

**Postcondition**

`(bool)*this == (bool)x`

**Returns**

\*this

The target of *x* is copied to *\*this*. If *x* has no target, then *\*this* will be empty.

If *x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1897 of file functional.

```
5.454.3.7 template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= ( _M_clear_type * )
[inline]
```

Function assignment to zero.

#### Postcondition

!(bool)\*this

#### Returns

\*this

The target of *\*this* is deallocated, leaving it empty.

Definition at line 1929 of file functional.

```
5.454.3.8 template<typename _Res , typename... _ArgTypes> void
std::function< _Res(_ArgTypes...)>::swap ( function<
_Res(_ArgTypes...)> & __x ) [inline]
```

Swap the targets of two function objects.

#### Parameters

*f* A function with identical call signature.

Swap the targets of *this* function object and *f*. This function will not throw an exception.

Definition at line 1982 of file functional.

References std::swap().

```
5.454.3.9 template<typename _Res , typename... _ArgTypes>
template<typename _Functor > const _Functor * std::function<
_Res(_ArgTypes...)>::target ( ) const
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2158 of file functional.

**5.454.3.10** `template<typename _Res , typename... _ArgTypes>  
template<typename _Functor > _Functor * std::function<  
_Res(_ArgTypes...)>::target ( )`

Access the stored target function object.

#### Returns

Returns a pointer to the stored target function object, if `typeid(Functor).equals(target_type())`; otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2139 of file functional.

**5.454.3.11** `template<typename _Res , typename... _ArgTypes> const type_info  
& std::function< _Res(_ArgTypes...)>::target_type ( ) const`

Determine the type of the target of this function object wrapper.

#### Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

Definition at line 2123 of file functional.

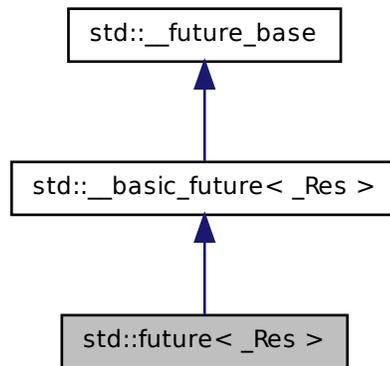
The documentation for this class was generated from the following file:

- [functional](#)

## 5.455 `std::future< _Res >` Class Template Reference

Primary template for future.

Inheritance diagram for `std::future<_Res>`:



## Public Member Functions

- `future` (`future` &&\_uf)
- `future` (`const future` &)
- `_Res` `get` ()
- `future` & `operator=` (`const future` &)
- `future` & `operator=` (`future` &&\_fut)
- `bool valid` () `const`
- `void wait` () `const`
- `template<typename _Rep, typename _Period >`  
`bool wait_for` (`const chrono::duration<_Rep, _Period > &__rel`) `const`
- `template<typename _Clock, typename _Duration >`  
`bool wait_until` (`const chrono::time_point<_Clock, _Duration > &__abs`)  
`const`

## Protected Types

- `typedef __future_base::_Result<_Res> & __result_type`

## Protected Member Functions

- [\\_\\_result\\_type \\_M\\_get\\_result\(\)](#)
- `void _M_swap (__basic_future &__that)`

## Friends

- `template<typename _Fn, typename... _Args>  
future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&  
_Args &&...)`
- class `packaged_task`
- class `promise< _Res >`

### 5.455.1 Detailed Description

`template<typename _Res> class std::future< _Res >`

Primary template for future.

Definition at line 559 of file future.

### 5.455.2 Constructor & Destructor Documentation

**5.455.2.1** `template<typename _Res > std::future< _Res >::future ( future<  
_Res > && _uf ) [inline]`

Move constructor.

Definition at line 577 of file future.

### 5.455.3 Member Function Documentation

**5.455.3.1** `template<typename _Res> __result_type std::__basic_future< _Res  
>::__M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

### 5.455.3.2 `template<typename _Res > _Res std::future<_Res >::get ( )` [inline]

Retrieving the value.

Definition at line 591 of file future.

References `std::_basic_future<_Res >::_M_get_result()`.

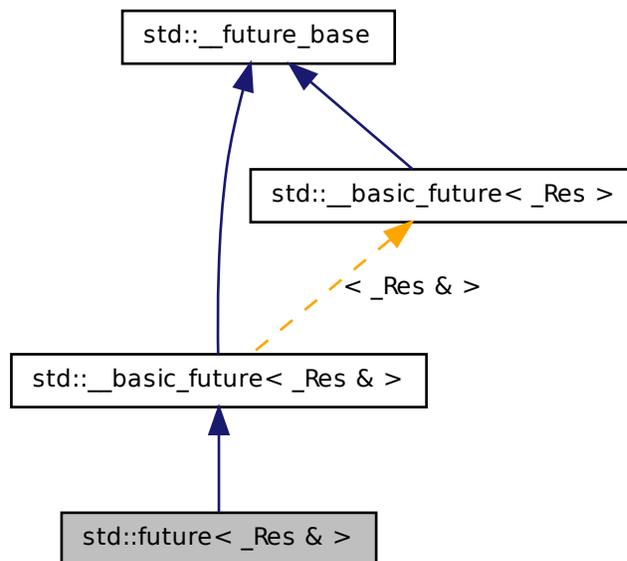
The documentation for this class was generated from the following file:

- [future](#)

## 5.456 `std::future<_Res &>` Class Template Reference

Partial specialization for `future<R&>`

Inheritance diagram for `std::future<_Res &>`:



## Public Member Functions

- `future` (`future` &&\_uf)
- `future` (const `future` &)
- `_Res` & `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&\_fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_rel) const
- bool `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_abs) const

## Protected Types

- typedef `__future_base::Result`< \_Res & > & `__result_type`

## Protected Member Functions

- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` &\_that)

## Friends

- template<typename \_Fn , typename... \_Args>  
`future`< typename result\_of< \_Fn(\_Args...)>::type > `async` (launch, \_Fn &&, \_Args &&...)
- class `packaged_task`
- class `promise`< `_Res` & >

### 5.456.1 Detailed Description

`template<typename _Res> class std::future< _Res & >`

Partial specialization for `future<R&>`

Definition at line 600 of file `future`.

## 5.456.2 Constructor & Destructor Documentation

**5.456.2.1** `template<typename _Res > std::future< _Res & >::future ( future< _Res & > && __uf ) [inline]`

Move constructor.

Definition at line 618 of file future.

## 5.456.3 Member Function Documentation

**5.456.3.1** `__result_type std::__basic_future< _Res & >::_M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

References `std::rethrow_exception()`.

**5.456.3.2** `template<typename _Res > _Res& std::future< _Res & >::get ( ) [inline]`

Retrieving the value.

Definition at line 632 of file future.

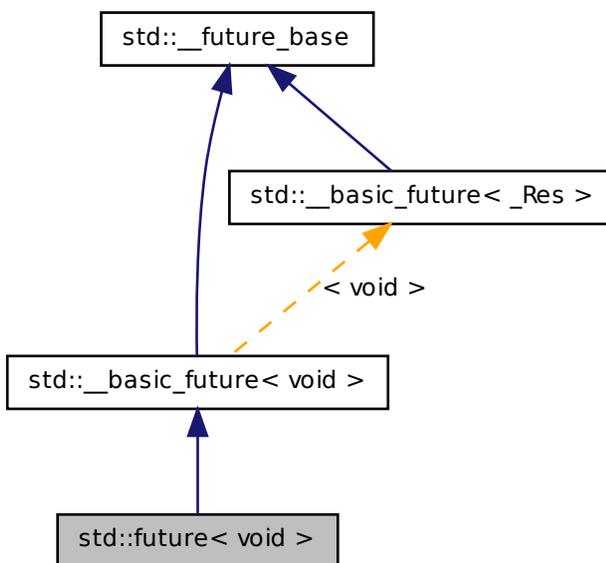
The documentation for this class was generated from the following file:

- [future](#)

## 5.457 `std::future< void >` Class Template Reference

Explicit specialization for [future<void>](#)

Inheritance diagram for `std::future< void >`:



## Public Member Functions

- `future` (`future` &&\_uf)
- `future` (const `future` &)
- void `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&\_fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_rel) const
- bool `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_abs) const

## Protected Types

- typedef `__future_base::_Result`< void > & `__result_type`

## Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap (__basic_future &__that)`

## Friends

- `template<typename _Fn, typename... _Args>  
future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&  
_Args &&...)`
- class `packaged_task`
- class `promise< void >`

### 5.457.1 Detailed Description

`template<> class std::future< void >`

Explicit specialization for `future<void>`

Definition at line 641 of file future.

### 5.457.2 Constructor & Destructor Documentation

**5.457.2.1** `std::future< void >::future ( future< void > && __uf )`  
[inline]

Move constructor.

Definition at line 659 of file future.

### 5.457.3 Member Function Documentation

**5.457.3.1** `__result_type std::__basic_future< void >::_M_get_result ( )`  
[inline, protected, inherited]

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

**5.457.3.2** `void std::future< void >::get ( )` [inline]

Retrieving the value.

Definition at line 673 of file future.

References `std::__basic_future<_Res>::_M_get_result()`.

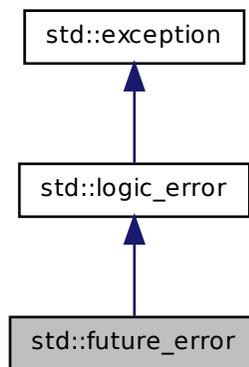
The documentation for this class was generated from the following file:

- [future](#)

## 5.458 `std::future_error` Class Reference

Exception type thrown by futures.

Inheritance diagram for `std::future_error`:



### Public Member Functions

- `future_error` (`error_code` `__ec`)
- `const error_code & code` () const throw ()
- `virtual const char * what` () const throw ()

#### 5.458.1 Detailed Description

Exception type thrown by futures.

Definition at line 85 of file future.

## 5.458.2 Member Function Documentation

### 5.458.2.1 `virtual const char* std::future_error::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

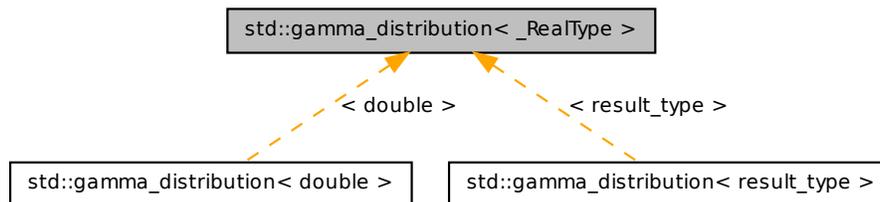
The documentation for this class was generated from the following file:

- [future](#)

## 5.459 `std::gamma_distribution<_RealType >` Class Template Reference

A gamma continuous distribution for random numbers.

Inheritance diagram for `std::gamma_distribution<_RealType >`:



### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- [gamma\\_distribution](#) (`const param_type &__p`)
- `_RealType` [alpha](#) () const
- `_RealType` [beta](#) () const
- `result_type` [max](#) () const
- `result_type` [min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type` [operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type` [operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- `void` [param](#) (`const param_type &__param`)
- `param_type` [param](#) () const
- `void` [reset](#) ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > &` [operator<<](#) (`std::basic_ostream< _CharT, _Traits > &, const std::gamma_distribution< _RealType1 > &`)
- `template<typename _RealType1 >`  
`bool` [operator==](#) (`const std::gamma_distribution< _RealType1 > &__d1, const std::gamma_distribution< _RealType1 > &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > &` [operator>>](#) (`std::basic_istream< _CharT, _Traits > &, std::gamma_distribution< _RealType1 > &`)

### 5.459.1 Detailed Description

`template<typename _RealType = double> class std::gamma_distribution< _RealType >`

A gamma continuous distribution for random numbers. The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)}(x/\beta)^{\alpha-1}e^{-x/\beta}$$

Definition at line 2350 of file random.h.

## 5.459.2 Member Typedef Documentation

**5.459.2.1** `template<typename _RealType = double> typedef _RealType  
std::gamma_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2357 of file `random.h`.

## 5.459.3 Constructor & Destructor Documentation

**5.459.3.1** `template<typename _RealType = double> std::gamma_distribution<  
_RealType>::gamma_distribution ( _RealType __alpha_val  
= _RealType (1), _RealType __beta_val = _RealType (1) )  
[inline, explicit]`

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2402 of file `random.h`.

## 5.459.4 Member Function Documentation

**5.459.4.1** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType>::alpha ( ) const  
[inline]`

Returns the  $\alpha$  of the distribution.

Definition at line 2423 of file `random.h`.

**5.459.4.2** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType>::beta ( ) const [inline]`

Returns the  $\beta$  of the distribution.

Definition at line 2430 of file `random.h`.

**5.459.4.3** `template<typename _RealType = double> result_type  
std::gamma_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2459 of file `random.h`.

Referenced by `std::gamma_distribution<result_type>::max()`.

**5.459.4.4** `template<typename _RealType = double> result_type  
std::gamma_distribution<_RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2452 of file random.h.

**5.459.4.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::gamma_distribution<_RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2467 of file random.h.

Referenced by `std::gamma_distribution< result_type >::operator()`.

**5.459.4.6** `template<typename _RealType > template<typename  
_UniformRandomNumberGenerator > gamma_distribution<  
_RealType >::result_type std::gamma_distribution<_RealType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables"  
ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 1975 of file random.tcc.

References `std::log()`, and `std::pow()`.

**5.459.4.7** `template<typename _RealType = double> void  
std::gamma_distribution<_RealType >::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2445 of file random.h.

**5.459.4.8** `template<typename _RealType = double> param_type  
std::gamma_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2437 of file `random.h`.

Referenced by `std::gamma_distribution<result_type>::operator()`.

**5.459.4.9** `template<typename _RealType = double> void  
std::gamma_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2416 of file `random.h`.

Referenced by `std::negative_binomial_distribution<_IntType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::fisher_f_distribution<_RealType>::reset()`, and `std::chi_squared_distribution<_RealType>::reset()`.

## 5.459.5 Friends And Related Function Documentation

**5.459.5.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::gamma_distribution<_RealType1> & ) [friend]`

Inserts a `gamma_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `gamma_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.459.5.2** `template<typename _RealType = double> template<typename _RealType1 > bool operator==( const std::gamma_distribution< _RealType1 > & _d1, const std::gamma_distribution< _RealType1 > & _d2 ) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2482 of file random.h.

**5.459.5.3** `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & , std::gamma_distribution< _RealType1 > & ) [friend]`

Extracts a gamma\_distribution random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A gamma\_distribution random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.460 `std::gamma_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef `gamma_distribution< _RealType > distribution_type`

### Public Member Functions

- `param_type ( _RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1))`

## 5.461 `std::geometric_distribution<_IntType>` Class Template Reference 2415

- `_RealType alpha () const`
- `_RealType beta () const`

### Friends

- class `gamma_distribution<_RealType>`
- bool `operator==(const param_type &__p1, const param_type &__p2)`

### 5.460.1 Detailed Description

`template<typename _RealType = double> struct std::gamma_distribution<_RealType>::param_type`

Parameter type.

Definition at line 2359 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.461 `std::geometric_distribution<_IntType>` Class Template Reference

A discrete geometric random number distribution.

### Classes

- struct `param_type`

### Public Types

- typedef `_IntType result_type`

### Public Member Functions

- `geometric_distribution (double __p=0.5)`
- `geometric_distribution (const param_type &__p)`
- `result_type max () const`

- [result\\_type min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng, const`  
`param\_type &__p)`
- `double p` () const
- `void param` (const `param\_type` &\_\_param)
- `param\_type param` () const
- `void reset` ()

### 5.461.1 Detailed Description

`template<typename _IntType = int> class std::geometric_distribution< _-`  
`IntType >`

A discrete geometric random number distribution. The formula for the geometric probability density function is  $p(i|p) = (1 - p)p^{i-1}$  where  $p$  is the parameter of the distribution.

Definition at line 3619 of file random.h.

### 5.461.2 Member Typedef Documentation

**5.461.2.1** `template<typename _IntType = int> typedef _IntType`  
`std::geometric_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3626 of file random.h.

### 5.461.3 Member Function Documentation

**5.461.3.1** `template<typename _IntType = int> result_type`  
`std::geometric_distribution< _IntType >::max ( ) const`  
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3712 of file random.h.

**5.461.3.2** `template<typename _IntType = int> result_type  
std::geometric_distribution<_IntType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3705 of file `random.h`.

**5.461.3.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::geometric_distribution<_IntType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3720 of file `random.h`.

References `std::geometric_distribution<_IntType>::operator()`, and  
`std::geometric_distribution<_IntType>::param()`.

Referenced by `std::geometric_distribution<_IntType>::operator()`.

**5.461.3.4** `template<typename _IntType = int> double  
std::geometric_distribution<_IntType>::p ( ) const [inline]`

Returns the distribution parameter `p`.

Definition at line 3683 of file `random.h`.

**5.461.3.5** `template<typename _IntType = int> void std::geometric_  
distribution<_IntType>::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3698 of file `random.h`.

**5.461.3.6** `template<typename _IntType = int> param_type  
std::geometric_distribution<_IntType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3690 of file random.h.

Referenced by `std::geometric_distribution< _IntType >::operator()()`, `std::operator==( )`, and `std::operator>>()`.

**5.461.3.7** `template<typename _IntType = int> void  
std::geometric_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 3677 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.462 `std::geometric_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef `geometric_distribution< _IntType >` `distribution_type`

### Public Member Functions

- `param_type` (double \_\_p=0.5)
- double `p` () const

### Friends

- class `geometric_distribution< _IntType >`
- bool `operator==(const param_type &__p1, const param_type &__p2)`

### 5.462.1 Detailed Description

`template<typename _IntType = int> struct std::geometric_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3628 of file random.h.

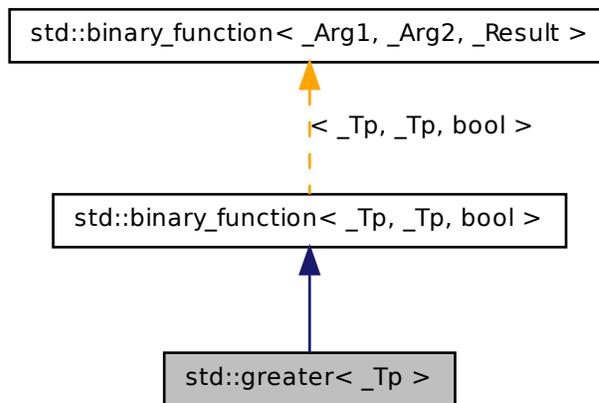
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.463 std::greater< \_Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.463.1 Detailed Description

`template<typename _Tp> struct std::greater< _Tp >`

One of the [comparison functors](#).

Definition at line 217 of file `stl_function.h`.

### 5.463.2 Member Typedef Documentation

**5.463.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.463.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.463.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

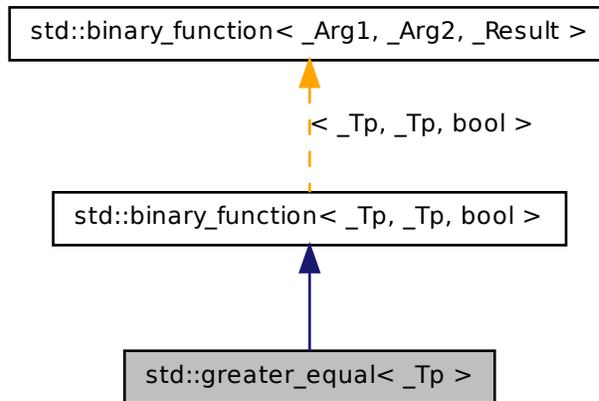
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.464 `std::greater_equal< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater\_equal< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.464.1 Detailed Description

```
template<typename _Tp> struct std::greater_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 235 of file `stl_function.h`.

## 5.464.2 Member Typedef Documentation

### 5.464.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

### 5.464.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

### 5.464.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.465 `std::gslice` Class Reference

Class defining multi-dimensional subset of an array.

### Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t, const [valarray](#)< size\_t > &, const [valarray](#)< size\_t > &)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

## Friends

- class `valarray`

### 5.465.1 Detailed Description

Class defining multi-dimensional subset of an array. The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 63 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 5.466 `std::gslice_array< _Tp >` Class Template Reference

Reference to multi-dimensional subset of an array.

### Public Types

- typedef `_Tp value_type`

### Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- `template<class _Dom >`  
void **operator%=** (const `_Expr< _Dom, _Tp >` &) const
- void **operator%=** (const [valarray](#)< `_Tp` > &) const
- void **operator&=** (const [valarray](#)< `_Tp` > &) const
- `template<class _Dom >`  
void **operator&=** (const `_Expr< _Dom, _Tp >` &) const

- `template<class _Dom >`  
`void operator*= (const _Expr< _Dom, _Tp > &) const`
- `void operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const _Tp &) const`
- `gslice_array & operator= (const gslice_array &)`
- `template<class _Dom >`  
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator|= (const _Expr< _Dom, _Tp > &) const`

## Friends

- `class valarray< _Tp >`

### 5.466.1 Detailed Description

`template<typename _Tp> class std::gslice_array< _Tp >`

Reference to multi-dimensional subset of an array. A `gslice_array` is a reference to the actual elements of an array specified by a `gslice`. The way to get a `gslice_array` is to call `operator[]`(`gslice`) on a `valarray`. The returned `gslice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `gslice_array` refers to.

**Parameters**

*Tp* Element type.

Definition at line 59 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice\\_array.h](#)

## 5.467 `std::has_nothrow_assign< _Tp >` Struct Template Reference

[has\\_nothrow\\_assign](#)

Inherits `integral_constant< bool, __has_nothrow_assign(_Tp)>`.

### 5.467.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_assign< _Tp >
```

[has\\_nothrow\\_assign](#)

Definition at line 275 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.468 `std::has_nothrow_copy_constructor< _Tp >` Struct Template Reference

[has\\_nothrow\\_copy\\_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_copy(_Tp)>`.

### 5.468.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_constructor< _Tp >
```

[has\\_nothrow\\_copy\\_constructor](#)

Definition at line 269 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.469 `std::has_nothrow_default_constructor`< `_Tp` > Struct Template Reference

[has\\_nothrow\\_default\\_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_constructor(_Tp)>`.

### 5.469.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_default_constructor< _Tp >`

[has\\_nothrow\\_default\\_constructor](#)

Definition at line 263 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.470 `std::has_trivial_assign`< `_Tp` > Struct Template Reference

[has\\_trivial\\_assign](#)

Inherits `integral_constant< bool, __has_trivial_assign(_Tp)>`.

### 5.470.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_assign< _Tp >`

[has\\_trivial\\_assign](#)

Definition at line 251 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## **5.471 `std::has_trivial_copy_constructor`< `_Tp` > Struct Template Reference 2427**

### **5.471 `std::has_trivial_copy_constructor`< `_Tp` > Struct Template Reference**

[has\\_trivial\\_copy\\_constructor](#)

Inherits `integral_constant< bool, __has_trivial_copy(_Tp)>`.

#### **5.471.1 Detailed Description**

```
template<typename _Tp> struct std::has_trivial_copy_constructor< _Tp >
```

[has\\_trivial\\_copy\\_constructor](#)

Definition at line 245 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### **5.472 `std::has_trivial_default_constructor`< `_Tp` > Struct Template Reference**

[has\\_trivial\\_default\\_constructor](#)

Inherits `integral_constant< bool, __has_trivial_constructor(_Tp)>`.

#### **5.472.1 Detailed Description**

```
template<typename _Tp> struct std::has_trivial_default_constructor< _Tp >
```

[has\\_trivial\\_default\\_constructor](#)

Definition at line 239 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### **5.473 `std::has_trivial_destructor`< `_Tp` > Struct Template Reference**

[has\\_trivial\\_destructor](#)

Inherits `integral_constant< bool, __has_trivial_destructor(_Tp)>`.

### 5.473.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_destructor< _Tp >`

[has\\_trivial\\_destructor](#)

Definition at line 257 of file `type_traits`.

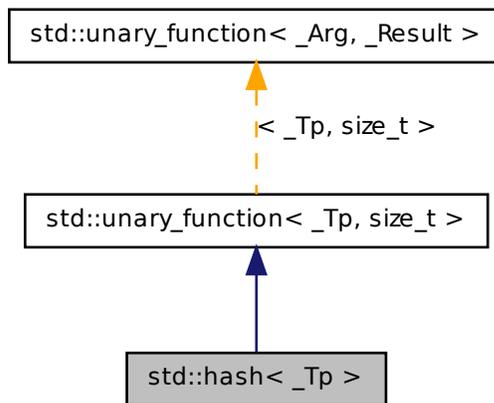
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.474 `std::hash< _Tp >` Struct Template Reference

Primary class template hash.

Inheritance diagram for `std::hash< _Tp >`:



### Public Types

- `typedef _Tp` [argument\\_type](#)

- typedef size\_t [result\\_type](#)

## Public Member Functions

- size\_t **operator()** (\_Tp \_\_val) const
- template<>  
size\_t **operator()** (double \_\_val) const
- template<>  
size\_t **operator()** (float \_\_val) const
- template<>  
size\_t **operator()** (unsigned long long \_\_val) const
- template<>  
size\_t **operator()** (unsigned long \_\_val) const
- template<>  
size\_t **operator()** (unsigned int \_\_val) const
- template<>  
size\_t **operator()** (unsigned short \_\_val) const
- template<>  
size\_t **operator()** (long long \_\_val) const
- template<>  
size\_t **operator()** (long \_\_val) const
- template<>  
size\_t **operator()** (int \_\_val) const
- template<>  
size\_t **operator()** (short \_\_val) const
- template<>  
size\_t **operator()** (char32\_t \_\_val) const
- template<>  
size\_t **operator()** (char16\_t \_\_val) const
- template<>  
size\_t **operator()** (wchar\_t \_\_val) const
- template<>  
size\_t **operator()** (unsigned char \_\_val) const
- template<>  
size\_t **operator()** (signed char \_\_val) const
- template<>  
size\_t **operator()** (char \_\_val) const
- template<>  
size\_t **operator()** (bool \_\_val) const

### 5.474.1 Detailed Description

**template<typename \_Tp> struct std::hash< \_Tp >**

Primary class template hash.

Definition at line 50 of file functional\_hash.h.

### 5.474.2 Member Typedef Documentation

**5.474.2.1 typedef \_Tp std::unary\_function< \_Tp , size\_t >::argument\_type  
[inherited]**

argument\_type is the type of the /// argument (no surprises here)

Definition at line 102 of file stl\_function.h.

**5.474.2.2 typedef size\_t std::unary\_function< \_Tp , size\_t >::result\_type  
[inherited]**

result\_type is the return type

Definition at line 105 of file stl\_function.h.

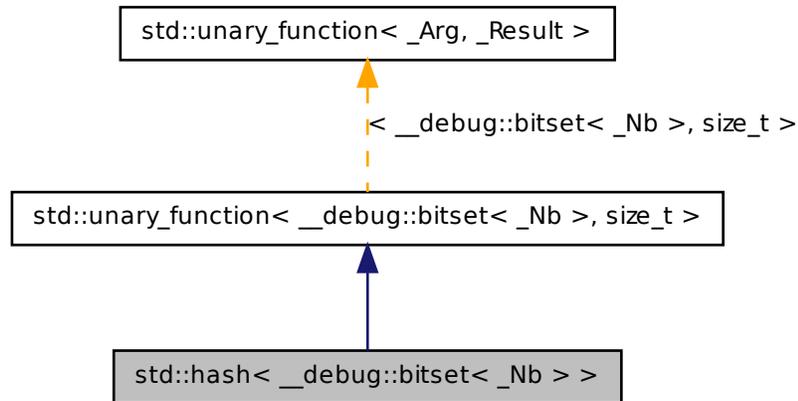
The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.475 std::hash< \_\_debug::bitset< \_Nb > > Struct Template Reference

[std::hash](#) specialization for [bitset](#).

Inheritance diagram for `std::hash< __debug::bitset< _Nb > >`:



## Public Types

- typedef `__debug::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (`const __debug::bitset< _Nb > &__b`) `const`

### 5.475.1 Detailed Description

`template<size_t _Nb> struct std::hash< __debug::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

Definition at line 387 of file `debug/bitset`.

## 5.475.2 Member Typedef Documentation

**5.475.2.1** `typedef __debug::bitset< _Nb > std::unary_function<  
__debug::bitset< _Nb >, size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.475.2.2** `typedef size_t std::unary_function< __debug::bitset< _Nb >, size_t  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

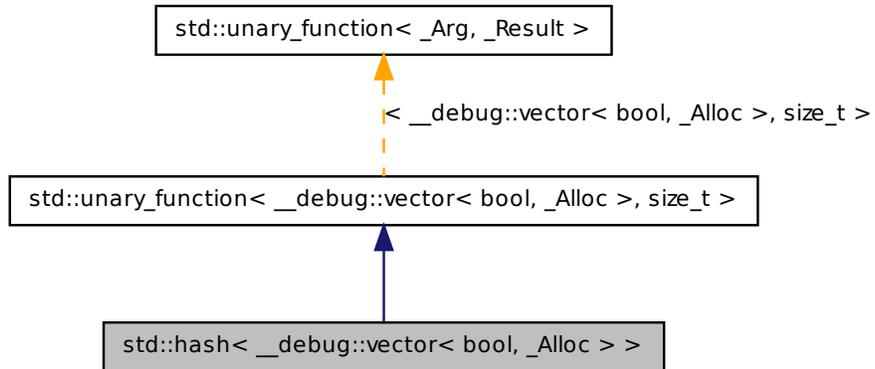
- [debug/bitset](#)

## 5.476 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

## 5.476 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

Inheritance diagram for `std::hash< __debug::vector< bool, _Alloc > >`:



### Public Types

- typedef `__debug::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (`const __debug::vector< bool, _Alloc > &__b`) `const`

#### 5.476.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __debug::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 545 of file `debug/vector`.

## 5.476.2 Member Typedef Documentation

**5.476.2.1** `typedef __debug::vector< bool, _Alloc > std::unary_function< __debug::vector< bool, _Alloc > , size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.476.2.2** `typedef size_t std::unary_function< __debug::vector< bool, _Alloc > , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

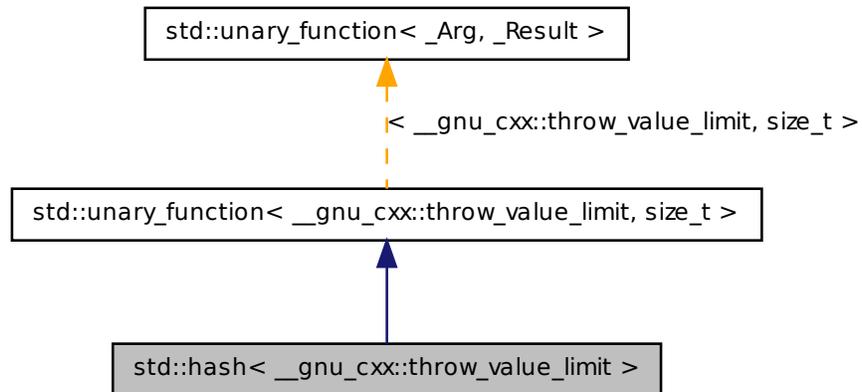
- [debug/vector](#)

## 5.477 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

## 5.477 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference 2435

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



### Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_limit &__val) const`

#### 5.477.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::throw_value_limit >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 728 of file `throw_allocator.h`.

## 5.477.2 Member Typedef Documentation

**5.477.2.1** `typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.477.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

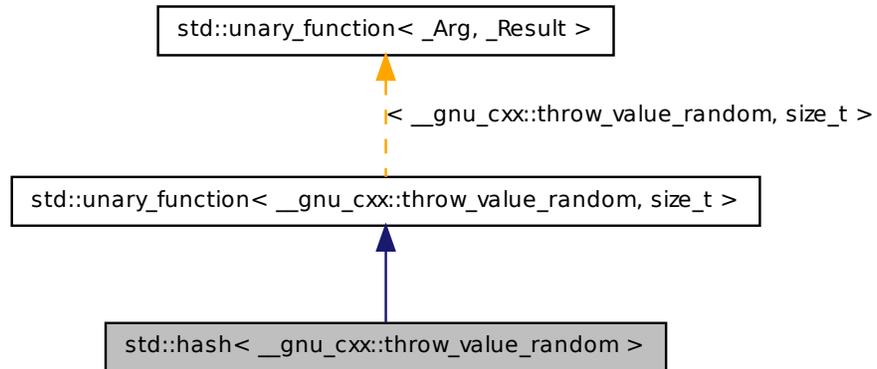
- [throw\\_allocator.h](#)

## 5.478 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

## 5.478 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



### Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`__val`) const

#### 5.478.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::throw_value_random >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 742 of file `throw_allocator.h`.

## 5.478.2 Member Typedef Documentation

**5.478.2.1** `typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.478.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

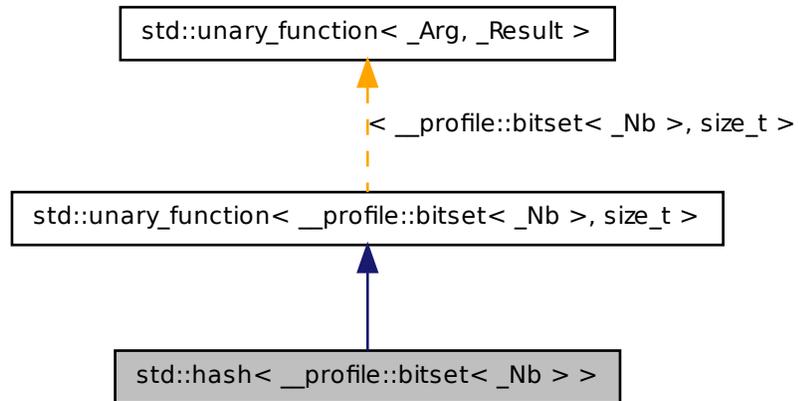
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.479 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inheritance diagram for `std::hash< __profile::bitset< _Nb > >`:



## Public Types

- typedef `__profile::bitset< _Nb >` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator() (const __profile::bitset< _Nb > &__b) const`

### 5.479.1 Detailed Description

`template<size_t _Nb> struct std::hash< __profile::bitset< _Nb > >`

`std::hash` specialization for `bitset`.

Definition at line 361 of file `profile/bitset`.

## 5.479.2 Member Typedef Documentation

**5.479.2.1** `typedef __profile::bitset< _Nb > std::unary_function<  
__profile::bitset< _Nb > , size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.479.2.2** `typedef size_t std::unary_function< __profile::bitset< _Nb > , size_t  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

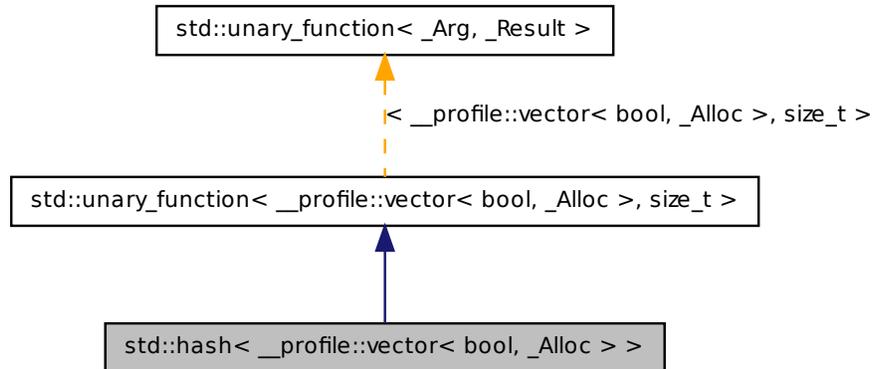
- [profile/bitset](#)

## 5.480 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

## 5.480 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

Inheritance diagram for `std::hash< __profile::vector< bool, _Alloc > >`:



### Public Types

- typedef `__profile::vector< bool, _Alloc >` [argument\\_type](#)
- typedef `size_t` [result\\_type](#)

### Public Member Functions

- `size_t operator() (const __profile::vector< bool, _Alloc > &__b) const`

#### 5.480.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __profile::vector< bool, _Alloc > >
```

[std::hash](#) specialization for `vector<bool>`.

Definition at line 469 of file `profile/vector`.

## 5.480.2 Member Typedef Documentation

**5.480.2.1** `typedef __profile::vector< bool, _Alloc > std::unary_function< __profile::vector< bool, _Alloc >, size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.480.2.2** `typedef size_t std::unary_function< __profile::vector< bool, _Alloc >, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

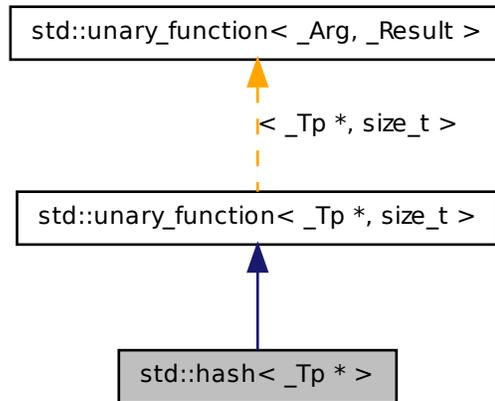
The documentation for this struct was generated from the following file:

- [profile/vector](#)

## 5.481 `std::hash< _Tp * >` Struct Template Reference

Partial specializations for pointer types.

Inheritance diagram for std::hash< \_Tp \* >:



## Public Types

- typedef `_Tp *` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator() (_Tp * __p) const`

### 5.481.1 Detailed Description

```
template<typename _Tp> struct std::hash< _Tp * >
```

Partial specializations for pointer types.

Definition at line 58 of file `functional_hash.h`.

## 5.481.2 Member Typedef Documentation

### 5.481.2.1 `typedef _Tp * std::unary_function< _Tp * , size_t >::argument_type` [`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.481.2.2 `typedef size_t std::unary_function< _Tp * , size_t >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

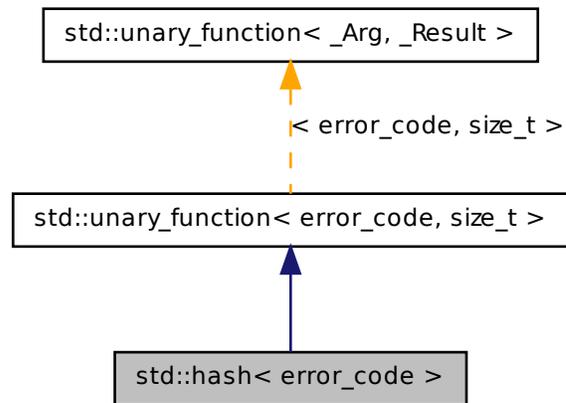
The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.482 `std::hash< error_code >` Struct Template Reference

[std::hash](#) specialization for `error_code`.

Inheritance diagram for `std::hash< error_code >`:



## Public Types

- typedef `error_code` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (const `error_code` &\_\_e) const

### 5.482.1 Detailed Description

`template<> struct std::hash< error_code >`

`std::hash` specialization for `error_code`.

Definition at line 352 of file `system_error`.

## 5.482.2 Member Typedef Documentation

### 5.482.2.1 `typedef error_code std::unary_function< error_code , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.482.2.2 `typedef size_t std::unary_function< error_code , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

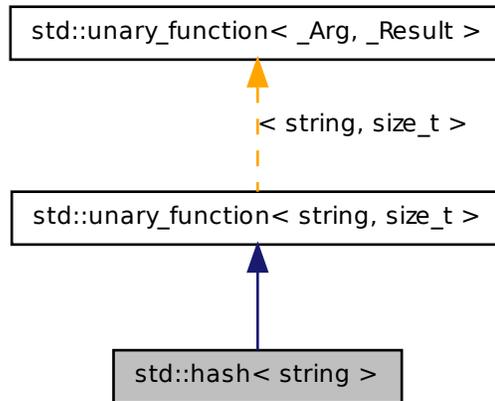
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.483 `std::hash< string >` Struct Template Reference

[std::hash](#) specialization for string.

Inheritance diagram for std::hash< string >:



## Public Types

- typedef `string` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator() (const string &__s) const`

### 5.483.1 Detailed Description

`template<> struct std::hash< string >`

`std::hash` specialization for `string`.

Definition at line 2886 of file `basic_string.h`.

## 5.483.2 Member Typedef Documentation

### 5.483.2.1 `typedef string std::unary_function< string , size_t >::argument_type` [`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.483.2.2 `typedef size_t std::unary_function< string , size_t >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

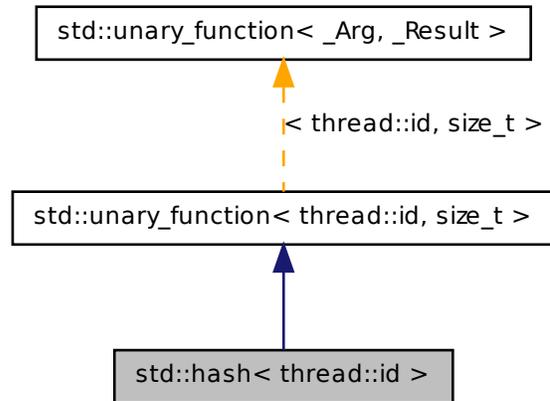
The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.484 `std::hash< thread::id >` Struct Template Reference

[std::hash](#) specialization for [thread::id](#).

Inheritance diagram for `std::hash< thread::id >`:



## Public Types

- typedef `thread::id` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (const `thread::id` &\_\_id) const

### 5.484.1 Detailed Description

`template<> struct std::hash< thread::id >`

`std::hash` specialization for `thread::id`.

Definition at line 226 of file `thread`.

## 5.484.2 Member Typedef Documentation

### 5.484.2.1 `typedef thread::id std::unary_function< thread::id , size_t >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.484.2.2 `typedef size_t std::unary_function< thread::id , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

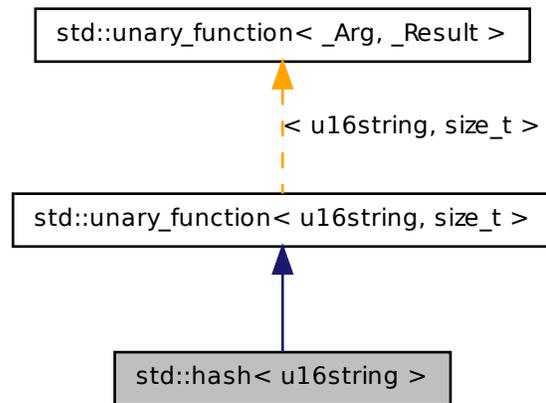
The documentation for this struct was generated from the following file:

- [thread](#)

## 5.485 `std::hash< u16string >` Struct Template Reference

[std::hash](#) specialization for `u16string`.

Inheritance diagram for std::hash< u16string >:



## Public Types

- typedef `u16string` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()(const u16string &__s) const`

### 5.485.1 Detailed Description

`template<> struct std::hash< u16string >`

`std::hash` specialization for `u16string`.

Definition at line 2911 of file `basic_string.h`.

## 5.485.2 Member Typedef Documentation

### 5.485.2.1 `typedef u16string std::unary_function< u16string , size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.485.2.2 `typedef size_t std::unary_function< u16string , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

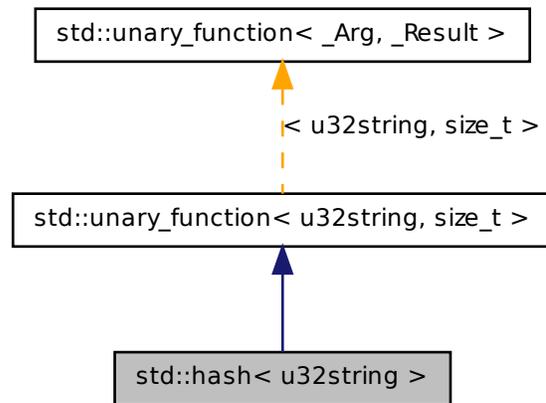
The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.486 `std::hash< u32string >` Struct Template Reference

[std::hash](#) specialization for `u32string`.

Inheritance diagram for std::hash< u32string >:



## Public Types

- typedef `u32string` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (const `u32string` &\_\_s) const

### 5.486.1 Detailed Description

`template<> struct std::hash< u32string >`

`std::hash` specialization for `u32string`.

Definition at line 2922 of file `basic_string.h`.

## 5.486.2 Member Typedef Documentation

### 5.486.2.1 `typedef u32string std::unary_function< u32string , size_t >::argument_type` [inherited]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.486.2.2 `typedef size_t std::unary_function< u32string , size_t >::result_type` [inherited]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

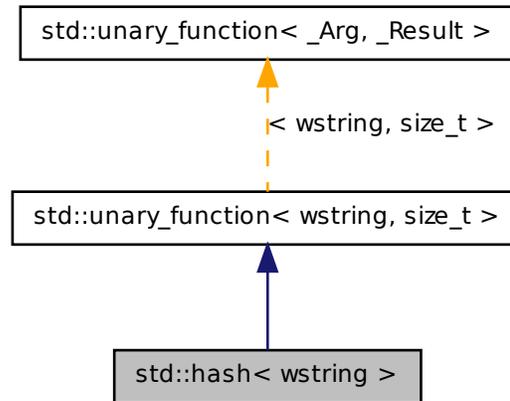
The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.487 `std::hash< wstring >` Struct Template Reference

[std::hash](#) specialization for `wstring`.

Inheritance diagram for std::hash< wstring >:



## Public Types

- typedef `wstring` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator() (const wstring &__s) const`

### 5.487.1 Detailed Description

`template<> struct std::hash< wstring >`

`std::hash` specialization for `wstring`.

Definition at line 2897 of file `basic_string.h`.

## 5.487.2 Member Typedef Documentation

### 5.487.2.1 `typedef wstring std::unary_function< wstring , size_t >::argument_type` [inherited]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.487.2.2 `typedef size_t std::unary_function< wstring , size_t >::result_type` [inherited]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

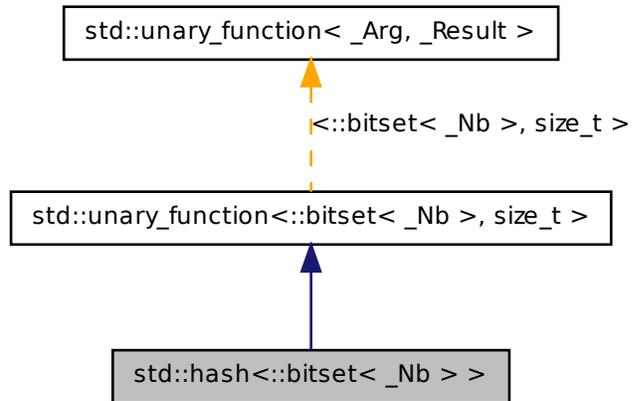
The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.488 `std::hash<::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inheritance diagram for std::hash<::bitset<\_Nb >>:



## Public Types

- typedef `::bitset<_Nb >` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (`const ::bitset<_Nb > &__b`) `const`

### 5.488.1 Detailed Description

`template<size_t _Nb> struct std::hash<::bitset<_Nb >>`

`std::hash` specialization for `bitset`.

Definition at line 1498 of file `bitset`.

## 5.488.2 Member Typedef Documentation

**5.488.2.1** `typedef ::bitset< _Nb > std::unary_function< ::bitset< _Nb > , size_t >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.488.2.2** `typedef size_t std::unary_function< ::bitset< _Nb > , size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

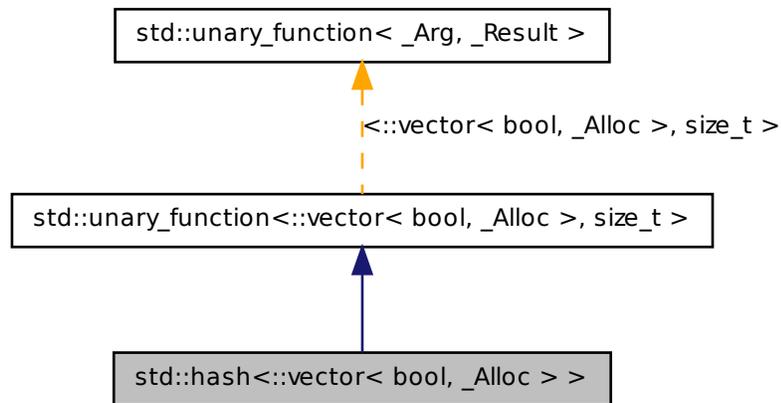
The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.489 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inheritance diagram for `std::hash<::vector< bool, _Alloc >>`:



## Public Types

- typedef `::vector< bool, _Alloc >` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator() (const ::vector< bool, _Alloc > &__b) const`

### 5.489.1 Detailed Description

`template<typename _Alloc> struct std::hash<::vector< bool, _Alloc >>`

`std::hash` specialization for `vector<bool>`.

Definition at line 1040 of file `stl_bvector.h`.

## 5.489.2 Member Typedef Documentation

### 5.489.2.1 `typedef ::vector< bool, _Alloc > std::unary_function< ::vector< bool, _Alloc >, size_t>::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.489.2.2 `typedef size_t std::unary_function< ::vector< bool, _Alloc >, size_t>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 5.490 `std::identity< _Tp >` Struct Template Reference

`identity`

### Public Types

- `typedef _Tp type`

### 5.490.1 Detailed Description

`template<typename _Tp> struct std::identity< _Tp >`

`identity`

Definition at line 44 of file `move.h`.

The documentation for this struct was generated from the following file:

- [move.h](#)

**5.491** `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType`  
> **Class Template Reference** **2461**

---

**5.491** `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType`  
> **Class Template Reference**

## Public Types

- typedef `_UIntType` `result_type`

## Public Member Functions

- `independent_bits_engine` ()
- `independent_bits_engine` (const `_RandomNumberEngine` &\_\_rne)
- `independent_bits_engine` (`result_type` \_\_s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type`>  
`independent_bits_engine` (`_Sseq` &\_\_q)
- `independent_bits_engine` (`_RandomNumberEngine` &&\_\_rne)
- const `_RandomNumberEngine` & `base` () const
- void `discard` (unsigned long long \_\_z)
- `result_type` `max` () const
- `result_type` `min` () const
- `result_type` `operator()` ()
- void `seed` (`result_type` \_\_s)
- template<typename `_Sseq` >  
void `seed` (`_Sseq` &\_\_q)
- void `seed` ()

## Friends

- bool `operator==` (const `independent_bits_engine` &\_\_lhs, const `independent_bits_engine` &\_\_rhs)
- template<typename `_CharT` , typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & `operator>>` (`std::basic_istream`< `_CharT`, `_Traits` > &\_\_is, `std::independent_bits_engine`< `_RandomNumberEngine`, \_\_w, `_UIntType` > &\_\_x)

### 5.491.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1010 of file random.h.

### 5.491.2 Member Typedef Documentation

**5.491.2.1** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type`

The type of the generated random value.

Definition at line 1019 of file random.h.

### 5.491.3 Constructor & Destructor Documentation

**5.491.3.1** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( ) [inline]`

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1026 of file random.h.

**5.491.3.2** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( const _RandomNumberEngine & __rne ) [inline, explicit]`

Copy constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

### 5.491 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>` 2463 > Class Template Reference

---

Definition at line 1036 of file random.h.

```
5.491.3.3 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
( _RandomNumberEngine && __rne ) [inline, explicit]
```

Move constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 1046 of file random.h.

```
5.491.3.4 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
( result_type __s ) [inline, explicit]
```

Seed constructs a `independent_bits_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

#### Parameters

*\_\_s* A seed value for the base class engine.

Definition at line 1056 of file random.h.

```
5.491.3.5 template<typename _RandomNumberEngine, size_t
__w, typename _UIntType> template<typename _Sseq ,
typename = typename std::enable_if<!std::is_same<_Sseq,
independent_bits_engine>::value && !std::is_same<_Sseq,
_RandomNumberEngine>::value> ::type> std::independent_
bits_engine<_RandomNumberEngine, __w, _UIntType
>::independent_bits_engine ( _Sseq & __q ) [inline,
explicit]
```

Generator construct a `independent_bits_engine` engine.

#### Parameters

*\_\_q* A seed sequence.

Definition at line 1069 of file random.h.

---

## 5.491.4 Member Function Documentation

**5.491.4.1** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> const _RandomNumberEngine& std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::base ( ) const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 1104 of file random.h.

**5.491.4.2** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard ( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

### Todo

Look for a faster way to do discard.

Definition at line 1131 of file random.h.

**5.491.4.3** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::max ( ) const [inline]`

Gets the maximum value in the generated random number range.

### Todo

This should be constexpr.

Definition at line 1122 of file random.h.

**5.491.4.4** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::min ( ) const [inline]`

Gets the minimum value in the generated random number range.

### Todo

This should be `constexpr`.

Definition at line 1113 of file `random.h`.

**5.491.4.5** `template<typename _RandomNumberEngine, size_t __w, typename  
_UIntType > independent_bits_engine<_RandomNumberEngine,  
__w, _UIntType >::result_type std::independent_bits_engine<  
_RandomNumberEngine, __w, _UIntType >::operator() ( )`

Gets the next value in the generated random number sequence.

Definition at line 719 of file `random.tcc`.

References `std::log()`.

**5.491.4.6** `template<typename _RandomNumberEngine, size_t __w,  
typename _UIntType> template<typename _Sseq > void  
std::independent_bits_engine<_RandomNumberEngine, __w,  
_UIntType >::seed ( _Sseq & __q ) [inline]`

Reseeds the `independent_bits_engine` object with the given seed sequence.

### Parameters

`__q` A seed generator function.

Definition at line 1096 of file `random.h`.

**5.491.4.7** `template<typename _RandomNumberEngine, size_t __w,  
typename _UIntType> void std::independent_bits_engine<  
_RandomNumberEngine, __w, _UIntType >::seed ( result_type __s  
) [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1086 of file `random.h`.

**5.491.4.8** `template<typename _RandomNumberEngine, size_t __w,  
typename _UIntType> void std::independent_bits_engine<  
_RandomNumberEngine, __w, _UIntType >::seed ( ) [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1078 of file random.h.

### 5.491.5 Friends And Related Function Documentation

**5.491.5.1** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==( const independent_bits_engine<_RandomNumberEngine, __w, _UIntType > & __lhs, const independent_bits_engine<_RandomNumberEngine, __w, _UIntType > & __rhs ) [friend]`

Compares two `independent_bits_engine` random number generator objects of the same type for equality.

#### Parameters

`__lhs` A `independent_bits_engine` random number generator object.

`__rhs` Another `independent_bits_engine` random number generator object.

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1156 of file random.h.

**5.491.5.2** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & __is, std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType > & __x ) [friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `independent_bits_engine` random number generator engine.

#### Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1174 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.492 `std::indirect_array< _Tp >` Class Template Reference

Reference to arbitrary subset of an array.

### Public Types

- typedef `_Tp value_type`

### Public Member Functions

- [indirect\\_array](#) (const [indirect\\_array](#) &)
- `template<class _Dom >`  
void **operator**`%=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`%=` (const [valarray](#)< `_Tp` > &) const
- void [operator](#)`&=` (const [valarray](#)< `_Tp` > &) const
- `template<class _Dom >`  
void **operator**`&=` (const `_Expr< _Dom, _Tp >` &) const
- `template<class _Dom >`  
void **operator**`*=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`*=` (const [valarray](#)< `_Tp` > &) const
- `template<class _Dom >`  
void **operator**`+=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`+=` (const [valarray](#)< `_Tp` > &) const
- void [operator](#)`-=` (const [valarray](#)< `_Tp` > &) const
- `template<class _Dom >`  
void **operator**`-=` (const `_Expr< _Dom, _Tp >` &) const
- `template<class _Dom >`  
void **operator**`/=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`/=` (const [valarray](#)< `_Tp` > &) const
- void [operator](#)`<<=` (const [valarray](#)< `_Tp` > &) const
- `template<class _Dom >`  
void **operator**`<<=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`=` (const `_Tp` &) const
- [indirect\\_array](#) & [operator](#)`=` (const [indirect\\_array](#) &)
- `template<class _Dom >`  
void **operator**`=` (const `_Expr< _Dom, _Tp >` &) const
- void [operator](#)`=` (const [valarray](#)< `_Tp` > &) const

- `template<class _Dom >`  
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator|= (const _Expr< _Dom, _Tp > &) const`

## Friends

- `class gslice_array< _Tp >`
- `class valarray< _Tp >`

### 5.492.1 Detailed Description

`template<class _Tp> class std::indirect_array< _Tp >`

Reference to arbitrary subset of an array. An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

#### Parameters

*Tp* Element type.

Definition at line 61 of file `indirect_array.h`.

### 5.492.2 Member Function Documentation

**5.492.2.1** `template<class _Tp> void std::indirect_array< _Tp >::operator%= ( const valarray< _Tp > & ) const`

Modulo slice elements by corresponding elements of *v*.

**5.492.2.2** `template<class _Tp> void std::indirect_array< _Tp >::operator&= ( const valarray< _Tp > & ) const`

Logical and slice elements with corresponding elements of *v*.

**5.492.2.3** `template<class _Tp> void std::indirect_array< _Tp >::operator*=( const valarray< _Tp > & ) const`

Multiply slice elements by corresponding elements of *v*.

**5.492.2.4** `template<class _Tp> void std::indirect_array< _Tp >::operator+=( const valarray< _Tp > & ) const`

Add corresponding elements of *v* to slice elements.

**5.492.2.5** `template<class _Tp> void std::indirect_array< _Tp >::operator-=( const valarray< _Tp > & ) const`

Subtract corresponding elements of *v* from slice elements.

**5.492.2.6** `template<class _Tp> void std::indirect_array< _Tp >::operator/=( const valarray< _Tp > & ) const`

Divide slice elements by corresponding elements of *v*.

**5.492.2.7** `template<class _Tp> void std::indirect_array< _Tp >::operator<<=( const valarray< _Tp > & ) const`

Left shift slice elements by corresponding elements of *v*.

**5.492.2.8** `template<class _Tp> void std::indirect_array< _Tp >::operator>>=( const valarray< _Tp > & ) const`

Right shift slice elements by corresponding elements of *v*.

**5.492.2.9** `template<class _Tp> void std::indirect_array< _Tp >::operator^=( const valarray< _Tp > & ) const`

Logical xor slice elements with corresponding elements of *v*.

**5.492.2.10** `template<class _Tp> void std::indirect_array< _Tp >::operator|=  
( const valarray< _Tp > & ) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [indirect\\_array.h](#)

## 5.493 `std::initializer_list< _E >` Class Template Reference

[initializer\\_list](#)

### Public Types

- typedef const \_E \* **const\_iterator**
- typedef const \_E & **const\_reference**
- typedef const \_E \* **iterator**
- typedef const \_E & **reference**
- typedef size\_t **size\_type**
- typedef \_E **value\_type**

### Public Member Functions

- const\_iterator **begin** () const
- const\_iterator **end** () const
- size\_type **size** () const

#### 5.493.1 Detailed Description

`template<class _E> class std::initializer_list< _E >`

[initializer\\_list](#)

Definition at line 45 of file `initializer_list`.

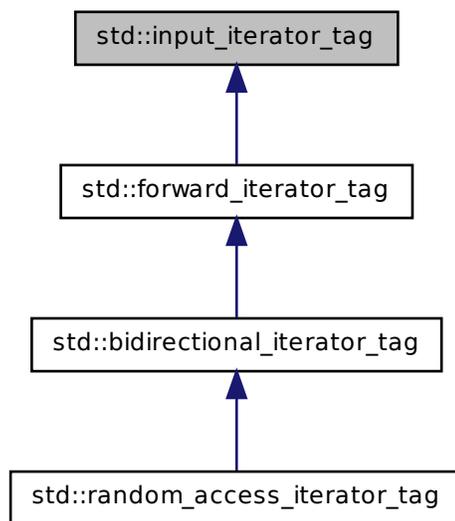
The documentation for this class was generated from the following file:

- [initializer\\_list](#)

## 5.494 `std::input_iterator_tag` Struct Reference

Marking input iterators.

Inheritance diagram for `std::input_iterator_tag`:



### 5.494.1 Detailed Description

Marking input iterators.

Definition at line 85 of file `stl_iterator_base_types.h`.

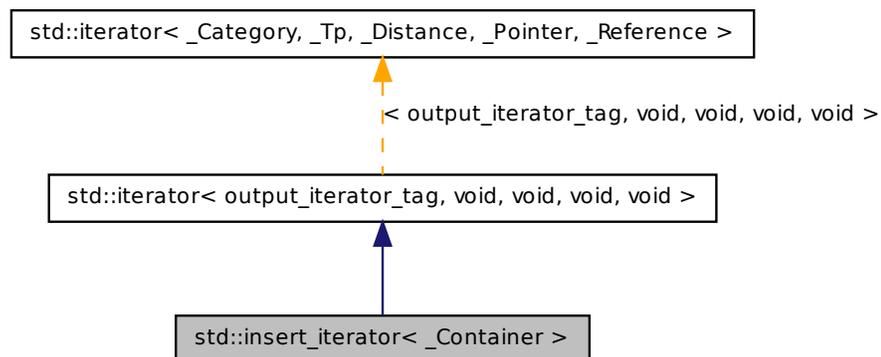
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.495 `std::insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::insert_iterator< _Container >`:



### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator*` ()
- `insert_iterator & operator++` ()
- `insert_iterator & operator++` (int)
- `insert_iterator & operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator & operator=` (`typename _Container::value_type &&__value`)

## Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

### 5.495.1 Detailed Description

**template<typename \_Container> class `std::insert_iterator<_Container>`**

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 579 of file `stl_iterator.h`.

### 5.495.2 Member Typedef Documentation

**5.495.2.1 `template<typename _Container> typedef _Container std::insert_iterator<_Container>::container_type`**

A nested typedef for the type of whatever container you used.

Definition at line 588 of file `stl_iterator.h`.

**5.495.2.2 `typedef void std::iterator<output_iterator_tag, void, void, void, void>::difference_type [inherited]`**

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.495.2.3 `typedef output_iterator_tag std::iterator<output_iterator_tag, void, void, void, void>::iterator_category [inherited]`**

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.495.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file stl\_iterator\_base\_types.h.

**5.495.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

**5.495.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file stl\_iterator\_base\_types.h.

### 5.495.3 Constructor & Destructor Documentation

**5.495.3.1** `template<typename _Container> std::insert_iterator< _Container >::insert_iterator ( _Container & __x, typename _Container::iterator __i ) [inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 594 of file stl\_iterator.h.

### 5.495.4 Member Function Documentation

**5.495.4.1** `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator*( ) [inline]`

Simply returns \*this.

Definition at line 648 of file stl\_iterator.h.

**5.495.4.2** `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++( int ) [inline]`

Simply returns `*this`. (This iterator does not *move*).

Definition at line 658 of file `stl_iterator.h`.

**5.495.4.3** `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator++( ) [inline]`

Simply returns `*this`. (This iterator does not *move*).

Definition at line 653 of file `stl_iterator.h`.

**5.495.4.4** `template<typename _Container> insert_iterator& std::insert_iterator<_Container>::operator=( const typename _Container::value_type & __value ) [inline]`

#### Parameters

*value* An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

#### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

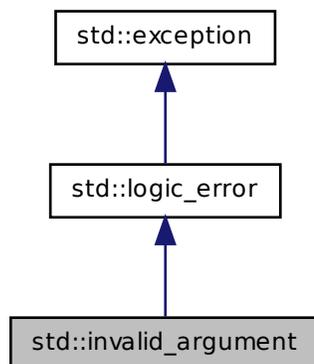
Definition at line 630 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.496 `std::invalid_argument` Class Reference

Inheritance diagram for `std::invalid_argument`:



### Public Member Functions

- `invalid_argument` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.496.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 80 of file `stdexcept`.

#### 5.496.2 Member Function Documentation

##### 5.496.2.1 virtual const char\* `std::logic_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

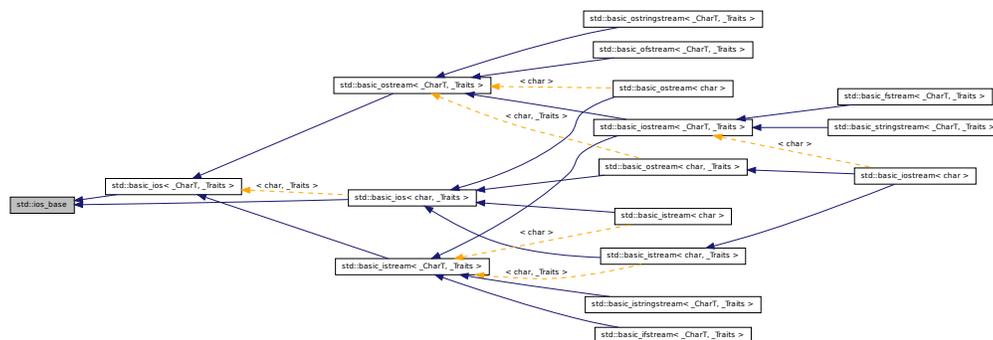
- [stdexcept](#)

## 5.497 std::ios\_base Class Reference

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios\\_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Inheritance diagram for `std::ios_base`:



### Classes

- class [failure](#)

*These are thrown to indicate problems with io.*  
 27.4.2.1.1 Class `ios_base::failure`.

### Public Types

- enum [event](#) { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* [event\\_callback](#) )(event, [ios\\_base](#) &, int)
- typedef `_Ios_Fmtflags` [fmtflags](#)
- typedef int [io\\_state](#)

- typedef `_Ios_Iostate` [iostate](#)
- typedef int `open_mode`
- typedef `_Ios_Openmode` [openmode](#)
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**

## Public Member Functions

- virtual `~ios_base` ()
- const `locale & _M_getloc` () const
- `fmtflags flags` () const
- `fmtflags flags` (`fmtflags __fmtfl`)
- `locale getloc` () const
- `locale imbue` (const `locale & __loc`) throw ()
- `long & iword` (int `__ix`)
- `streamsize precision` (`streamsize __prec`)
- `streamsize precision` () const
- void `*& pword` (int `__ix`)
- void `register_callback` (`event_callback __fn`, int `__index`)
- `fmtflags setf` (`fmtflags __fmtfl`)
- `fmtflags setf` (`fmtflags __fmtfl`, `fmtflags __mask`)
- void `unsetf` (`fmtflags __mask`)
- `streamsize width` () const
- `streamsize width` (`streamsize __wide`)

## Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync=true`)
- static int `xalloc` () throw ()

## Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`

- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void `_M_call_callbacks` (`event` `__ev`) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int `__index`, bool `__iword`)
- void `_M_init` () throw ()

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `ios_base _M_exception`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `streamsize _M_precision`
- `ios_base _M_streambuf_state`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### 5.497.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 207 of file `ios_base.h`.

### 5.497.2 Member Typedef Documentation

#### 5.497.2.1 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 444 of file `ios_base.h`.

### 5.497.2.2 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 263 of file `ios_base.h`.

### 5.497.2.3 typedef `_Ios_Iostate` `std::ios_base::iostate`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 338 of file `ios_base.h`.

### 5.497.2.4 typedef `_Ios_Openmode` `std::ios_base::openmode`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 369 of file `ios_base.h`.

### 5.497.2.5 typedef `_Ios_Seekdir` `std::ios_base::seekdir`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.

- end, equivalent to SEEK\_END in the C standard library.

Definition at line 401 of file ios\_base.h.

## 5.497.3 Member Enumeration Documentation

### 5.497.3.1 enum std::ios\_base::event

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 427 of file ios\_base.h.

## 5.497.4 Constructor & Destructor Documentation

### 5.497.4.1 virtual std::ios\_base::~ios\_base ( ) [virtual]

Invokes each callback with erase\_event. Destroys local storage.

Note that the ios\_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase\_event (unless copyfmt is used).

## 5.497.5 Member Function Documentation

### 5.497.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline]

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios\_base.h.

Referenced by std::money\_get<\_CharT, \_InIter >::do\_get(), std::num\_get<\_CharT, \_InIter >::do\_get(), std::time\_get<\_CharT, \_InIter >::do\_get\_date(), std::time\_get<\_CharT, \_InIter >::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter >::do\_get\_time(), std::time\_get<\_CharT, \_InIter >::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter >::do\_get\_year(), std::time\_put<\_CharT, \_OutIter >::do\_put(), std::num\_put<\_CharT, \_OutIter >::do\_put(), and std::time\_put<\_CharT, \_OutIter >::put().

### 5.497.5.2 `fmtflags std::ios_base::flags ( ) const [inline]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

### 5.497.5.3 `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

### 5.497.5.4 `locale std::ios_base::getloc ( ) const [inline]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.497.5.5 locale std::ios\_base::imbue ( const locale & \_\_loc ) throw ()**

Setting a new locale.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue\_event.

Reimplemented in [std::basic\\_ios< \\_CharT, \\_Traits >](#), and [std::basic\\_ios< char, \\_Traits >](#).

**5.497.5.6 long& std::ios\_base::iword ( int \_\_ix ) [inline]**

Access to integer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios\_base.h.

**5.497.5.7 streamsize std::ios\_base::precision ( ) const [inline]**

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits >::copyfmt(), and std::operator<<().

#### 5.497.5.8 streamsize std::ios\_base::precision ( streamsize \_\_prec ) [inline]

Changing flags.

##### Parameters

*prec* The new precision value.

##### Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios\_base.h.

#### 5.497.5.9 void\*& std::ios\_base::pword ( int \_\_ix ) [inline]

Access to void pointer array.

##### Parameters

*\_\_ix* Index into the array.

##### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios\_base.h.

#### 5.497.5.10 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index )

Add the callback *\_\_fn* with parameter *\_\_index*.

**Parameters**

- \_\_fn* The function to add.
- \_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.497.5.11 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl, fmtflags \_\_mask )  
[inline]**

Setting new format flags.

**Parameters**

- fmtfl* Additional flags to set.
- mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

**5.497.5.12 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl ) [inline]**

Setting new format flags.

**Parameters**

- fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::internal()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.497.5.13** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
`[static]`

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.497.5.14** `void std::ios_base::unsetf ( fmtflags __mask )` `[inline]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.497.5.15** `streamsize std::ios_base::width ( ) const` `[inline]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outputter>::do_put()`, and `std::operator>>()`.

**5.497.5.16** `streamsize std::ios_base::width ( streamsize __wide ) [inline]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

**5.497.5.17** `static int std::ios_base::xalloc ( ) throw () [static]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.497.6 Member Data Documentation****5.497.6.1** `const fmtflags std::ios_base::adjustfield [static]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.497.6.2** `const openmode std::ios_base::app [static]`

Seek to end before each write.

Definition at line 372 of file `ios_base.h`.

**5.497.6.3 const openmode std::ios\_base::ate [static]**

Open and seek to end immediately after opening.

Definition at line 375 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.497.6.4 const iostate std::ios\_base::badbit [static]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 342 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.497.6.5 const fmtflags std::ios\_base::basefield [static]**

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 321 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.497.6.6 const seekdir std::ios\_base::beg [static]**

Request a seek relative to the beginning of the stream.

Definition at line 404 of file ios\_base.h.

**5.497.6.7 const openmode std::ios\_base::binary [static]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 5.497.6.8 const fmtflags std::ios\_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 266 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), and std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.497.6.9 const seekdir std::ios\_base::cur [static]

Request a seek relative to the current position within the sequence.

Definition at line 407 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 5.497.6.10 const fmtflags std::ios\_base::dec [static]

Converts integer input or generates integer output in decimal base.

Definition at line 269 of file ios\_base.h.

#### 5.497.6.11 const seekdir std::ios\_base::end [static]

Request a seek relative to the current end of the sequence.

Definition at line 410 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

#### 5.497.6.12 const iostate std::ios\_base::eofbit [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 345 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter

>::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), and std::ws().

#### 5.497.6.13 const iostate std::ios\_base::failbit [static]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 350 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), and std::basic\_ostream< \_CharT, \_Traits >::seekp().

#### 5.497.6.14 const fmtflags std::ios\_base::fixed [static]

Generate floating-point output in fixed-point notation.

Definition at line 272 of file ios\_base.h.

#### 5.497.6.15 const fmtflags std::ios\_base::floatfield [static]

A mask of scientific|fixed. Useful for the 2-arg form of *setf*.

Definition at line 324 of file ios\_base.h.

#### 5.497.6.16 const iostate std::ios\_base::goodbit [static]

Indicates all is well.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),

std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 5.497.6.17 const fmtflags std::ios\_base::hex [static]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 275 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.497.6.18 const openmode std::ios\_base::in [static]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 383 of file ios\_base.h.

Referenced by std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

#### 5.497.6.19 const fmtflags std::ios\_base::internal [static]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 280 of file ios\_base.h.

#### 5.497.6.20 const fmtflags std::ios\_base::left [static]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 284 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.497.6.21 const fmtflags std::ios\_base::oct [static]**

Converts integer input or generates integer output in octal base.

Definition at line 287 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.497.6.22 const openmode std::ios\_base::out [static]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 386 of file ios\_base.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::seekp(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.497.6.23 const fmtflags std::ios\_base::right [static]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.)

Definition at line 291 of file ios\_base.h.

**5.497.6.24 const fmtflags std::ios\_base::scientific [static]**

Generates floating-point output in scientific notation.

Definition at line 294 of file ios\_base.h.

**5.497.6.25 const fmtflags std::ios\_base::showbase [static]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 298 of file ios\_base.h.

**5.497.6.26 const fmtflags std::ios\_base::showpoint [static]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 302 of file ios\_base.h.

**5.497.6.27 const fmtflags std::ios\_base::showpos [static]**

Generates a + sign in non-negative generated numeric output.

Definition at line 305 of file `ios_base.h`.

#### 5.497.6.28 `const fmtflags std::ios_base::skipws` [static]

Skips leading white space before certain input operations.

Definition at line 308 of file `ios_base.h`.

#### 5.497.6.29 `const openmode std::ios_base::trunc` [static]

Open for input. Default for `ofstream`.

Definition at line 389 of file `ios_base.h`.

#### 5.497.6.30 `const fmtflags std::ios_base::unitbuf` [static]

Flushes output after each output operation.

Definition at line 311 of file `ios_base.h`.

#### 5.497.6.31 `const fmtflags std::ios_base::uppercase` [static]

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

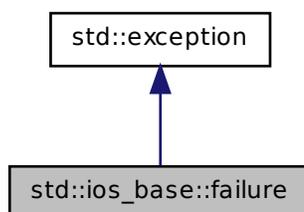
- [ios\\_base.h](#)

## 5.498 `std::ios_base::failure` Class Reference

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

Inheritance diagram for `std::ios_base::failure`:



## Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- virtual const char \* **what** () const throw ()

### 5.498.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios\\_base::failure](#).

Definition at line 217 of file `ios_base.h`.

### 5.498.2 Member Function Documentation

#### 5.498.2.1 virtual const char\* `std::ios_base::failure::what` ( ) const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 5.499 `std::is_base_of< _Base, _Derived >` Struct Template Reference

[is\\_base\\_of](#)

Inherits `integral_constant< bool, __is_base_of(_Base, _Derived)>`.

### 5.499.1 Detailed Description

```
template<typename _Base, typename _Derived> struct std::is_base_of< _Base,
_Derived >
```

[is\\_base\\_of](#)

Definition at line 283 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.500 `std::is_bind_expression< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Inherits `false_type`.

### 5.500.1 Detailed Description

```
template<typename _Tp> struct std::is_bind_expression< _Tp >
```

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Definition at line 790 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.501 `std::is_bind_expression< _Bind< _Signature >>` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

### 5.501.1 Detailed Description

```
template<typename _Signature> struct std::is_bind_expression< _Bind< _Signature >>
```

Class template `_Bind` is always a bind expression.

Definition at line 1366 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.502 `std::is_bind_expression< _Bind_result< _Result, _Signature >>` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

### 5.502.1 Detailed Description

```
template<typename _Result, typename _Signature> struct std::is_bind_expression< _Bind_result< _Result, _Signature >>
```

Class template `_Bind` is always a bind expression.

Definition at line 1374 of file `functional`.

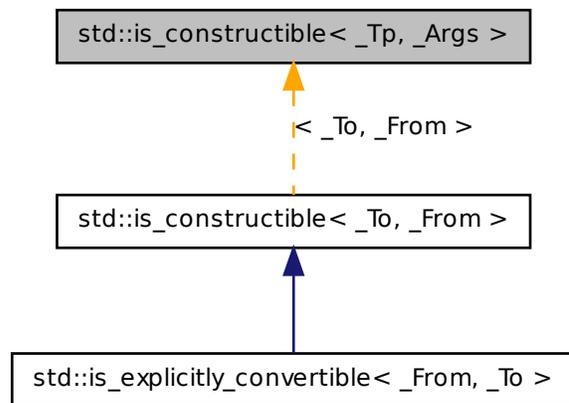
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.503 `std::is_constructible< _Tp, _Args >` Struct Template Reference

[is\\_constructible](#)

Inheritance diagram for `std::is_constructible< _Tp, _Args >`:



### 5.503.1 Detailed Description

```
template<typename _Tp, typename... _Args> struct std::is_constructible< _Tp,
_AArgs >
```

[is\\_constructible](#)

Definition at line 231 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.504 `std::is_convertible< _From, _To >` Struct Template Reference

[is\\_convertible](#)

Inherits `integral_constant< bool, __is_convertible_helper< _From, _To >::__value >`.

### 5.504.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_convertible< _From,
_To >
```

[is\\_convertible](#)

Definition at line 309 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.505 `std::is_error_code_enum< _Tp >` Struct Template Reference

[is\\_error\\_code\\_enum](#)

Inherits `false_type`.

### 5.505.1 Detailed Description

```
template<typename _Tp> struct std::is_error_code_enum< _Tp >
```

[is\\_error\\_code\\_enum](#)

Definition at line 52 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## **5.506** `std::is_error_condition_enum< _Tp >` Struct Template Reference

[is\\_error\\_condition\\_enum](#)

Inherits `false_type`.

### **5.506.1** Detailed Description

```
template<typename _Tp> struct std::is_error_condition_enum< _Tp >
```

[is\\_error\\_condition\\_enum](#)

Definition at line 56 of file `system_error`.

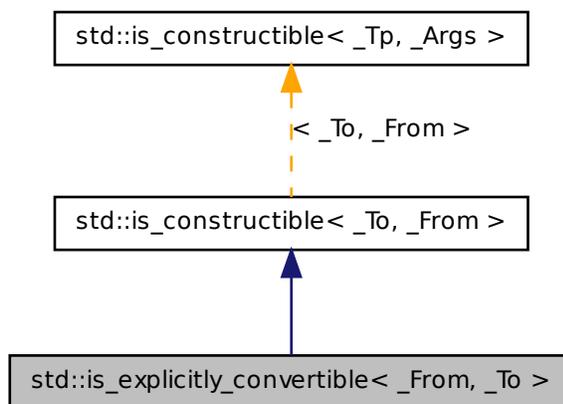
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## **5.507** `std::is_explicitly_convertible< _From, _To >` Struct Template Reference

[is\\_explicitly\\_convertible](#)

Inheritance diagram for `std::is_explicitly_convertible<_From, _To>`:



### 5.507.1 Detailed Description

**template**<typename `_From`, typename `_To`> **struct** `std::is_explicitly_convertible<_From, _To>`

[is\\_explicitly\\_convertible](#)

Definition at line 316 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.508 `std::is_lvalue_reference< typename >` Struct Template Reference

[is\\_lvalue\\_reference](#)

Inherits `false_type`.

### 5.508.1 Detailed Description

`template<typename> struct std::is_lvalue_reference< typename >`

[is\\_lvalue\\_reference](#)

Definition at line 69 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.509 `std::is_placeholder< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Inherits `integral_constant< int, 0 >`.

### 5.509.1 Detailed Description

`template<typename _Tp> struct std::is_placeholder< _Tp >`

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Definition at line 799 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.510 `std::is_placeholder< _Placeholder< _Num > >` Struct Template Reference

Inherits `integral_constant< int, _Num >`.

### 5.510.1 Detailed Description

`template<int _Num> struct std::is_placeholder< _Placeholder< _Num > >`

Partial specialization of [is\\_placeholder](#) that provides the placeholder number for the placeholder objects defined by libstdc++.

Definition at line 856 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.511 std::is\_pod< \_Tp > Struct Template Reference

[is\\_pod](#)

Inherits `integral_constant< bool, __is_pod(_Tp)>`.

### 5.511.1 Detailed Description

`template<typename _Tp> struct std::is_pod< _Tp >`

[is\\_pod](#)

Definition at line 191 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.512 std::is\_reference< \_Tp > Struct Template Reference

[is\\_reference](#)

Inherits `integral_constant< bool,(is_lvalue_reference< _Tp >::value||is_rvalue_reference< _Tp >::value)>`.

### 5.512.1 Detailed Description

`template<typename _Tp> struct std::is_reference< _Tp >`

[is\\_reference](#)

## **5.513 `std::is_rvalue_reference< typename >` Struct Template Reference 2505**

Definition at line 89 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## **5.513 `std::is_rvalue_reference< typename >` Struct Template Reference**

[is\\_rvalue\\_reference](#)

Inherits `false_type`.

### **5.513.1 Detailed Description**

`template<typename> struct std::is_rvalue_reference< typename >`

[is\\_rvalue\\_reference](#)

Definition at line 78 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## **5.514 `std::is_signed< _Tp >` Struct Template Reference**

[is\\_signed](#)

Inherits `integral_constant< bool, __is_signed_helper< _Tp >::value >`.

### **5.514.1 Detailed Description**

`template<typename _Tp> struct std::is_signed< _Tp >`

[is\\_signed](#)

Definition at line 163 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.515 `std::is_standard_layout`< `_Tp` > Struct Template Reference

[is\\_standard\\_layout](#)

Inherits `integral_constant< bool, __is_standard_layout(_Tp)>`.

### 5.515.1 Detailed Description

```
template<typename _Tp> struct std::is_standard_layout< _Tp >
```

[is\\_standard\\_layout](#)

Definition at line 184 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.516 `std::is_trivial`< `_Tp` > Struct Template Reference

[is\\_trivial](#)

Inherits `integral_constant< bool, __is_trivial(_Tp)>`.

### 5.516.1 Detailed Description

```
template<typename _Tp> struct std::is_trivial< _Tp >
```

[is\\_trivial](#)

Definition at line 178 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.517 `std::is_unsigned`< `_Tp` > Struct Template Reference

[is\\_unsigned](#)

## 5.518 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference 2507

Inherits `integral_constant< bool, (is_arithmetic< _Tp >::value &&!is_signed< _Tp >::value)>`.

### 5.517.1 Detailed Description

```
template<typename _Tp> struct std::is_unsigned< _Tp >
```

[is\\_unsigned](#)

Definition at line 169 of file `type_traits`.

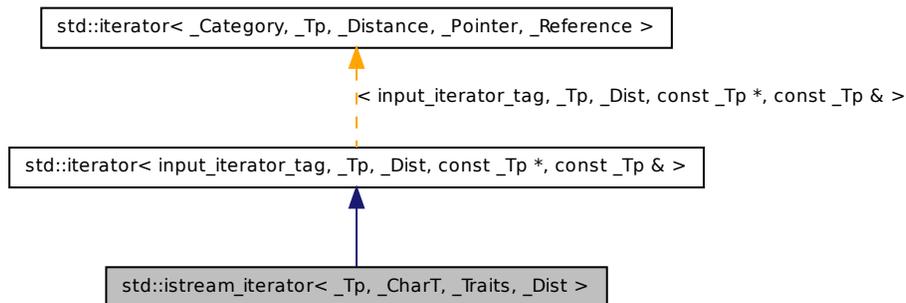
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.518 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Provides input iterator semantics for streams.

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` **difference\_type**
- typedef `basic_istream< _CharT, _Traits >` **istream\_type**

- typedef [input\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef const \_Tp \* [pointer](#)
- typedef const \_Tp & [reference](#)
- typedef \_Traits [traits\\_type](#)
- typedef \_Tp [value\\_type](#)

## Public Member Functions

- [istream\\_iterator](#) ()
- [istream\\_iterator](#) ([istream\\_type](#) &\_\_s)
- [istream\\_iterator](#) (const [istream\\_iterator](#) &\_\_obj)
- bool [\\_M\\_equal](#) (const [istream\\_iterator](#) &\_\_x) const
- const \_Tp & [operator\\*](#) () const
- [istream\\_iterator](#) & [operator++](#) ()
- [istream\\_iterator](#) [operator++](#) (int)
- const \_Tp \* [operator->](#) () const

### 5.518.1 Detailed Description

`template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> class std::istream_iterator<_Tp, _CharT, _Traits, _Dist >`

Provides input iterator semantics for streams.

Definition at line 47 of file `stream_iterator.h`.

### 5.518.2 Member Typedef Documentation

**5.518.2.1** `typedef _Dist std::iterator< input\_iterator\_tag , _Tp, _Dist , const _Tp * , const _Tp & >::difference_type` [[inherited](#)]

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.518.2.2** `typedef input\_iterator\_tag std::iterator< input\_iterator\_tag , _Tp, _Dist , const _Tp * , const _Tp & >::iterator_category` [[inherited](#)]

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.518 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > Class Template Reference** 2509

---

**5.518.2.3** `typedef const _Tp * std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp *, const _Tp & >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file stl\_iterator\_base\_types.h.

**5.518.2.4** `typedef const _Tp & std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp *, const _Tp & >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

**5.518.2.5** `typedef _Tp std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp *, const _Tp & >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file stl\_iterator\_base\_types.h.

### 5.518.3 Constructor & Destructor Documentation

**5.518.3.1** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( ) [inline]`

Construct end of input stream iterator.

Definition at line 62 of file stream\_iterator.h.

**5.518.3.2** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( istream_type & __s ) [inline]`

Construct start of input stream iterator.

Definition at line 66 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.519 `std::istreambuf_iterator< _CharT, _Traits >` Class Template Reference

Provides input iterator semantics for streambufs.

Inheritance diagram for `std::istreambuf_iterator< _CharT, _Traits >`:



### Public Types

- typedef `_Traits::off_type` [difference\\_type](#)
  - typedef `input_iterator_tag` [iterator\\_category](#)
  - typedef `_CharT *` [pointer](#)
  - typedef `_CharT &` [reference](#)
  - typedef `_CharT` [value\\_type](#)
- 
- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
  - typedef `_Traits::int_type` [int\\_type](#)
  - typedef `basic_streambuf< _CharT, _Traits >` [streambuf\\_type](#)
  - typedef `basic_istream< _CharT, _Traits >` [istream\\_type](#)

### Public Member Functions

- `istreambuf_iterator` () throw ()
- `istreambuf_iterator` (`istream_type &__s`) throw ()
- `istreambuf_iterator` (`streambuf_type *__s`) throw ()
- `bool equal` (`const istreambuf_iterator &__b`) const
- `char_type operator*` () const
- `istreambuf_iterator operator++` (int)
- `istreambuf_iterator & operator++` ()

### Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2` (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator<`  
`_CharT2 >`, `_CharT2 *`)

## 5.519 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

- `template<typename _CharT2 > __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type >::__type` **copy** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `ostreambuf_iterator< _CharT2 >`)
- `template<typename _CharT2 > __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 >::__type >::__type` **find** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `const _CharT2 &`)

### 5.519.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::istreambuf_iterator< _CharT, _Traits >`

Provides input iterator semantics for streambufs.

Definition at line 50 of file `streambuf_iterator.h`.

### 5.519.2 Member Typedef Documentation

**5.519.2.1** `template<typename _CharT, typename _Traits > typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 58 of file `streambuf_iterator.h`.

**5.519.2.2** `typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT & >::difference_type`  
[*inherited*]

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.519.2.3** `template<typename _CharT, typename _Traits > typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type`

Public typedefs.

Definition at line 60 of file `streambuf_iterator.h`.

**5.519.2.4** `template<typename _CharT , typename _Traits > typedef  
basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT,  
_Traits >::istream_type`

Public typedefs.

Definition at line 62 of file `streambuf_iterator.h`.

**5.519.2.5** `typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT  
, _Traits::off_type , _CharT * , _CharT & >::iterator_category  
[inherited]`

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.519.2.6** `typedef _CharT * std::iterator< input_iterator_tag , _CharT ,  
_Traits::off_type , _CharT * , _CharT & >::pointer [inherited]`

This type represents a `pointer-to-value_type`.

Definition at line 123 of file `stl_iterator_base_types.h`.

**5.519.2.7** `typedef _CharT & std::iterator< input_iterator_tag , _CharT  
, _Traits::off_type , _CharT * , _CharT & >::reference  
[inherited]`

This type represents a `reference-to-value_type`.

Definition at line 125 of file `stl_iterator_base_types.h`.

**5.519.2.8** `template<typename _CharT , typename _Traits > typedef  
basic_streambuf<_CharT, _Traits> std::istreambuf_iterator<  
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 61 of file `streambuf_iterator.h`.

**5.519.2.9** `template<typename _CharT , typename _Traits > typedef _Traits  
std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 59 of file `streambuf_iterator.h`.

## **5.519 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

**5.519.2.10** `typedef _CharT std::iterator< input_iterator_tag , _CharT  
, _Traits::off_type , _CharT * , _CharT & >::value_type  
[inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

### **5.519.3 Constructor & Destructor Documentation**

**5.519.3.1** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 ) throw () [inline]`

Construct end of input stream iterator.

Definition at line 96 of file `streambuf_iterator.h`.

**5.519.3.2** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 istream_type & __s ) throw () [inline]`

Construct start of input stream iterator.

Definition at line 100 of file `streambuf_iterator.h`.

**5.519.3.3** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 streambuf_type * __s ) throw () [inline]`

Construct start of streambuf iterator.

Definition at line 104 of file `streambuf_iterator.h`.

### **5.519.4 Member Function Documentation**

**5.519.4.1** `template<typename _CharT , typename _Traits > bool  
std::istreambuf_iterator< _CharT, _Traits >::equal ( const  
istreambuf_iterator< _CharT, _Traits > & __b ) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 160 of file `streambuf_iterator.h`.

**5.519.4.2** `template<typename _CharT, typename _Traits > char_type  
std::istreambuf_iterator< _CharT, _Traits >::operator*( ) const  
[inline]`

Return the current character pointed to by iterator. This returns `/// streambuf.sgetc()`. It cannot be assigned. NB: The result of `/// operator*()` on an end of stream is undefined.

Definition at line 111 of file `streambuf_iterator.h`.

**5.519.4.3** `template<typename _CharT, typename _Traits >  
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits  
>::operator++( int ) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 140 of file `streambuf_iterator.h`.

References `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

**5.519.4.4** `template<typename _CharT, typename _Traits >  
istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits  
>::operator++( ) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 125 of file `streambuf_iterator.h`.

References `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

## **5.520** `std::iterator< _Category, _Tp, _Distance, _- Pointer, _Reference > Struct Template Reference`

Common iterator class.

Inheritance diagram for `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference`

>:



## Public Types

- typedef `_Distance` [difference\\_type](#)
- typedef `_Category` [iterator\\_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value\\_type](#)

### 5.520.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t,
typename _Pointer = _Tp*, typename _Reference = _Tp&> struct std::iterator<
_Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class. This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 114 of file `stl_iterator_base_types.h`.

### 5.520.2 Member Typedef Documentation

**5.520.2.1** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Distance std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::difference_type`

Distance between iterators is represented as this type.

Reimplemented in [std::reverse\\_iterator<\\_Iterator >](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.520.2.2** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category`

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.520.2.3** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value\_type.

Reimplemented in [std::reverse\\_iterator< \\_Iterator >](#).

Definition at line 123 of file `stl_iterator_base_types.h`.

**5.520.2.4** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`

This type represents a reference-to-value\_type.

Reimplemented in [std::reverse\\_iterator< \\_Iterator >](#).

Definition at line 125 of file `stl_iterator_base_types.h`.

**5.520.2.5** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

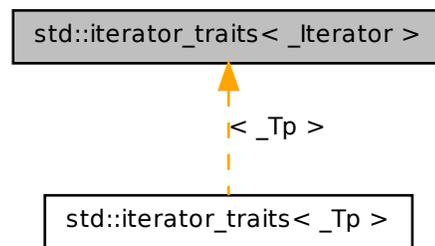
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.521 `std::iterator_traits<_Iterator>` Struct Template Reference

Traits class for iterators.

Inheritance diagram for `std::iterator_traits<_Iterator>`:



### Public Types

- `typedef _Iterator::difference_type` **difference\_type**
- `typedef _Iterator::iterator_category` **iterator\_category**
- `typedef _Iterator::pointer` **pointer**
- `typedef _Iterator::reference` **reference**
- `typedef _Iterator::value_type` **value\_type**

### 5.521.1 Detailed Description

```
template<typename _Iterator> struct std::iterator_traits<_Iterator>
```

Traits class for iterators. This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 137 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.522 `std::iterator_traits< _Tp * >` Struct Template Reference

Partial specialization for pointer types.

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

### 5.522.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 148 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.523 `std::iterator_traits< const _Tp * >` Struct Template Reference

Partial specialization for const pointer types.

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

### 5.523.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

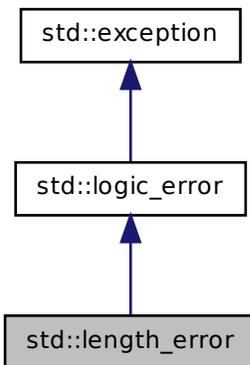
Definition at line 159 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.524 std::length\_error Class Reference

Inheritance diagram for std::length\_error:



### Public Member Functions

- **length\_error** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const throw ()

### 5.524.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a [basic\\_string](#) instance).

Definition at line 88 of file `stdexcept`.

### 5.524.2 Member Function Documentation

#### 5.524.2.1 `virtual const char* std::logic_error::what ( ) const throw ()` [`virtual, inherited`]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

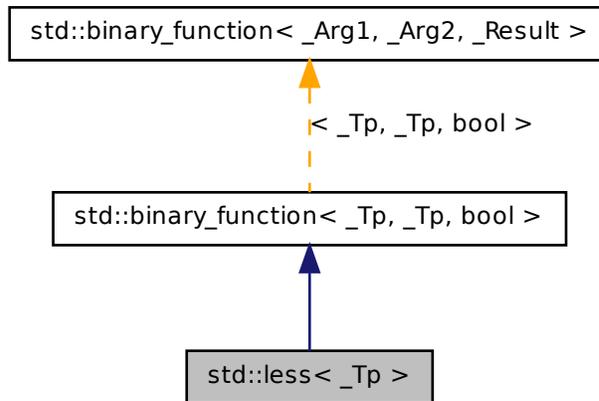
The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.525 `std::less< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::less< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.525.1 Detailed Description

```
template<typename _Tp> struct std::less< _Tp >
```

One of the [comparison functors](#).

Definition at line 226 of file `stl_function.h`.

## 5.525.2 Member Typedef Documentation

**5.525.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.525.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.525.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

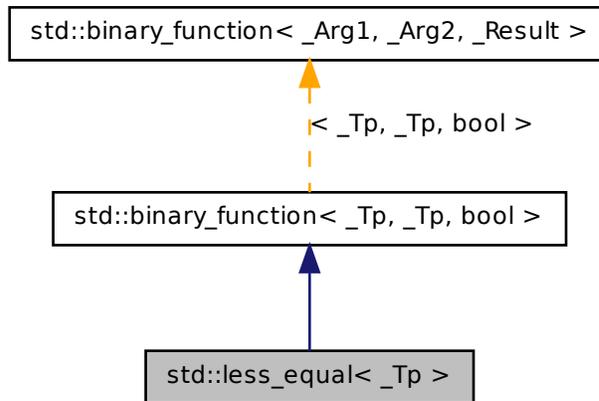
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.526 `std::less_equal< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::less\_equal< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.526.1 Detailed Description

```
template<typename _Tp> struct std::less_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 244 of file `stl_function.h`.

## 5.526.2 Member Typedef Documentation

**5.526.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [*inherited*]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.526.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [*inherited*]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.526.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [*inherited*]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.527 `std::linear_congruential_engine< _UIntType, _a, _c, _m >` Class Template Reference

A model of a linear congruential random number generator.

### Public Types

- `typedef _UIntType result_type`

### Public Member Functions

- `linear_congruential_engine` (`result_type __s=default_seed`)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type>`  
`linear_congruential_engine` (`_Sseq &__q`)

- void `discard` (unsigned long long `__z`)
- `result_type` `max` () const
- `result_type` `min` () const
- `result_type` `operator()` ()
- void `seed` (`result_type` `__s`=default\_seed)
- template<typename `_Sseq` >  
`std::enable_if< std::is_class<_Sseq >::value >::type` `seed` (`_Sseq` &`__q`)

## Static Public Attributes

- static const `result_type` `default_seed`
- static const `result_type` `increment`
- static const `result_type` `modulus`
- static const `result_type` `multiplier`

## Friends

- template<typename `_UIntType1` , `_UIntType1` `__a1`, `_UIntType1` `__c1`, `_UIntType1` `__m1`, type-  
 name `_CharT`, typename `_Traits` >  
`std::basic_ostream<_CharT, _Traits >` & `operator<<` (`std::basic_ostream<_`  
`CharT, _Traits >` &, const `std::linear_congruential_engine<_UIntType1, __a1,`  
`__c1, __m1 >` &)
- bool `operator==` (const `linear_congruential_engine` &`__lhs`, const `linear_`  
`congruential_engine` &`__rhs`)
- template<typename `_UIntType1` , `_UIntType1` `__a1`, `_UIntType1` `__c1`, `_UIntType1` `__m1`, type-  
 name `_CharT`, typename `_Traits` >  
`std::basic_istream<_CharT, _Traits >` & `operator>>` (`std::basic_istream<_`  
`CharT, _Traits >` &, `std::linear_congruential_engine<_UIntType1, __a1, __c1,`  
`__m1 >` &)

### 5.527.1 Detailed Description

`template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>` class `std::linear_congruential_engine<_UIntType, __a, __c, __m >`

A model of a linear congruential random number generator. A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 157 of file random.h.

## 5.527.2 Member Typedef Documentation

**5.527.2.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type`

The type of the generated random value.

Definition at line 166 of file random.h.

## 5.527.3 Constructor & Destructor Documentation

**5.527.3.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine ( result_type __s = default_seed ) [inline, explicit]`

Constructs a linear\_congruential\_engine random number generator engine with seed `__s`. The default seed value is 1.

### Parameters

`__s` The initial seed value.

Definition at line 184 of file random.h.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

**5.527.3.2** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value> ::type> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine ( _Sseq & __q ) [inline, explicit]`

Constructs a linear\_congruential\_engine random number generator engine seeded from the seed sequence `__q`.

### Parameters

`__q` the seed sequence.

## 5.527 `std::linear_congruential_engine<_UIntType, __a, __c, __m >` Class 2527 Template Reference

---

Definition at line 197 of file `random.h`.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

### 5.527.4 Member Function Documentation

**5.527.4.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m >::discard ( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

#### Todo

Look for a faster way to do discard.

Definition at line 247 of file `random.h`.

**5.527.4.2** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m >::max ( ) const [inline]`

Gets the largest possible value in the output range.

#### Todo

This should be `constexpr`.

Definition at line 238 of file `random.h`.

**5.527.4.3** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m >::min ( ) const [inline]`

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

#### Todo

This should be `constexpr`.

Definition at line 229 of file `random.h`.

**5.527.4.4** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::operator() ( ) [inline]`

Gets the next random number in the sequence.

Definition at line 257 of file random.h.

**5.527.4.5** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq > std::enable_if< std::is_class< _Sseq >::value >::type std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed ( _Sseq & __q )`

Reseeds the linear\_congruential\_engine random number generator engine sequence using values from the seed sequence \_\_q.

#### Parameters

`__q` the seed sequence.

Seeds the LCR engine with a value generated by \_\_q.

Definition at line 142 of file random.tcc.

References std::\_\_lg(), and std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::seed().

**5.527.4.6** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed ( result_type __s = default_seed )`

Reseeds the linear\_congruential\_engine random number generator engine sequence to the seed \_\_s.

#### Parameters

`__s` The new seed.

Seeds the LCR with integral value \_\_s, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 126 of file random.tcc.

Referenced by std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::linear\_congruential\_engine(), and std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::seed().

## 5.527.5 Friends And Related Function Documentation

5.527.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<<< ( std::basic_ostream<_CharT, _Traits > &, const std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1 > & ) [friend]`

Writes the textual representation of the state  $x(i)$  of  $x$  to `__os`.

### Parameters

`__os` The output stream.

`__lcr` A % [linear\\_congruential\\_engine](#) random number generator.

### Returns

`__os`.

5.527.5.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator==( const linear_congruential_engine<_UIntType, __a, __c, __m > & __lhs, const linear_congruential_engine<_UIntType, __a, __c, __m > & __rhs ) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

### Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 275 of file `random.h`.

```

5.527.5.3 template<typename _UIntType, _UIntType __a, _UIntType __c,
    _UIntType __m> template<typename _UIntType1, _UIntType1
    __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
    , typename _Traits > std::basic_istream<_CharT, _Traits>&
    operator>> ( std::basic_istream<_CharT, _Traits > & ,
    std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &
    ) [friend]

```

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

#### Parameters

`__is` The input stream.

`__lcr` A % [linear\\_congruential\\_engine](#) random number generator.

#### Returns

`__is`.

### 5.527.6 Member Data Documentation

```

5.527.6.1 template<typename _UIntType, _UIntType __a, _UIntType __c,
    _UIntType __m> const _UIntType std::linear_congruential_engine<
    _UIntType, __a, __c, __m >::increment [static]

```

An increment.

Definition at line 171 of file random.h.

```

5.527.6.2 template<typename _UIntType, _UIntType __a, _UIntType __c,
    _UIntType __m> const _UIntType std::linear_congruential_engine<
    _UIntType, __a, __c, __m >::modulus [static]

```

The modulus.

Definition at line 173 of file random.h.

```

5.527.6.3 template<typename _UIntType, _UIntType __a, _UIntType __c,
    _UIntType __m> const _UIntType std::linear_congruential_engine<
    _UIntType, __a, __c, __m >::multiplier [static]

```

The multiplier.

Definition at line 169 of file `random.h`.

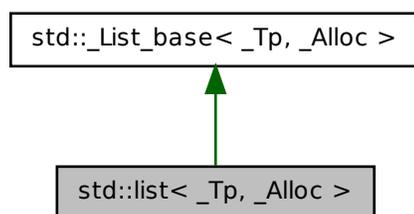
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.528 `std::list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::list< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_List_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [list](#) ()
- [list](#) (const allocator\_type &\_\_a)
- [list](#) (const [list](#) &\_\_x)
- [list](#) ([list](#) &&\_\_x)
- [list](#) (size\_type \_\_n, const value\_type &\_\_value=value\_type(), const allocator\_type &\_\_a=allocator\_type())
- [list](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[list](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [back](#) ()
- const\_reference [back](#) () const
- [const\\_iterator](#) [begin](#) () const
- [iterator](#) [begin](#) ()
- [const\\_iterator](#) [cbegin](#) () const
- [const\\_iterator](#) [cend](#) () const
- void [clear](#) ()
- [const\\_reverse\\_iterator](#) [crbegin](#) () const
- [const\\_reverse\\_iterator](#) [crend](#) () const
- template<typename... \_Args>  
[iterator](#) [emplace](#) ([iterator](#) \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- [iterator](#) [end](#) ()
- [const\\_iterator](#) [end](#) () const
- [iterator](#) [erase](#) ([iterator](#) \_\_position)
- [iterator](#) [erase](#) ([iterator](#) \_\_first, [iterator](#) \_\_last)
- const\_reference [front](#) () const
- reference [front](#) ()
- allocator\_type [get\\_allocator](#) () const
- void [insert](#) ([iterator](#) \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- [iterator](#) [insert](#) ([iterator](#) \_\_position, const value\_type &\_\_x)
- [iterator](#) [insert](#) ([iterator](#) \_\_position, value\_type &&\_\_x)

- void **insert** (iterator \_\_p, initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type **max\_size** () const
- void **merge** (list &\_\_x)
- void **merge** (list &&\_\_x)
- template<typename \_StrictWeakOrdering >  
void **merge** (list &\_\_x, \_StrictWeakOrdering \_\_comp)
- template<typename \_StrictWeakOrdering >  
void **merge** (list &&, \_StrictWeakOrdering)
- list & **operator=** (list &&\_\_x)
- list & **operator=** (initializer\_list< value\_type > \_\_l)
- list & **operator=** (const list &\_\_x)
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (value\_type &&\_\_x)
- void **push\_back** (const value\_type &\_\_x)
- void **push\_front** (const value\_type &\_\_x)
- void **push\_front** (value\_type &&\_\_x)
- **reverse\_iterator** **rbegin** ()
- **const\_reverse\_iterator** **rbegin** () const
- void **remove** (const \_Tp &\_\_value)
- template<typename \_Predicate >  
void **remove\_if** (\_Predicate)
- **const\_reverse\_iterator** **rend** () const
- **reverse\_iterator** **rend** ()
- void **resize** (size\_type \_\_new\_size, value\_type \_\_x=value\_type())
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering)
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_position, list &\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x)
- void **splice** (iterator \_\_position, list &&\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_position, list &\_\_x, iterator \_\_i)
- void **swap** (list &\_\_x)
- template<typename \_BinaryPredicate >  
void **unique** (\_BinaryPredicate)
- void **unique** ()

## Protected Types

- typedef [\\_List\\_node](#)< [\\_Tp](#) > [\\_Node](#)
- typedef [\\_Alloc](#)::template rebind< [\\_List\\_node](#)< [\\_Tp](#) > >::other [\\_Node\\_alloc\\_type](#)

## Protected Member Functions

- template<typename [\\_Integer](#) >  
void [\\_M\\_assign\\_dispatch](#) ([\\_Integer](#) \_\_n, [\\_Integer](#) \_\_val, [\\_true\\_type](#))
- template<typename [\\_InputIterator](#) >  
void [\\_M\\_assign\\_dispatch](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [\\_false\\_type](#))
- void [\\_M\\_check\\_equal\\_allocators](#) ([list](#) &\_\_x)
- void [\\_M\\_clear](#) ()
- template<typename... [\\_Args](#)>  
[\\_Node](#) \* [\\_M\\_create\\_node](#) ([\\_Args](#) &&...\_\_args)
- void [\\_M\\_erase](#) ([iterator](#) \_\_position)
- void [\\_M\\_fill\\_assign](#) ([size\\_type](#) \_\_n, const [value\\_type](#) &\_\_val)
- void [\\_M\\_fill\\_initialize](#) ([size\\_type](#) \_\_n, const [value\\_type](#) &\_\_x)
- [\\_List\\_node](#)< [\\_Tp](#) > \* [\\_M\\_get\\_node](#) ()
- const [\\_Node\\_alloc\\_type](#) & [\\_M\\_get\\_Node\\_allocator](#) () const
- [\\_Node\\_alloc\\_type](#) & [\\_M\\_get\\_Node\\_allocator](#) ()
- [\\_Tp\\_alloc\\_type](#) [\\_M\\_get\\_Tp\\_allocator](#) () const
- void [\\_M\\_init](#) ()
- template<typename [\\_Integer](#) >  
void [\\_M\\_initialize\\_dispatch](#) ([\\_Integer](#) \_\_n, [\\_Integer](#) \_\_x, [\\_true\\_type](#))
- template<typename [\\_InputIterator](#) >  
void [\\_M\\_initialize\\_dispatch](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, [\\_false\\_type](#))
- template<typename... [\\_Args](#)>  
void [\\_M\\_insert](#) ([iterator](#) \_\_position, [\\_Args](#) &&...\_\_args)
- void [\\_M\\_put\\_node](#) ([\\_List\\_node](#)< [\\_Tp](#) > \*\_\_p)
- void [\\_M\\_transfer](#) ([iterator](#) \_\_position, [iterator](#) \_\_first, [iterator](#) \_\_last)

## Protected Attributes

- [\\_List\\_impl](#) [\\_M\\_impl](#)

### 5.528.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::list<_Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <----> B <----> C <----> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 417 of file `stl_list.h`.

### 5.528.2 Constructor & Destructor Documentation

```
5.528.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list ( ) [inline]
```

Default constructor creates no elements.

Definition at line 499 of file `stl_list.h`.

```
5.528.2.2 template<typename Tp, typename _Alloc = std::allocator<Tp>>  
std::list<Tp, _Alloc>::list ( const allocator_type & _a )  
[inline, explicit]
```

Creates a list with no elements.

#### Parameters

*a* An allocator object.

Definition at line 507 of file `stl_list.h`.

```
5.528.2.3 template<typename Tp, typename _Alloc = std::allocator<Tp>>  
std::list<Tp, _Alloc>::list ( size_type _n, const value_type  
& _value = value_type() , const allocator_type & _a =  
allocator_type() ) [inline, explicit]
```

Creates a list with copies of an exemplar element.

#### Parameters

*n* The number of elements to initially create.

*value* An element to copy.

*a* An allocator object.

This constructor fills the list with *n* copies of *value*.

Definition at line 519 of file `stl_list.h`.

```
5.528.2.4 template<typename Tp, typename _Alloc = std::allocator<Tp>>  
std::list<Tp, _Alloc>::list ( const list<Tp, _Alloc> & _x )  
[inline]
```

List copy constructor.

#### Parameters

*x* A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *x*.

Definition at line 531 of file `stl_list.h`.

References `std::list< Tp, _Alloc>::begin()`, and `std::list< Tp, _Alloc>::end()`.

```
5.528.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list ( list<_Tp, _Alloc> && __x )
[inline]
```

List move constructor.

#### Parameters

*x* A list of identical element and allocator types.

The newly-created list contains the exact contents of *x*. The contents of *x* are a valid, but unspecified list.

Definition at line 543 of file stl\_list.h.

```
5.528.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list ( initializer_list<value_type> __l,
const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#) of value\_type.

*a* An allocator object.

Create a list consisting of copies of the elements in the [initializer\\_list](#) *l*. This is linear in *l.size()*.

Definition at line 554 of file stl\_list.h.

```
5.528.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator> std::list<_Tp, _Alloc>::list (
_InputIterator __first, _InputIterator __last, const allocator_type &
__a = allocator_type() ) [inline]
```

Builds a list from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*a* An allocator object.

Create a list consisting of copies of the elements from *[first,last)*. This is linear in N (where N is distance(*first,last*)).

Definition at line 571 of file stl\_list.h.

### 5.528.3 Member Function Documentation

**5.528.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename... _Args> _Node* std::list<_Tp, _Alloc  
>::_M_create_node ( _Args &&... __args ) [inline,  
protected]`

#### Parameters

*x* An instance of user data.

Allocates space for a new node and constructs a copy of *x* in it.

Definition at line 476 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc >::emplace()`, and `std::list<_Tp, _Alloc >::insert()`.

**5.528.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::assign ( size_type __n, const  
value_type & __val ) [inline]`

Assigns a given value to a list.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a list with *n* copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 642 of file `stl_list.h`.

**5.528.3.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > void std::list<_Tp, _Alloc  
>::assign ( _InputIterator __first, _InputIterator __last )  
[inline]`

Assigns a range to a list.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a list with copies of the elements in the range  $[first, last)$ .

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 659 of file `stl_list.h`.

```
5.528.3.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::assign ( initializer_list< value_type >  
    __l ) [inline]
```

Assigns an `initializer_list` to a list.

#### Parameters

*l* An `initializer_list` of `value_type`.

Replace the contents of the list with copies of the elements in the `initializer_list l`. This is linear in `l.size()`.

Definition at line 675 of file `stl_list.h`.

References `std::list<_Tp, _Alloc >::assign()`.

Referenced by `std::list<_Tp, _Alloc >::assign()`.

```
5.528.3.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::list<_Tp, _Alloc >::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the list.

Definition at line 848 of file `stl_list.h`.

```
5.528.3.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::list<_Tp, _Alloc >::back ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 860 of file `stl_list.h`.

```
5.528.3.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list<_Tp, _Alloc >::begin ( ) [inline]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 690 of file `stl_list.h`.

Referenced by `std::list< _Tp, _Alloc >::list()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::operator=()`, `std::operator==( )`, `std::list< _Tp, _Alloc >::remove_if()`, `std::list< _Tp, _Alloc >::resize()`, `std::list< _Tp, _Alloc >::sort()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::unique()`.

**5.528.3.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list< _Tp, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 699 of file `stl_list.h`.

**5.528.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list< _Tp, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 763 of file `stl_list.h`.

**5.528.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list< _Tp, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 772 of file `stl_list.h`.

**5.528.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1127 of file `stl_list.h`.

**5.528.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 781 of file `stl_list.h`.

**5.528.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list<_Tp, _Alloc >::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 790 of file `stl_list.h`.

**5.528.3.14** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> list<_Tp, _Alloc >::iterator list::emplace ( iterator  
_position, _Args &&... _args )`

Constructs object in list before specified iterator.

#### Parameters

*position* A `const_iterator` into the list.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.`

Definition at line 87 of file `list.tcc`.

References `std::list<_Tp, _Alloc >::_M_create_node()`.

**5.528.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::list<_Tp, _Alloc >::empty ( ) const [inline]`

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 800 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc >::sort()`, and `std::list<_Tp, _Alloc >::splice()`.

**5.528.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list<_Tp, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 708 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc >::list()`, `std::list<_Tp, _Alloc >::merge()`, `std::list<_Tp, _Alloc >::operator=()`, `std::operator==( )`, `std::list<_Tp, _Alloc >::remove_if()`, `std::list<_Tp, _Alloc >::resize()`, `std::list<_Tp, _Alloc >::splice()`, and `std::list<_Tp, _Alloc >::unique()`.

**5.528.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list<_Tp, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 717 of file `stl_list.h`.

**5.528.3.18** `template<typename _Tp , typename _Alloc > list<_Tp, _Alloc  
>::iterator list::erase ( iterator __position )`

Remove element at given position.

#### Parameters

*position* Iterator pointing to element to be erased.

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 108 of file `list.tcc`.

Referenced by `std::list<_Tp, _Alloc >::operator=()`, and `std::list<_Tp, _Alloc >::resize()`.

**5.528.3.19** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list<_Tp, _Alloc >::erase ( iterator __first, iterator  
__last ) [inline]`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1093 of file stl\_list.h.

**5.528.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::list<_Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 832 of file stl\_list.h.

**5.528.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::list<_Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 840 of file stl\_list.h.

**5.528.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
allocator_type std::list<_Tp, _Alloc >::get_allocator ( ) const  
[inline]`

Get a copy of the memory allocation object.

Reimplemented from `std::_List_base<_Tp, _Alloc >`.

Definition at line 681 of file stl\_list.h.

**5.528.3.23** `template<typename _Tp , typename _Alloc > list< _Tp, _Alloc >::iterator list::insert ( iterator __position, const value_type & __x )`

Inserts given value into list before specified iterator.

#### Parameters

*position* An iterator into the list.  
*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 98 of file list.tcc.

References `std::list< _Tp, _Alloc >::_M_create_node()`.

Referenced by `std::list< _Tp, _Alloc >::operator=()`, and `std::list< _Tp, _Alloc >::resize()`.

**5.528.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into list before specified iterator.

#### Parameters

*position* An iterator into the list.  
*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 994 of file stl\_list.h.

**5.528.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::insert ( iterator __p, initializer_list<  
value_type > __l ) [inline]`

Inserts the contents of an [initializer\\_list](#) into list before specified iterator.

#### Parameters

- p* An iterator into the list.
- l* An [initializer\\_list](#) of value\_type.

This function will insert copies of the data in the [initializer\\_list](#) *l* into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1011 of file stl\_list.h.

References `std::list< _Tp, _Alloc >::insert()`.

Referenced by `std::list< _Tp, _Alloc >::insert()`.

**5.528.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::insert ( iterator __position, size_type  
__n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the list.

#### Parameters

- position* An iterator into the list.
- n* Number of elements to be inserted.
- x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1028 of file stl\_list.h.

**5.528.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > void std::list< _Tp,  
_Alloc >::insert ( iterator __position, _InputIterator __first,  
_InputIterator __last ) [inline]`

Inserts a range into the list.

**Parameters**

*position* An iterator into the list.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range [*first*,*last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1049 of file `stl_list.h`.

**5.528.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::list<_Tp, _Alloc>::max_size( ) const [inline]`

Returns the `size()` of the largest possible list.

Definition at line 810 of file `stl_list.h`.

**5.528.3.29** `template<typename _Tp, typename _Alloc > template<typename  
_StrictWeakOrdering > void list::merge ( list<_Tp, _Alloc > &&  
__x, _StrictWeakOrdering __comp )`

Merge sorted lists according to comparison function.

**Parameters**

*x* Sorted list to merge.

*StrictWeakOrdering* Comparison function defining sort order.

Assumes that both *x* and this list are sorted according to `StrictWeakOrdering`. Merges elements of *x* into this list in sorted order, leaving *x* empty when complete. Elements in this list precede elements in *x* that are equivalent according to `StrictWeakOrdering()`.

Definition at line 270 of file `list.tcc`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

**5.528.3.30** `template<typename _Tp, typename _Alloc > void list::merge ( list<_Tp, _Alloc > && __x )`

Merge sorted lists.

**Parameters**

*x* Sorted list to merge.

Assumes that both  $x$  and this list are sorted according to operator<(). Merges elements of  $x$  into this list in sorted order, leaving  $x$  empty when complete. Elements in this list precede elements in  $x$  that are equal.

Definition at line 236 of file list.tcc.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

Referenced by `std::list<_Tp, _Alloc >::sort()`.

### 5.528.3.31 `template<typename _Tp, typename _Alloc > list<_Tp, _Alloc > & list::operator= ( const list<_Tp, _Alloc > & __x )`

List assignment operator.

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### Parameters

$x$  A list of identical element and allocator types.

All the elements of  $x$  are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 133 of file list.tcc.

References `std::list<_Tp, _Alloc >::begin()`, `std::list<_Tp, _Alloc >::end()`, `std::list<_Tp, _Alloc >::erase()`, and `std::list<_Tp, _Alloc >::insert()`.

### 5.528.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc >::operator= ( initializer_list<value_type > __l ) [inline]`

List initializer list assignment operator.

#### Parameters

$l$  An [initializer\\_list](#) of `value_type`.

Replace the contents of the list with copies of the elements in the [initializer\\_list](#)  $l$ . This is linear in `l.size()`.

Definition at line 624 of file `stl_list.h`.

```
5.528.3.33 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
list& std::list<_Tp, _Alloc >::operator= ( list<_Tp, _Alloc > &&  
_x ) [inline]
```

List move assignment operator.

#### Parameters

*x* A list of identical element and allocator types.

The contents of *x* are moved into this list (without copying). *x* is a valid, but unspecified list

Definition at line 607 of file `stl_list.h`.

References `std::swap()`.

```
5.528.3.34 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::pop_back ( ) [inline]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 946 of file `stl_list.h`.

```
5.528.3.35 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::pop_front ( ) [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 906 of file `stl_list.h`.

**5.528.3.36** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::push_back ( const value_type & __x  
) [inline]`

Add data to the end of the list.

#### Parameters

**x** Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 920 of file stl\_list.h.

**5.528.3.37** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::push_front ( const value_type & __x  
) [inline]`

Add data to the front of the list.

#### Parameters

**x** Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 879 of file stl\_list.h.

**5.528.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::list<_Tp, _Alloc >::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 726 of file stl\_list.h.

**5.528.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list<_Tp, _Alloc >::rbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 735 of file stl\_list.h.

**5.528.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc >::remove ( const _Tp & __value )`

Remove all elements equal to *value*.

**Parameters**

*value* The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.528.3.41** `template<typename _Tp, typename _Alloc > template<typename  
_Predicate > void list::remove_if ( _Predicate __pred )`

Remove all elements satisfying a predicate.

**Parameters**

*Predicate* Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 340 of file list.tcc.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

**5.528.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::list<_Tp, _Alloc >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 744 of file stl\_list.h.

**5.528.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list<_Tp, _Alloc >::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 753 of file stl\_list.h.

**5.528.3.44** `template<typename _Tp, typename _Alloc > void list::resize ( size_type __new_size, value_type __x = value_type() )`

Resizes the list to the specified number of elements.

#### Parameters

*new\_size* Number of elements the list should contain.

*x* Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 118 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, `std::list< _Tp, _Alloc >::erase()`, and `std::list< _Tp, _Alloc >::insert()`.

**5.528.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::reverse ( ) [inline]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1347 of file stl\_list.h.

**5.528.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::size ( ) const [inline]`

Returns the number of elements in the list.

Definition at line 805 of file stl\_list.h.

References `__gnu_cxx::distance()`.

**5.528.3.47** `template<typename _Tp, typename _Alloc > template<typename _StrictWeakOrdering > void list::sort ( _StrictWeakOrdering __comp )`

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 379 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

### 5.528.3.48 `template<typename _Tp, typename _Alloc > void list::sort ( )`

Sort the elements.

Sorts the elements of this list in  $N \log N$  time. Equivalent elements remain in list order.

Definition at line 302 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

### 5.528.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice ( iterator __position, list< _Tp, _Alloc > && __x, iterator __i ) [inline]`

Insert element from another list.

#### Parameters

*position* Iterator referencing the element to insert before.

*x* Source list.

*i* Iterator referencing the element to move.

Removes the element in list *x* referenced by *i* and inserts it into the current list before *position*.

Definition at line 1177 of file stl\_list.h.

### 5.528.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice ( iterator __position, list< _Tp, _Alloc > && __x, iterator __first, iterator __last ) [inline]`

Insert range from another list.

#### Parameters

*position* Iterator referencing the element to insert before.

*x* Source list.

*first* Iterator referencing the start of range in *x*.

*last* Iterator referencing the end of range in *x*.

Removes elements in the range [first,last) and inserts them before *position* in constant time.

Undefined if *position* is in [first,last).

Definition at line 1213 of file stl\_list.h.

```
5.528.3.51 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice ( iterator __position, list<
_Tp, _Alloc > && __x ) [inline]
```

Insert contents of another list.

#### Parameters

*position* Iterator referencing the element to insert before.

*x* Source list.

The elements of *x* are inserted in constant time in front of the element referenced by *position*. *x* becomes an empty list.

Requires this != *x*.

Definition at line 1147 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::begin(), std::list< \_Tp, \_Alloc >::empty(), and std::list< \_Tp, \_Alloc >::end().

Referenced by std::list< \_Tp, \_Alloc >::sort().

```
5.528.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap ( list< _Tp, _Alloc > & __x )
[inline]
```

Swaps data with another list.

#### Parameters

*x* A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global [std::swap\(\)](#) function is specialized such that std::swap(l1,l2) will feed to this function.

Definition at line 1110 of file stl\_list.h.

References std::swap().

Referenced by std::list< \_Tp, \_Alloc >::sort(), and std::swap().

**5.528.3.53** `template<typename _Tp , typename _Alloc > template<typename  
_BinaryPredicate > void list::unique ( _BinaryPredicate  
_binary_pred )`

Remove consecutive elements satisfying a predicate.

#### Parameters

*BinaryPredicate* Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 358 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

**5.528.3.54** `template<typename _Tp , typename _Alloc > void list::unique ( )`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 215 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.529 std::locale Class Reference

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

## Classes

- class [facet](#)

*Localization functionality base class.*

*The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*

- class [id](#)

*Facet ID class.*

*The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.*

## Public Types

- typedef int [category](#)

## Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &\_\_other) throw ()
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- [locale](#) (const char \*\_\_s)
- template<typename \_Facet >  
[locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)
- [~locale](#) () throw ()
- template<typename \_Facet >  
[locale combine](#) (const [locale](#) &\_\_other) const
- [string name](#) () const
- bool [operator!=](#) (const [locale](#) &\_\_other) const throw ()
- template<typename \_Char , typename \_Traits , typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s2) const
- template<typename \_CharT , typename \_Traits , typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &\_\_other) throw ()
- bool [operator==](#) (const [locale](#) &\_\_other) const throw ()

## Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale](#) [global](#) (const [locale](#) &)

## Static Public Attributes

- static const [category](#) [none](#)
- static const [category](#) [ctype](#)
- static const [category](#) [numeric](#)
- static const [category](#) [collate](#)
- static const [category](#) [time](#)
- static const [category](#) [monetary](#)
- static const [category](#) [messages](#)
- static const [category](#) [all](#)

## Friends

- struct [\\_\\_use\\_cache](#)
- class [\\_Impl](#)
- class [facet](#)
- template<typename [\\_Facet](#) >  
bool [has\\_facet](#) (const [locale](#) &) throw ()
- template<typename [\\_Facet](#) >  
const [\\_Facet](#) & [use\\_facet](#) (const [locale](#) &)

### 5.529.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file [locale\\_classes.h](#).

### 5.529.2 Member Typedef Documentation

#### 5.529.2.1 typedef int std::locale::category

Definition of [locale::category](#).

Definition at line 67 of file locale\_classes.h.

### 5.529.3 Constructor & Destructor Documentation

#### 5.529.3.1 std::locale::locale ( ) throw ()

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by combine().

#### 5.529.3.2 std::locale::locale ( const locale & \_\_other ) throw ()

Copy constructor.

Constructs a copy of *other*.

#### Parameters

*other* The locale to copy.

#### 5.529.3.3 std::locale::locale ( const char \* \_\_s ) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

*s* Name of the locale to construct.

#### Exceptions

[\*std::runtime\\_error\*](#) if *s* is null or an undefined locale.

#### 5.529.3.4 std::locale::locale ( const locale & \_\_base, const char \* \_\_s, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

**Parameters**

- base* The locale to copy.
- s* Name of the locale to use facets from.
- cat* Set of categories defining the facets to use from *s*.

**Exceptions**

- std::runtime\_error* if *s* is null or an undefined locale.

### 5.529.3.5 `std::locale::locale ( const locale & __base, const locale & __add, category __cat )`

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

**Parameters**

- base* The locale to copy.
- add* The locale to use facets from.
- cat* Set of categories defining the facets to use from *add*.

### 5.529.3.6 `template<typename _Facet > std::locale::locale ( const locale & __other, _Facet * __f )`

Construct locale with another facet.

Constructs a copy of the locale *other*. The facet is added to , replacing an existing facet of type *Facet* if there is one. If *f* is null, this locale is a copy of *other*.

**Parameters**

- other* The locale to copy.
- f* The facet to add in.

Definition at line 43 of file locale\_classes.tcc.

### 5.529.3.7 `std::locale::~~locale ( ) throw ()`

Locale destructor.

## 5.529.4 Member Function Documentation

### 5.529.4.1 static const locale& std::locale::classic ( ) [static]

Return reference to the C locale.

### 5.529.4.2 template<typename \_Facet > locale std::locale::combine ( const locale & *\_\_other* ) const

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type *Facet* from the locale *other* into the new locale.

#### Parameters

*Facet* The facet type to copy from *other*

*other* The locale to copy from.

#### Returns

Newly constructed locale.

#### Exceptions

[\*std::runtime\\_error\*](#) if *other* has no facet of type *Facet*.

Definition at line 61 of file locale\_classes.tcc.

References locale().

### 5.529.4.3 static locale std::locale::global ( const locale & ) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

#### Parameters

*locale* The new locale to make global.

#### Returns

Copy of the old global locale.

**5.529.4.4** `string std::locale::name ( ) const`

Return locale name.

**Returns**

Locale name or "\*" if unnamed.

**5.529.4.5** `bool std::locale::operator!=( const locale & __other ) const throw ()`  
[**inline**]

Locale inequality.

**Parameters**

*other* The locale to compare against.

**Returns**

! (\*this == other)

Definition at line 234 of file locale\_classes.h.

**5.529.4.6** `template<typename _Char , typename _Traits , typename _Alloc >`  
`bool std::locale::operator() ( const basic_string< _Char, _Traits,`  
`_Alloc > & __s1, const basic_string< _Char, _Traits, _Alloc > &`  
`__s2 ) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

**Parameters**

*s1* First string to compare.

*s2* Second string to compare.

**Returns**

True if collate<Char> facet compares *s1* < *s2*, else false.

**5.529.4.7** `const locale& std::locale::operator= ( const locale & __other ) throw ()`

Assignment operator.

Set this locale to be a copy of *other*.

#### Parameters

*other* The locale to copy.

#### Returns

A reference to this locale.

**5.529.4.8** `bool std::locale::operator== ( const locale & __other ) const throw ()`

Locale equality.

#### Parameters

*other* The locale to compare against.

#### Returns

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

### 5.529.5 Friends And Related Function Documentation

**5.529.5.1** `template<typename _Facet > bool has_facet ( const locale & )  
throw () [friend]`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

#### Parameters

*Facet* The facet type to test the presence of.

*locale* The locale to test.

#### Returns

true if locale contains a facet of type *Facet*, else false.

**5.529.5.2** `template<typename _Facet > const _Facet& use_facet ( const locale & ) [friend]`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

**Parameters**

*Facet* The facet type to access.

*locale* The locale to use.

**Returns**

Reference to facet of type `Facet`.

**Exceptions**

`std::bad_cast` if locale doesn't contain a facet of type `Facet`.

**5.529.6 Member Data Documentation****5.529.6.1** `const category std::locale::all [static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

**5.529.6.2** `const category std::locale::collate [static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

**5.529.6.3 const category std::locale::ctype [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

**5.529.6.4 const category std::locale::messages [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

**5.529.6.5 const category std::locale::monetary [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

**5.529.6.6 const category std::locale::none [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

**5.529.6.7** `const category std::locale::numeric` [`static`]

Category values.

The standard category values are `none`, `ctype`, `numeric`, `collate`, `time`, `monetary`, and `messages`. They form a bitmask that supports union and intersection. The category `all` is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

**5.529.6.8** `const category std::locale::time` [`static`]

Category values.

The standard category values are `none`, `ctype`, `numeric`, `collate`, `time`, `monetary`, and `messages`. They form a bitmask that supports union and intersection. The category `all` is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

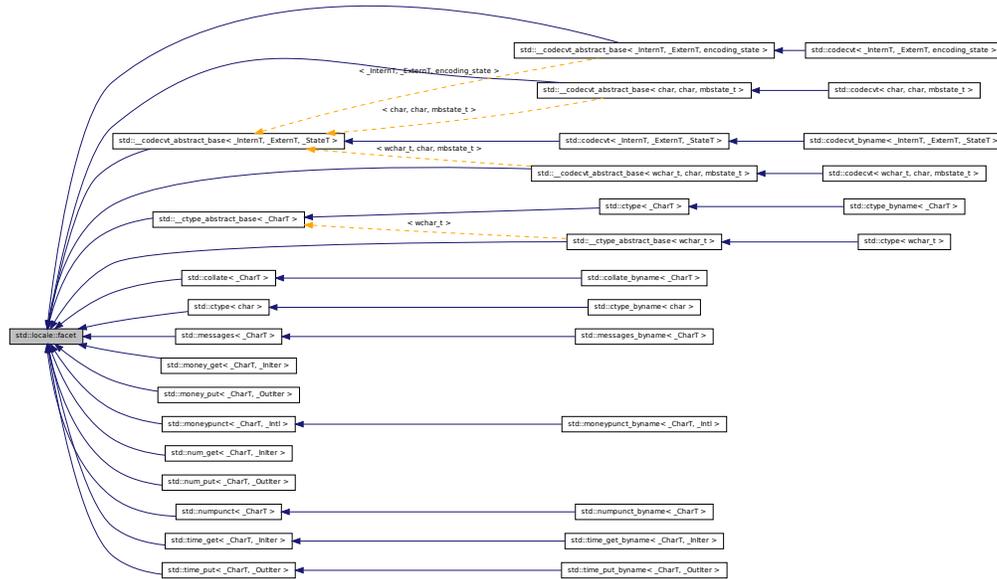
- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

**5.530** `std::locale::facet` Class Reference

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Inheritance diagram for std::locale::facet:



## Protected Member Functions

- `facet` (size\_t \_\_refs=0) throw ()
- virtual `~facet` ()

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Friends

- class `locale`
- class `locale::_Impl`

### 5.530.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management. Facets may not be copied or assigned.

Definition at line 337 of file `locale_classes.h`.

### 5.530.2 Constructor & Destructor Documentation

#### 5.530.2.1 `std::locale::facet::facet ( size_t __refs = 0 ) throw () [inline, explicit, protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

#### Parameters

*refs* The initial value for reference count.

Definition at line 369 of file `locale_classes.h`.

#### 5.530.2.2 `virtual std::locale::facet::~~facet ( ) [protected, virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.531 `std::locale::id` Class Reference

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

## Public Member Functions

- [id](#) ()
- `size_t _M_id` () const throw ()

## Friends

- `template<typename _Facet >`  
`bool has_facet` (const [locale](#) &) throw ()
- class [locale](#)
- class [locale::\\_Impl](#)
- `template<typename _Facet >`  
`const _Facet & use_facet` (const [locale](#) &)

### 5.531.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Definition at line 432 of file `locale_classes.h`.

### 5.531.2 Constructor & Destructor Documentation

#### 5.531.2.1 `std::locale::id::id` ( ) [[inline](#)]

Constructor.

Definition at line 463 of file `locale_classes.h`.

### 5.531.3 Friends And Related Function Documentation

#### 5.531.3.1 `template<typename _Facet > bool has_facet` ( const [locale](#) & ) `throw` () [[friend](#)]

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Parameters**

*Facet* The facet type to test the presence of.

*locale* The locale to test.

**Returns**

true if locale contains a facet of type Facet, else false.

**5.531.3.2 `template<typename _Facet > const _Facet& use_facet ( const locale & ) [friend]`**

Return a facet.

`use_facet` looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws [std::bad\\_cast](#) if the locale doesn't contain a facet of type Facet.

**Parameters**

*Facet* The facet type to access.

*locale* The locale to use.

**Returns**

Reference to facet of type Facet.

**Exceptions**

[std::bad\\_cast](#) if locale doesn't contain a facet of type Facet.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

**5.532 `std::lock_guard< _Mutex > Class Template Reference`**

Scoped lock idiom.

**Public Types**

- `typedef _Mutex mutex_type`

## Public Member Functions

- `lock_guard` (mutex\_type &\_m)
- `lock_guard` (mutex\_type &\_m, `adopt_lock_t`)
- `lock_guard` (const `lock_guard` &)
- `lock_guard` & `operator=` (const `lock_guard` &)

### 5.532.1 Detailed Description

```
template<typename _Mutex> class std::lock_guard< _Mutex >
```

Scoped lock idiom.

Definition at line 393 of file mutex.

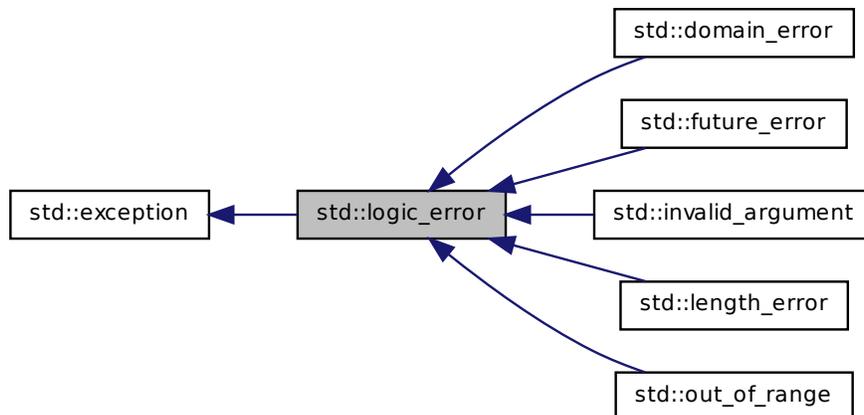
The documentation for this class was generated from the following file:

- [mutex](#)

## 5.533 std::logic\_error Class Reference

One of two subclasses of exception.

Inheritance diagram for std::logic\_error:



## Public Member Functions

- [logic\\_error](#) (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const throw ()

### 5.533.1 Detailed Description

One of two subclasses of exception. Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 53 of file `stdexcept`.

### 5.533.2 Constructor & Destructor Documentation

#### 5.533.2.1 `std::logic_error::logic_error ( const string & __arg ) [explicit]`

Takes a character string describing the error.

### 5.533.3 Member Function Documentation

#### 5.533.3.1 `virtual const char* std::logic_error::what ( ) const throw () [virtual]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

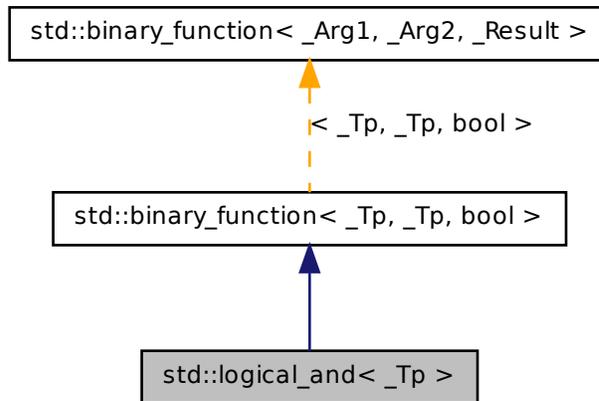
The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.534 `std::logical_and< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for `std::logical_and<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.534.1 Detailed Description

```
template<typename _Tp> struct std::logical_and<_Tp>
```

One of the [Boolean operations functors](#).

Definition at line 263 of file `stl_function.h`.

## 5.534.2 Member Typedef Documentation

**5.534.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.534.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.534.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

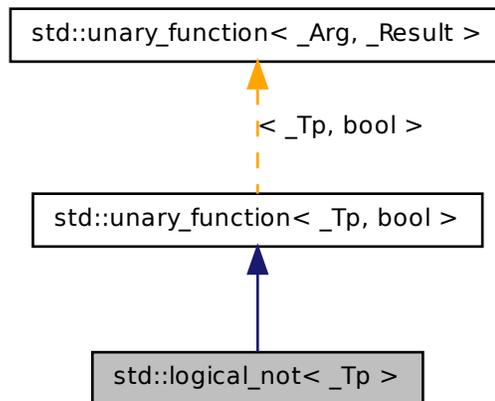
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.535 `std::logical_not< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical\_not< \_Tp >:



## Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- `bool operator() (const _Tp &__x) const`

### 5.535.1 Detailed Description

`template<typename _Tp> struct std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 281 of file `stl_function.h`.

## 5.535.2 Member Typedef Documentation

### 5.535.2.1 `typedef _Tp std::unary_function< _Tp , bool >::argument_type` [`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.535.2.2 `typedef bool std::unary_function< _Tp , bool >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

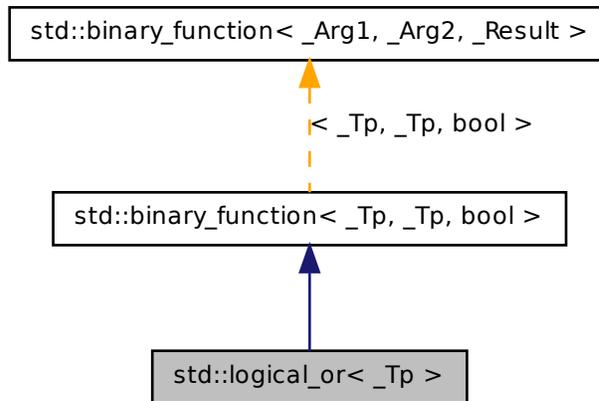
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.536 `std::logical_or< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical\_or< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.536.1 Detailed Description

```
template<typename _Tp> struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 272 of file `stl_function.h`.

## 5.536.2 Member Typedef Documentation

**5.536.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.536.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.536.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.537 `std::lognormal_distribution< _RealType >` Class Template Reference

A [lognormal\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## 5.537 `std::lognormal_distribution<_RealType>` Class Template Reference 2577

### Public Member Functions

- `lognormal_distribution` (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- `lognormal_distribution` (const `param_type` &\_\_p)
- `_RealType m` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- void `param` (const `param_type` &\_\_param)
- `param_type param` () const
- void `reset` ()
- `_RealType s` () const

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits > & operator<<` (`std::basic_ostream<_CharT, _Traits > &, const std::lognormal_distribution<_RealType1 > &`)
- `template<typename _RealType1 > bool operator==` (const `std::lognormal_distribution<_RealType1 > &__d1, const std::lognormal_distribution<_RealType1 > &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits > & operator>>` (`std::basic_istream<_CharT, _Traits > &, std::lognormal_distribution<_RealType1 > &`)

### 5.537.1 Detailed Description

`template<typename _RealType = double> class std::lognormal_distribution<_RealType >`

A `lognormal_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2174 of file `random.h`.

## 5.537.2 Member Typedef Documentation

### 5.537.2.1 `template<typename _RealType = double> typedef _RealType std::lognormal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2181 of file random.h.

## 5.537.3 Member Function Documentation

### 5.537.3.1 `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2265 of file random.h.

### 5.537.3.2 `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2258 of file random.h.

### 5.537.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::lognormal_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2273 of file random.h.

References `std::lognormal_distribution< _RealType >::operator()`, and `std::lognormal_distribution< _RealType >::param()`.

Referenced by `std::lognormal_distribution< _RealType >::operator()`.

### 5.537.3.4 `template<typename _RealType = double> void std::lognormal_distribution< _RealType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

## 5.537 std::lognormal\_distribution<\_RealType> Class Template Reference 2579

### Parameters

*\_\_param* The new parameter set of the distribution.

Definition at line 2251 of file random.h.

```
5.537.3.5 template<typename _RealType = double> param_type  
std::lognormal_distribution<_RealType>::param ( ) const  
[inline]
```

Returns the parameter set of the distribution.

Definition at line 2243 of file random.h.

Referenced by std::lognormal\_distribution<\_RealType>::operator()().

```
5.537.3.6 template<typename _RealType = double> void  
std::lognormal_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2225 of file random.h.

References std::normal\_distribution<\_RealType>::reset().

## 5.537.4 Friends And Related Function Documentation

```
5.537.4.1 template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::lognormal_distribution<_RealType1> & ) [friend]
```

Inserts a lognormal\_distribution random number distribution *\_\_x* into the output stream *\_\_os*.

### Parameters

*\_\_os* An output stream.

*\_\_x* A lognormal\_distribution random number distribution.

### Returns

The output stream with the state of *\_\_x* inserted or in an error state.

**5.537.4.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::lognormal_distribution<  
_RealType1 > & __d1, const std::lognormal_distribution<  
_RealType1 > & __d2 ) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2289 of file random.h.

**5.537.4.3** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream< _CharT, _Traits > & ,  
std::lognormal_distribution< _RealType1 > & ) [friend]`

Extracts a lognormal\_distribution random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A lognormal\_distribution random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.538 `std::lognormal_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef `lognormal_distribution< _RealType > distribution_type`

### Public Member Functions

- `param_type` (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- `_RealType m` () const
- `_RealType s` () const

## Friends

- `bool operator==(const param\_type &__p1, const param\_type &__p2)`

### 5.538.1 Detailed Description

`template<typename \_RealType = double> struct std::lognormal\_distribution<  
\_RealType >::param\_type`

Parameter type.

Definition at line 2183 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.539 `std::make_signed< _Tp >` Struct Template Reference

[make\\_signed](#)

### Public Types

- `typedef \_\_make\_signed\_selector< \_Tp >::\_\_type type`

### 5.539.1 Detailed Description

`template<typename \_Tp> struct std::make\_signed< \_Tp >`

[make\\_signed](#)

Definition at line 587 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.540 `std::make_unsigned< _Tp >` Struct Template Reference

[make\\_unsigned](#)

## Public Types

- typedef `__make_unsigned_selector< _Tp >::__type` **type**

### 5.540.1 Detailed Description

`template<typename _Tp> struct std::make_unsigned< _Tp >`

[make\\_unsigned](#)

Definition at line 510 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.541 `std::map< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const\_pointer**
- typedef `_Pair_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

**Public Member Functions**

- `map ()`
- `map (const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `map (map &&__x)`
- `map (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())`
- `map (const map &__x)`
- `template<typename _InputIterator > map (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator > map (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k) const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `size_type count (const key_type &__x) const`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `iterator erase (iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `allocator_type get_allocator () const`
- `template<typename _InputIterator > void insert (_InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, const value_type &__x)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `key_compare key_comp () const`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`

- size\_type `max_size` () const
- `map` & `operator=` (`map` &&\_\_x)
- `map` & `operator=` (const `map` &\_\_x)
- `map` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- mapped\_type & `operator[]` (const key\_type &\_\_k)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- size\_type `size` () const
- void `swap` (`map` &\_\_x)
- iterator `upper_bound` (const key\_type &\_\_x)
- const\_iterator `upper_bound` (const key\_type &\_\_x) const
- value\_compare `value_comp` () const

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**< (const `map`< \_K1, \_T1, \_C1, \_A1 > &, const `map`< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**== (const `map`< \_K1, \_T1, \_C1, \_A1 > &, const `map`< \_K1, \_T1, \_C1, \_A1 > &)

### 5.541.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class
std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_map.h`.

## 5.541.2 Constructor & Destructor Documentation

**5.541.2.1** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map ( ) [inline]`

Default constructor creates no elements.

Definition at line 150 of file stl\_map.h.

**5.541.2.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map ( const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline, explicit]`

Creates a map with no elements.

### Parameters

*comp* A comparison object.

*a* An allocator object.

Definition at line 159 of file stl\_map.h.

**5.541.2.3** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map ( const map< _Key, _Tp, _Compare, _Alloc > & __x ) [inline]`

Map copy constructor.

### Parameters

*x* A map of identical element and allocator types.

The newly-created map uses a copy of the allocation object used by *x*.

Definition at line 170 of file stl\_map.h.

**5.541.2.4** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( map<_Key, _Tp, _Compare, _Alloc> && __x ) [inline]`

Map move constructor.

#### Parameters

*x* A map of identical element and allocator types.

The newly-created map contains the exact contents of *x*. The contents of *x* are a valid, but unspecified map.

Definition at line 181 of file `stl_map.h`.

**5.541.2.5** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( initializer_list<value_type> __l, const _Compare & __c = _Compare(), const allocator_type & __a = allocator_type() ) [inline]`

Builds a map from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#).

*comp* A comparison object.

*a* An allocator object.

Create a map consisting of copies of the elements in the [initializer\\_list](#) *l*. This is linear in *N* if the range is already sorted, and *N*log*N* otherwise (where *N* is *l.size()*).

Definition at line 195 of file `stl_map.h`.

**5.541.2.6** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::map<_Key, _Tp, _Compare, _Alloc>::map ( _InputIterator __first, _InputIterator __last ) [inline]`

Builds a map from a range.

#### Parameters

*first* An input iterator.

## 5.541 `std::map<_Key, _Tp, _Compare, _Alloc >` Class Template Reference 2587

*last* An input iterator.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 212 of file stl\_map.h.

```
5.541.2.7 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator > std::map<
_Key, _Tp, _Compare, _Alloc >::map ( _InputIterator __first,
_InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from a range.

### Parameters

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 228 of file stl\_map.h.

## 5.541.3 Member Function Documentation

```
5.541.3.1 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare,
_Alloc >::at ( const key_type & __k ) [inline]
```

Access to map data.

### Parameters

*k* The key for which data should be retrieved.

### Returns

A reference to the data whose key is equivalent to *k*, if such a data is present in the map.

**Exceptions**

*std::out\_of\_range* If no such data is present.

Definition at line 465 of file stl\_map.h.

**5.541.3.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 307 of file stl\_map.h.

**5.541.3.3** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 316 of file stl\_map.h.

**5.541.3.4** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 380 of file stl\_map.h.

**5.541.3.5** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 389 of file stl\_map.h.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2589

**5.541.3.6** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc >::clear ( ) [inline]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 663 of file stl\_map.h.

**5.541.3.7** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::count ( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

### Parameters

*x* Key of (key, value) pairs to be located.

### Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 723 of file stl\_map.h.

**5.541.3.8** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 398 of file stl\_map.h.

**5.541.3.9** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc >::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 407 of file `stl_map.h`.

**5.541.3.10** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> bool std::map<_Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline]`

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 416 of file `stl_map.h`.

**5.541.3.11** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 334 of file `stl_map.h`.

**5.541.3.12** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 325 of file `stl_map.h`.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2591

**5.541.3.13** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::map<_Key, _Tp, _Compare, _Alloc >::equal_range ( const key_type & __x )`  
`[inline]`

Finds a subsequence matching given key.

### Parameters

**x** Key of (key, value) pairs to be located.

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 792 of file stl\_map.h.

**5.541.3.14** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::map<_Key, _Tp, _Compare, _Alloc >::equal_range ( const key_type & __x ) const` `[inline]`

Finds a subsequence matching given key.

### Parameters

**x** Key of (key, value) pairs to be located.

### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 811 of file `stl_map.h`.

```
5.541.3.15 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc
>::erase ( iterator __position ) [inline]
```

Erases an element from a map.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 573 of file `stl_map.h`.

```
5.541.3.16 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> size_type std::map<_Key, _Tp, _Compare, _Alloc
>::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

#### Parameters

*x* Key of element to be erased.

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 603 of file `stl_map.h`.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2593

**5.541.3.17** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::erase( iterator __first, iterator __last ) [inline]`

Erases a [first,last) range of elements from a map.

### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

### Returns

The iterator *last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 622 of file stl\_map.h.

**5.541.3.18** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::find( const key_type & __x ) const [inline]`

Tries to locate an element in a map.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 711 of file stl\_map.h.

**5.541.3.19** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::find ( const key_type & __x ) [inline]`

Tries to locate an element in a map.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 696 of file `stl_map.h`.

**5.541.3.20** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::map<_Key, _Tp, _Compare, _Alloc>::get_allocator ( ) const [inline]`

Get a copy of the memory allocation object.

Definition at line 297 of file `stl_map.h`.

**5.541.3.21** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>::insert ( const value_type & __x ) [inline]`

Attempts to insert a `std::pair` into the map.

#### Parameters

*x* Pair to be inserted (see `std::make_pair` for easy creation of pairs).

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2595

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 500 of file stl\_map.h.

```
5.541.3.22 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc  
>::insert ( std::initializer_list<value_type> > __list ) [inline]
```

Attempts to insert a list of std::pairs into the map.

### Parameters

*list* A std::initializer\_list<value\_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 512 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

```
5.541.3.23 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc  
>::insert ( iterator __position, const value_type & __x )  
[inline]
```

Attempts to insert a [std::pair](#) into the map.

### Parameters

*position* An iterator that serves as a hint as to where the pair should be inserted.

*x* Pair to be inserted (see std::make\_pair for easy creation of pairs).

### Returns

An iterator that points to the element with key of *x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument [insert\(\)](#) does. Note that the first parameter

is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 540 of file `stl_map.h`.

```
5.541.3.24 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> template<typename _InputIterator > void
std::map< _Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator
__first, _InputIterator __last ) [inline]
```

Template function that attempts to insert a range of elements.

#### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 553 of file `stl_map.h`.

```
5.541.3.25 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> key_compare std::map< _Key, _Tp, _Compare,
_Alloc >::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 672 of file `stl_map.h`.

```
5.541.3.26 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2597

### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 738 of file `stl_map.h`.

```
5.541.3.27 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::lower_bound ( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 753 of file `stl_map.h`.

```
5.541.3.28 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc  
>::max_size ( ) const [inline]
```

Returns the maximum size of the map.

Definition at line 426 of file `stl_map.h`.

```
5.541.3.29 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc  
>::operator= ( map< _Key, _Tp, _Compare, _Alloc > && __x )  
[inline]
```

Map move assignment operator.

**Parameters**

*x* A map of identical element and allocator types.

The contents of *x* are moved into this map (without copying). *x* is a valid, but unspecified map.

Definition at line 266 of file `stl_map.h`.

References `std::swap()`.

```
5.541.3.30 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> map& std::map<_Key, _Tp, _Compare, _Alloc
>::operator= ( const map<_Key, _Tp, _Compare, _Alloc > & __x
) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**Parameters**

*x* A map of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 251 of file `stl_map.h`.

```
5.541.3.31 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> map& std::map<_Key, _Tp, _Compare, _Alloc
>::operator= ( initializer_list<value_type > __l ) [inline]
```

Map list assignment operator.

**Parameters**

*l* An [initializer\\_list](#).

This function fills a map with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 287 of file `stl_map.h`.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2599

**5.541.3.32** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare, _Alloc>::operator[] ( const key_type & _k ) [inline]`

Subscript ( [] ) access to map data.

### Parameters

*k* The key for which data should be retrieved.

### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 443 of file stl\_map.h.

**5.541.3.33** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 343 of file stl\_map.h.

**5.541.3.34** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 352 of file stl\_map.h.

**5.541.3.35** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 370 of file `stl_map.h`.

**5.541.3.36** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 361 of file `stl_map.h`.

**5.541.3.37** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc>::size ( ) const [inline]`

Returns the size of the map.

Definition at line 421 of file `stl_map.h`.

**5.541.3.38** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc>::swap ( map<_Key, _Tp, _Compare, _Alloc> & __x ) [inline]`

Swaps data with another map.

#### Parameters

**x** A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

## 5.541 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2601

Definition at line 653 of file stl\_map.h.

Referenced by std::swap().

```
5.541.3.39 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::upper_bound ( const key_type & __x ) const  
[inline]
```

Finds the end of a subsequence matching given key.

### Parameters

**x** Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 773 of file stl\_map.h.

```
5.541.3.40 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

### Parameters

**x** Key of (key, value) pair to be located.

### Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 763 of file stl\_map.h.

```
5.541.3.41 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> value_compare std::map< _Key, _Tp, _Compare,  
_Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 680 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 5.542 `std::mask_array< _Tp >` Class Template Reference

Reference to selected subset of an array.

### Public Types

- `typedef _Tp value_type`

### Public Member Functions

- [mask\\_array](#) (const [mask\\_array](#) &)
- `template<class _Dom >`  
void **operator%=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- void [operator%=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- void [operator&=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator&=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator\*%=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- void [operator\\*%=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator\*+=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- void [operator\\*+=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- void [operator\\*-=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator\*-=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator/=%** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- void [operator/=%](#) (const [valarray](#)< [\\_Tp](#) > &) const
- void [operator<<=](#) (const [valarray](#)< [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator<<=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const
- `template<class _Dom >`  
void **operator/=** (const [\\_Expr](#)< [\\_Dom](#), [\\_Tp](#) > &) const

- `mask_array` & `operator=` (const `mask_array` &)
- void `operator=` (const `valarray<_Tp>` &) const
- template<class `_Ex`>  
void `operator=` (const `_Expr<_Ex, _Tp>` & `_e`) const
- void `operator=` (const `_Tp` &) const
- void `operator>>=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator>>=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator^=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator^=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator|=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator|=` (const `_Expr<_Dom, _Tp>` &) const

## Friends

- class `valarray<_Tp>`

### 5.542.1 Detailed Description

template<class `_Tp`> class `std::mask_array<_Tp>`

Reference to selected subset of an array. A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of `bool`. The way to get a `mask_array` is to call `operator[]`(`valarray<bool>`) on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array (false, true, false, true) as an argument, the `mask_array` has two elements referring to `array[1]` and `array[3]` in the underlying array.

#### Parameters

*`Tp`* Element type.

Definition at line 61 of file `mask_array.h`.

### 5.542.2 Member Function Documentation

5.542.2.1 template<class `_Tp`> void `std::mask_array<_Tp>::operator%=` (const `valarray<_Tp>` & ) const

Modulo slice elements by corresponding elements of `v`.

**5.542.2.2** `template<class _Tp> void std::mask_array< _Tp >::operator&= ( const valarray< _Tp > & ) const`

Logical and slice elements with corresponding elements of *v*.

**5.542.2.3** `template<class _Tp> void std::mask_array< _Tp >::operator*= ( const valarray< _Tp > & ) const`

Multiply slice elements by corresponding elements of *v*.

**5.542.2.4** `template<class _Tp> void std::mask_array< _Tp >::operator+= ( const valarray< _Tp > & ) const`

Add corresponding elements of *v* to slice elements.

**5.542.2.5** `template<class _Tp> void std::mask_array< _Tp >::operator-= ( const valarray< _Tp > & ) const`

Subtract corresponding elements of *v* from slice elements.

**5.542.2.6** `template<class _Tp> void std::mask_array< _Tp >::operator/= ( const valarray< _Tp > & ) const`

Divide slice elements by corresponding elements of *v*.

**5.542.2.7** `template<class _Tp> void std::mask_array< _Tp >::operator<<= ( const valarray< _Tp > & ) const`

Left shift slice elements by corresponding elements of *v*.

**5.542.2.8** `template<class _Tp> void std::mask_array< _Tp >::operator>>= ( const valarray< _Tp > & ) const`

Right shift slice elements by corresponding elements of *v*.

**5.542.2.9** `template<class _Tp> void std::mask_array< _Tp >::operator^= ( const valarray< _Tp > & ) const`

Logical xor slice elements with corresponding elements of *v*.

---

## 5.543 `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>` Class Template Reference 2605

5.542.2.10 `template<class _Tp> void std::mask_array<_Tp>::operator|= (const valarray<_Tp> & ) const`

Logical or slice elements with corresponding elements of `v`.

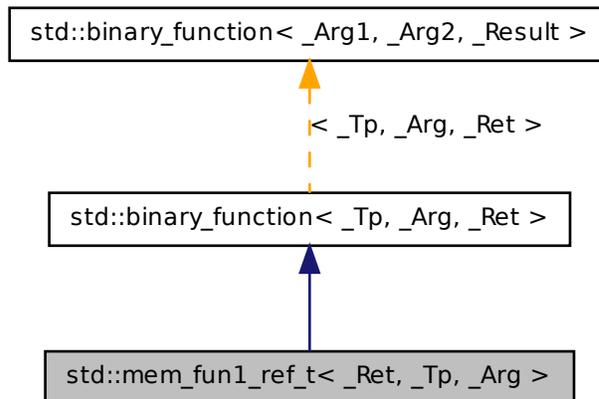
The documentation for this class was generated from the following file:

- [mask\\_array.h](#)

## 5.543 `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- `mem_fun1_ref_t(_Ret(_Tp::*__pf)(_Arg))`
- `_Ret operator()(_Tp &__r, _Arg __x) const`

### 5.543.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`

One of the [adaptors for member /// pointers](#).

Definition at line 632 of file `stl_function.h`.

### 5.543.2 Member Typedef Documentation

**5.543.2.1** `typedef _Tp std::binary_function<_Tp, _Arg, _Ret >::first_argument_type` [*inherited*]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.543.2.2** `typedef _Ret std::binary_function<_Tp, _Arg, _Ret >::result_type` [*inherited*]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.543.2.3** `typedef _Arg std::binary_function<_Tp, _Arg, _Ret >::second_argument_type` [*inherited*]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

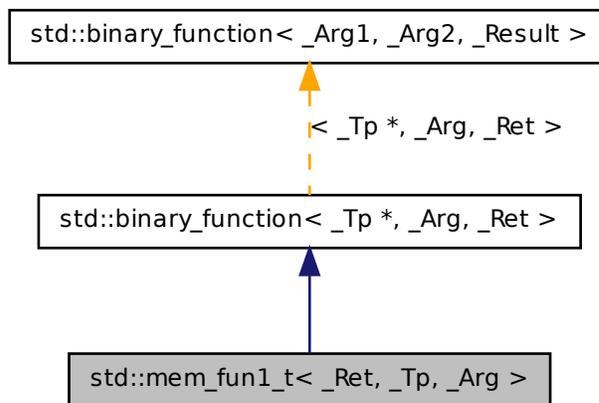
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.544 `std::mem_fun1_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun1_t< _Ret, _Tp, _Arg >`:



### Public Types

- typedef `_Tp *` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `mem_fun1_t` (`_Ret`(`_Tp::*_pf`)(`_Arg`))
- `_Ret operator()` (`_Tp *_p`, `_Arg __x`) const

### 5.544.1 Detailed Description

**template<typename \_Ret, typename \_Tp, typename \_Arg> class std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >**

One of the [adaptors for member /// pointers](#).

Definition at line 596 of file stl\_function.h.

### 5.544.2 Member Typedef Documentation

**5.544.2.1 typedef \_Tp \* std::binary\_function< \_Tp \* , \_Arg , \_Ret >::first\_argument\_type [inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl\_function.h.

**5.544.2.2 typedef \_Ret std::binary\_function< \_Tp \* , \_Arg , \_Ret >::result\_type [inherited]**

type of the return type

Definition at line 118 of file stl\_function.h.

**5.544.2.3 typedef \_Arg std::binary\_function< \_Tp \* , \_Arg , \_Ret >::second\_argument\_type [inherited]**

the type of the second argument

Definition at line 117 of file stl\_function.h.

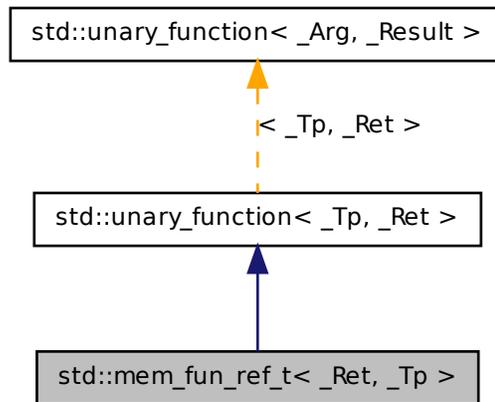
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.545 std::mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem\_fun\_ref\_t< \_Ret, \_Tp >:



## Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

## Public Member Functions

- `mem_fun_ref_t` (`_Ret`(`_Tp`::\*`__pf`)())
- `_Ret operator()` (`_Tp` &`__r`) const

### 5.545.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member /// pointers](#).

Definition at line 560 of file `stl_function.h`.

## 5.545.2 Member Typedef Documentation

### 5.545.2.1 `typedef _Tp std::unary_function< _Tp , _Ret >::argument_type` [`inherited`]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.545.2.2 `typedef _Ret std::unary_function< _Tp , _Ret >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

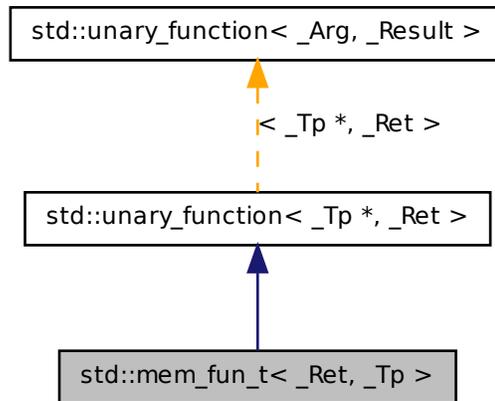
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.546 `std::mem_fun_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member `///` pointers](#).

Inheritance diagram for std::mem\_fun\_t< \_Ret, \_Tp >:



## Public Types

- typedef `_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

## Public Member Functions

- `mem_fun_t` (`_Ret`(`_Tp::*_pf`())
- `_Ret operator()` (`_Tp *_p`) const

### 5.546.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::mem_fun_t< _Ret, _Tp >`

One of the [adaptors for member /// pointers](#).

Definition at line 524 of file `stl_function.h`.

## 5.546.2 Member Typedef Documentation

### 5.546.2.1 `typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.546.2.2 `typedef _Ret std::unary_function< _Tp *, _Ret >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

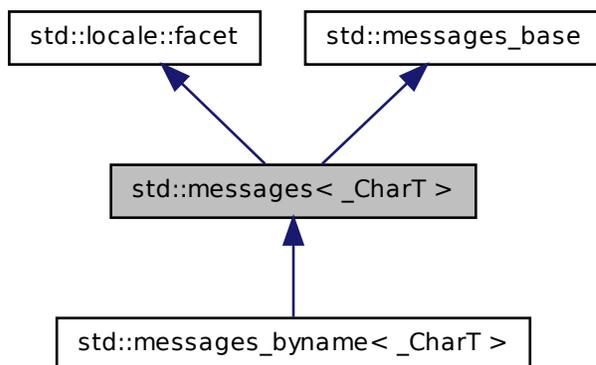
- [stl\\_function.h](#)

## 5.547 `std::messages< _CharT >` Class Template Reference

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

Inheritance diagram for std::messages<\_CharT>:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

## Public Member Functions

- **messages** (size\_t \_\_refs=0)
- **messages** (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- **string\_type** **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const **string\_type** &\_\_s) const
- catalog **open** (const `basic_string<char>` &\_\_s, const **locale** &\_\_loc) const
- catalog **open** (const `basic_string<char>` &, const **locale** &, const char \*) const

## Static Public Attributes

- static **locale::id** id

## Protected Member Functions

- virtual `~messages ()`
- `string_type _M_convert_from_char (char *) const`
- `char * _M_convert_to_char (const string_type &__msg) const`
- virtual void `do_close (catalog) const`
- `template<> string do_get (catalog, int, int, const string &) const`
- `template<> wstring do_get (catalog, int, int, const wstring &) const`
- virtual `string_type do_get (catalog, int, int, const string_type &__default) const`
- virtual catalog `do_open (const basic_string< char > &, const locale &) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_messages`
- `const char * _M_name_messages`

## Friends

- class `locale::_Impl`

### 5.547.1 Detailed Description

`template<typename _CharT> class std::messages< _CharT >`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined. This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around `gettext`, provided by `libintl`. The

second version (ieec) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for `char` and `wchar_t` instantiations.

The `messages` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `messages` facet.

Definition at line 1689 of file `locale_facets_nonio.h`.

## 5.547.2 Member Typedef Documentation

### 5.547.2.1 `template<typename _CharT > typedef _CharT std::messages<_CharT >::char_type`

Public typedefs.

Reimplemented in [std::messages\\_byname<\\_CharT >](#).

Definition at line 1695 of file `locale_facets_nonio.h`.

### 5.547.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::messages<_CharT >::string_type`

Public typedefs.

Reimplemented in [std::messages\\_byname<\\_CharT >](#).

Definition at line 1696 of file `locale_facets_nonio.h`.

## 5.547.3 Constructor & Destructor Documentation

### 5.547.3.1 `template<typename _CharT > std::messages<_CharT >::messages ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 43 of file `messages_members.h`.

**5.547.3.2** `template<typename _CharT > std::messages< _CharT >::messages`  
( `_c_locale __cloc`, `const char * __s`, `size_t __refs = 0` )  
[**explicit**]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

*cloc* The C locale.

*s* The name of a locale.

*refs* Refcount to pass to the base class.

Definition at line 49 of file `messages_members.h`.

**5.547.3.3** `template<typename _CharT > std::messages< _CharT`  
>::~messages( ) [**protected**, **virtual**]

Destructor.

Definition at line 78 of file `messages_members.h`.

## 5.547.4 Member Data Documentation

**5.547.4.1** `template<typename _CharT > locale::id std::messages< _CharT`  
>::id [**static**]

Numpunct facet id.

Definition at line 1707 of file `locale_facets_nonio.h`.

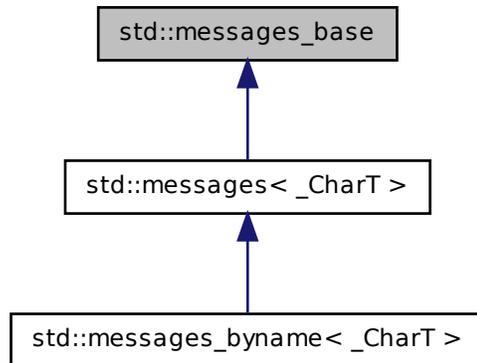
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.548 std::messages\_base Struct Reference

Messages facet base class providing catalog typedef.

Inheritance diagram for `std::messages_base`:



## Public Types

- typedef int **catalog**

### 5.548.1 Detailed Description

Messages facet base class providing catalog typedef.

Definition at line 1662 of file `locale_facets_nonio.h`.

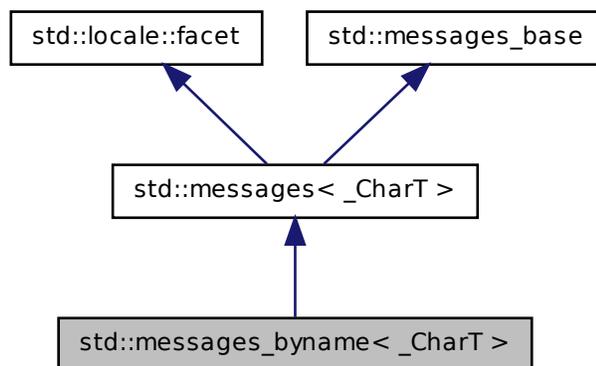
The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.549 `std::messages_byname<_CharT>` Class Template Reference

class `messages_byname` [22.2.7.2].

Inheritance diagram for `std::messages_byname<_CharT>`:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

## Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- **string\_type get** (catalog \_\_c, int \_\_set, int \_\_msgid, const **string\_type** &\_\_s) const
- catalog **open** (const **basic\_string**< char > &, const **locale** &, const char \*) const
- catalog **open** (const **basic\_string**< char > &\_\_s, const **locale** &\_\_loc) const

## Static Public Attributes

- static **locale::id** **id**

## Protected Member Functions

- [string\\_type](#) `_M_convert_from_char` (char \*) const
- char \* `_M_convert_to_char` (const [string\\_type](#) &\_\_msg) const
- virtual void `do_close` (catalog) const
- template<>  
[wstring](#) `do_get` (catalog, int, int, const [wstring](#) &) const
- template<>  
[string](#) `do_get` (catalog, int, int, const [string](#) &) const
- virtual [string\\_type](#) `do_get` (catalog, int, int, const [string\\_type](#) &\_\_default) const
- virtual catalog `do_open` (const [basic\\_string](#)<char> &, const [locale](#) &) const

## Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` &\_\_cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` &\_\_cloc, const char \*\_\_s, `__c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`__c_locale` &\_\_cloc)
- static `__c_locale` `_S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale` `_M_c_locale_messages`
- const char \* `_M_name_messages`

## Friends

- class `locale::Impl`

### 5.549.1 Detailed Description

`template<typename _CharT> class std::messages_byname<_CharT>`

class [messages\\_byname](#) [22.2.7.2].

Definition at line 1906 of file `locale_facets_nonio.h`.

## 5.549.2 Member Typedef Documentation

### 5.549.2.1 `template<typename _CharT > typedef _CharT std::messages_byname<_CharT >::char_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT >](#).

Definition at line 1909 of file `locale_facets_nonio.h`.

### 5.549.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::messages_byname<_CharT >::string_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT >](#).

Definition at line 1910 of file `locale_facets_nonio.h`.

## 5.549.3 Member Data Documentation

### 5.549.3.1 `template<typename _CharT > locale::id std::messages<_CharT >::id [static, inherited]`

Numpunct facet id.

Definition at line 1707 of file `locale_facets_nonio.h`.

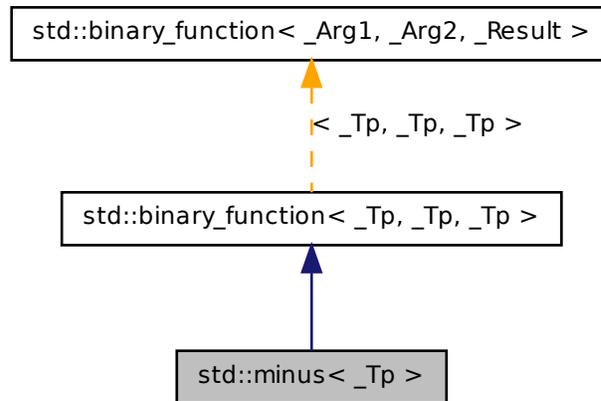
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.550 `std::minus<_Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::minus< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

### 5.550.1 Detailed Description

```
template<typename _Tp> struct std::minus< _Tp >
```

One of the [math functors](#).

Definition at line 144 of file `stl_function.h`.

## 5.550.2 Member Typedef Documentation

### 5.550.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

### 5.550.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

### 5.550.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

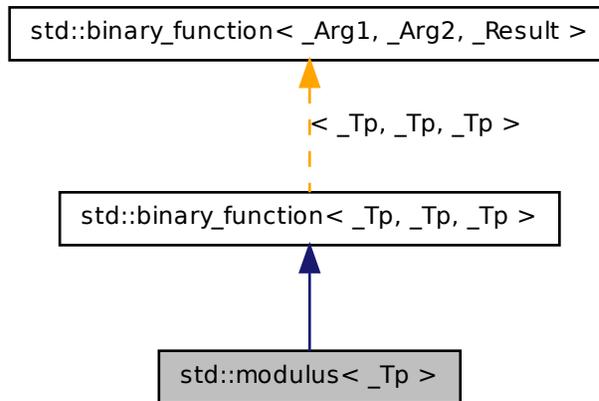
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.551 `std::modulus< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::modulus< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

### 5.551.1 Detailed Description

```
template<typename _Tp> struct std::modulus< _Tp >
```

One of the [math functors](#).

Definition at line 171 of file `stl_function.h`.

## 5.551.2 Member Typedef Documentation

### 5.551.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

### 5.551.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

### 5.551.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

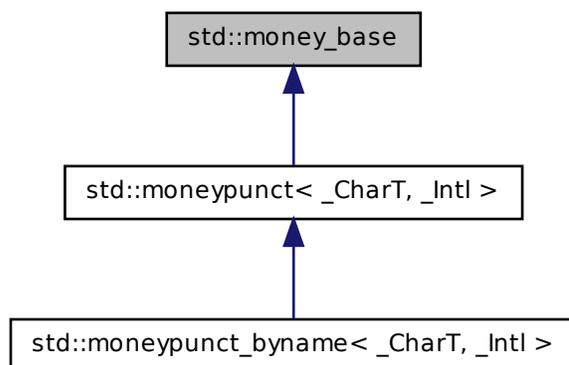
- [stl\\_function.h](#)

## 5.552 `std::money_base` Class Reference

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

Inheritance diagram for std::money\_base:



## Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }

## Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

### 5.552.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

**See also**

[moneypunct::pos\\_format\(\)](#) and [moneypunct::neg\\_format\(\)](#) for details of how these fields are interpreted.

Definition at line 835 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

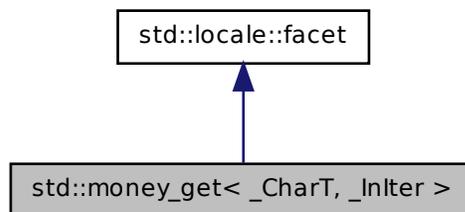
- [locale\\_facets\\_nonio.h](#)

## 5.553 `std::money_get< _CharT, _InIter >` Class Template Reference

Primary class template [money\\_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string.

Inheritance diagram for `std::money_get< _CharT, _InIter >`:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string< \\_CharT >](#) [string\\_type](#)

## Public Member Functions

- `money_get` (`size_t __refs=0`)
- `iter_type get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits`) const
- `iter_type get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units`) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`  
`iter_type _M_extract` (`iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits`) const
- virtual `iter_type do_get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits`) const
- virtual `iter_type do_get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units`) const

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (`_c_locale &__cloc`) throw ()
- static void `_S_create_c_locale` (`_c_locale &__cloc, const char *__s, _c_locale __old=0`)
- static void `_S_destroy_c_locale` (`_c_locale &__cloc`)
- static `_c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char \* `_S_get_c_name` () throw ()
- static `_c_locale _S_lc_ctype_c_locale` (`_c_locale __cloc, const char *__s`)

## Friends

- class `locale::_Impl`

### 5.553.1 Detailed Description

**template<typename \_CharT, typename \_InIter> class std::money\_get< \_CharT, \_InIter >**

Primary class template [money\\_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string. The [money\\_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [money\\_get](#) facet.

Definition at line 1364 of file locale\_facets\_nonio.h.

### 5.553.2 Member Typedef Documentation

**5.553.2.1 template<typename \_CharT , typename \_InIter > typedef \_CharT std::money\_get< \_CharT, \_InIter >::char\_type**

Public typedefs.

Definition at line 1370 of file locale\_facets\_nonio.h.

**5.553.2.2 template<typename \_CharT , typename \_InIter > typedef \_InIter std::money\_get< \_CharT, \_InIter >::iter\_type**

Public typedefs.

Definition at line 1371 of file locale\_facets\_nonio.h.

**5.553.2.3 template<typename \_CharT , typename \_InIter > typedef basic\_string<\_CharT> std::money\_get< \_CharT, \_InIter >::string\_type**

Public typedefs.

Definition at line 1372 of file locale\_facets\_nonio.h.

### 5.553.3 Constructor & Destructor Documentation

**5.553.3.1** `template<typename _CharT , typename _InIter > std::money_get< _CharT, _InIter >::money_get ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 1386 of file locale\_facets\_nonio.h.

**5.553.3.2** `template<typename _CharT , typename _InIter > virtual std::money_get< _CharT, _InIter >::~~money_get ( ) [inline, protected, virtual]`

Destructor.

Definition at line 1454 of file locale\_facets\_nonio.h.

### 5.553.4 Member Function Documentation

**5.553.4.1** `template<typename _CharT , typename _InIter > _InIter std::money_get< _CharT, _InIter >::do_get ( iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units ) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for details.

Definition at line 363 of file locale\_facets\_nonio.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`.

Referenced by `std::money_get< _CharT, _InIter >::get()`.

**5.553.4.2** `template<typename _CharT , typename _InIter > _InIter  
std::money_get< _CharT, _InIter >::do_get ( iter_type __s,  
iter_type __end, bool __intl, ios_base & __io, ios_base::iostate &  
__err, string_type & __digits ) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for details.

Definition at line 376 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, and `std::_ctype_abstract_base< _CharT >::widen()`.

**5.553.4.3** `template<typename _CharT , typename _InIter > iter_type  
std::money_get< _CharT, _InIter >::get ( iter_type __s, iter_type  
__end, bool __intl, ios_base & __io, ios_base::iostate & __err,  
string_type & __digits ) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

#### Parameters

*s* Start of characters to parse.

*end* End of characters to parse.

*intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.

*io* Source of facets and io state.

*err* Error field to set if parsing fails.

*digits* Place to store result of parsing.

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1447 of file locale\_facets\_nonio.h.

References std::money\_get< \_CharT, \_InIter >::do\_get().

```
5.553.4.4 template<typename _CharT , typename _InIter > iter_type  
std::money_get< _CharT, _InIter >::get ( iter_type __s, iter_type  
__end, bool __intl, ios_base & __io, ios_base::iostate & __err, long  
double & __units ) const [inline]
```

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value `moneypunct::frac_digits()` \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets *err*=(err|io.failbit). If the stream is consumed before finishing parsing, sets *err*=(err|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

#### Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use\_facet<moneypunct<CharT,intl> >.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- units* Place to store result of parsing.

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1416 of file locale\_facets\_nonio.h.

References std::money\_get< \_CharT, \_InIter >::do\_get().

### 5.553.5 Member Data Documentation

```
5.553.5.1 template<typename _CharT , typename _InIter > locale::id  
std::money_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

Definition at line 1376 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

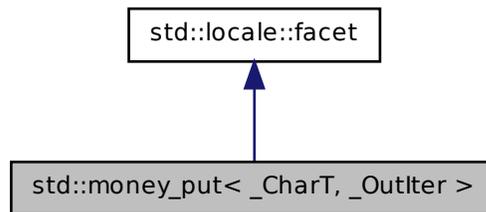
- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.554 `std::money_put< _CharT, _OutIter >` Class Template Reference

Primary class template [money\\_put](#).

This facet encapsulates the code to format and output a monetary amount.

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)
- typedef [basic\\_string< \\_CharT >](#) [string\\_type](#)

### Public Member Functions

- [money\\_put](#) (size\_t \_\_refs=0)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, bool \_\_intl, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const [string\\_type](#) &\_\_digits) const
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, bool \_\_intl, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, long double \_\_units) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_put()`
- `template<bool _Intl>`  
`iter_type _M_insert(iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const`
- virtual `iter_type do_put(iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const`
- virtual `iter_type do_put(iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const`

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale(_c_locale &__cloc) throw()`
- static `void _S_create_c_locale(_c_locale &__cloc, const char *__s, _c_locale __old=0)`
- static `void _S_destroy_c_locale(_c_locale &__cloc)`
- static `_c_locale _S_get_c_locale()`
- static `_GLIBCXX_CONST const char * _S_get_c_name() throw()`
- static `_c_locale _S_lc_ctype_c_locale(_c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.554.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::money_put<_CharT, _OutIter >`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount. The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1515 of file `locale_facets_nonio.h`.

## 5.554.2 Member Typedef Documentation

**5.554.2.1** `template<typename _CharT, typename _OutIter > typedef _CharT  
std::money_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 1520 of file locale\_facets\_nonio.h.

**5.554.2.2** `template<typename _CharT, typename _OutIter > typedef _OutIter  
std::money_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 1521 of file locale\_facets\_nonio.h.

**5.554.2.3** `template<typename _CharT, typename _OutIter > typedef  
basic_string<_CharT> std::money_put< _CharT, _OutIter  
>::string_type`

Public typedefs.

Definition at line 1522 of file locale\_facets\_nonio.h.

## 5.554.3 Constructor & Destructor Documentation

**5.554.3.1** `template<typename _CharT, typename _OutIter >  
std::money_put< _CharT, _OutIter >::money_put ( size_t __refs =  
0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

### Parameters

*refs* Passed to the base facet class.

Definition at line 1536 of file locale\_facets\_nonio.h.

**5.554.3.2** `template<typename _CharT, typename _OutIter > virtual  
std::money_put< _CharT, _OutIter >::~~money_put ( )  
[inline, protected, virtual]`

Destructor.

Definition at line 1586 of file `locale_facets_nonio.h`.

## 5.554.4 Member Function Documentation

**5.554.4.1** `template<typename _CharT, typename _OutIter > _OutIter  
std::money_put<_CharT, _OutIter >::do_put ( iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, long double __units )  
const [protected, virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function is a hook for derived classes to change the value returned.

### See also

[put\(\)](#).

### Parameters

*s* The stream to write to.

*intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.

*io* Source of facets and io state.

*fill* `char_type` to use for padding.

*units* Place to store result of parsing.

### Returns

Iterator after writing.

Definition at line 568 of file `locale_facets_nonio.tcc`.

References `std::ios_base::getloc()`, and `std::__ctype_abstract_base<_CharT >::widen()`.

Referenced by `std::money_put<_CharT, _OutIter >::put()`.

**5.554.4.2** `template<typename _CharT, typename _OutIter > _OutIter  
std::money_put<_CharT, _OutIter >::do_put ( iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, const string_type &  
__digits )const [protected, virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function is a hook for derived classes to change the value returned.

### See also

[put\(\)](#).

### Parameters

*s* The stream to write to.

*intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.

*io* Source of facets and io state.

*fill* `char_type` to use for padding.

*units* Place to store result of parsing.

### Returns

Iterator after writing.

Definition at line 606 of file `locale_facets_nonio.tcc`.

```
5.554.4.3 template<typename _CharT , typename _OutIter > iter_type
std::money_put< _CharT, _OutIter >::put ( iter_type __s, bool
__intl, ios_base & __io, char_type __fill, const string_type &
__digits ) const [inline]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function works by returning the result of [do\\_put\(\)](#).

### Parameters

*s* The stream to write to.

*intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.

*io* Source of facets and io state.

*fill* `char_type` to use for padding.

*units* Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 1579 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _OutIter >::do_put()`.

```
5.554.4.4 template<typename _CharT , typename _OutIter > iter_type
std::money_put<_CharT, _OutIter >::put ( iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units )
const [inline]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function works by returning the result of `do_put()`.

**Parameters**

*s* The stream to write to.

*intl* Parameter to use `_facet<moneypunct<CharT,intl> >`.

*io* Source of facets and io state.

*fill* `char_type` to use for padding.

*units* Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 1556 of file `locale_facets_nonio.h`.

References `std::money_put<_CharT, _OutIter >::do_put()`.

**5.554.5 Member Data Documentation**

```
5.554.5.1 template<typename _CharT , typename _OutIter > locale::id
std::money_put<_CharT, _OutIter >::id [static]
```

Numpunct facet id.

Definition at line 1526 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

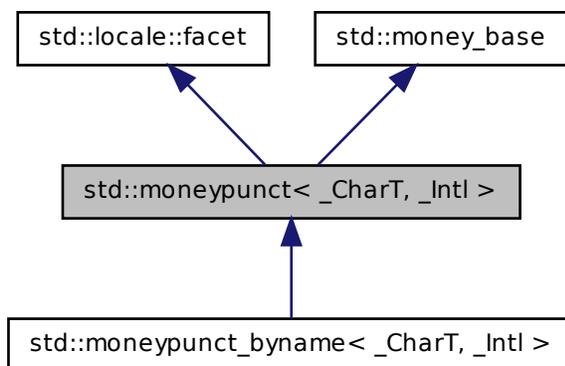
- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.555 `std::moneypunct< _CharT, _Intl >` Class Template Reference

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Inheritance diagram for `std::moneypunct< _CharT, _Intl >`:



### Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache< _CharT, _Intl >` `__cache_type`
- enum **part** {  
     `none`, `space`, `symbol`, `sign`,  
     `value` }
- typedef `_CharT` `char_type`
- typedef `basic_string< _CharT >` `string_type`

## Public Member Functions

- [moneypunct](#) (size\_t \_\_refs=0)
  - [moneypunct](#) (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
  - [moneypunct](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
  - [string\\_type curr\\_symbol](#) () const
  - [char\\_type decimal\\_point](#) () const
  - [int frac\\_digits](#) () const
  - [string grouping](#) () const
  - [string\\_type negative\\_sign](#) () const
  - [string\\_type positive\\_sign](#) () const
  - [char\\_type thousands\\_sep](#) () const
- pattern [pos\\_format](#) () const
  - pattern [neg\\_format](#) () const

## Static Public Member Functions

- static \_GLIBCXX\_CONST pattern [\\_S\\_construct\\_pattern](#) (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* [\\_S\\_atoms](#)
- static const pattern [\\_S\\_default\\_pattern](#)
- static [locale::id](#) [id](#)
- static const bool [intl](#)

## Protected Member Functions

- virtual [~moneypunct](#) ()
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale \_\_cloc=NULL, const char \*\_\_name=NULL)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- template<>  
void [\\_M\\_initialize\\_moneypunct](#) (\_\_c\_locale, const char \*)
- virtual [string\\_type do\\_curr\\_symbol](#) () const

- virtual `char_type do_decimal_point ()` const
- virtual `int do_frac_digits ()` const
- virtual `string do_grouping ()` const
- virtual `pattern do_neg_format ()` const
- virtual `string_type do_negative_sign ()` const
- virtual `pattern do_pos_format ()` const
- virtual `string_type do_positive_sign ()` const
- virtual `char_type do_thousands_sep ()` const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

### 5.555.1 Detailed Description

```
template<typename _CharT, bool _Intl> class std::moneypunct< _CharT, _Intl >
```

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 929 of file `locale_facets_nonio.h`.

### 5.555.2 Member Typedef Documentation

**5.555.2.1** `template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type`

Public typedefs.

Reimplemented in `std::moneypunct_byname< _CharT, _Intl >`.

Definition at line 935 of file `locale_facets_nonio.h`.

```
5.555.2.2 template<typename _CharT, bool _Intl> typedef
    basic_string<_CharT> std::moneypunct< _CharT, _Intl
    >::string_type
```

Public typedefs.

Reimplemented in [std::moneypunct\\_byname< \\_CharT, \\_Intl >](#).

Definition at line 936 of file locale\_facets\_nonio.h.

### 5.555.3 Constructor & Destructor Documentation

```
5.555.3.1 template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
    _Intl >::moneypunct ( size_t __refs = 0 ) [inline, explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 958 of file locale\_facets\_nonio.h.

```
5.555.3.2 template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
    _Intl >::moneypunct ( __cache_type * __cache, size_t __refs = 0 )
    [inline, explicit]
```

Constructor performs initialization.

This is an internal constructor.

#### Parameters

*cache* Cache for optimization.

*refs* Passed to the base facet class.

Definition at line 970 of file locale\_facets\_nonio.h.

```
5.555.3.3 template<typename _CharT, bool _Intl> std::moneypunct< _CharT,
    _Intl >::moneypunct ( __c_locale __cloc, const char * __s, size_t
    __refs = 0 ) [inline, explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

- oloc* The C locale.
- s* The name of a locale.
- refs* Passed to the base facet class.

Definition at line 985 of file locale\_facets\_nonio.h.

**5.555.3.4** `template<typename _CharT, bool _Intl> virtual std::moneypunct<_CharT, _Intl >::~~moneypunct ( ) [protected, virtual]`

Destructor.

**5.555.4 Member Function Documentation**

**5.555.4.1** `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl >::curr_symbol ( ) const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1055 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_curr_symbol()`.

**5.555.4.2** `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl >::decimal_point ( ) const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 999 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_decimal_point()`.

**5.555.4.3** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const  
[inline, protected, virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[curr\\_symbol\(\)](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1201 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::curr_symbol()`.

**5.555.4.4** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct< _CharT, _Intl >::do_decimal_point ( ) const  
[inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1163 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

**5.555.4.5** `template<typename _CharT, bool _Intl> virtual int  
std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const  
[inline, protected, virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[frac\\_digits\(\)](#) for details.

**Returns**

Number of digits in amount fraction.

Definition at line 1241 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::frac_digits()`.

**5.555.4.6** `template<typename _CharT, bool _Intl> virtual string  
std::moneypunct<_CharT, _Intl >::do_grouping ( ) const  
[inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1188 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::grouping()`.

**5.555.4.7** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct<_CharT, _Intl >::do_neg_format ( ) const  
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[neg\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1269 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::neg_format()`.

**5.555.4.8** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const  
[inline, protected, virtual]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

[negative\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1227 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

**5.555.4.9** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const  
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[pos\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1255 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

**5.555.4.10** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const  
[inline, protected, virtual]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

[positive\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1214 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::positive_sign()`.

**5.555.4.11** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl >::do_thousands_sep ( ) const  
[inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1175 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

**5.555.4.12** `template<typename _CharT, bool _Intl> int std::moneypunct<  
_CharT, _Intl >::frac_digits ( ) const [inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1105 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

**5.555.4.13** `template<typename _CharT, bool _Intl> string std::moneypunct< _CharT, _Intl >::grouping ( ) const [inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `and` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1042 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

**5.555.4.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct< _CharT, _Intl >::neg_format ( ) const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern `{symbol,sign,value,none}`,

`curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `10.01`.

### Returns

Pattern for money values.

Definition at line 1145 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_neg_format()`.

**5.555.4.15** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl >::negative_sign ( ) const  
[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

### Returns

*string\_type* representing a negative sign.

Definition at line 1089 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_negative_sign()`.

**5.555.4.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct<  
_CharT, _Intl >::pos_format ( ) const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be

present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

### Returns

Pattern for money values.

Definition at line 1141 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_pos_format()`.

**5.555.4.17** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct< _CharT, _Intl >::positive_sign ( ) const  
[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `money_punct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

### Returns

*string\_type* representing a positive sign.

Definition at line 1072 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_positive_sign()`.

**5.555.4.18** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const  
[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `money_punct<char_type>::do_thousands_sep()`.

### Returns

`char_type` representing a thousands separator.

Definition at line 1012 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

### 5.555.5 Member Data Documentation

**5.555.5.1** `template<typename _CharT, bool _Intl> locale::id  
std::moneypunct<_CharT, _Intl >::id [static]`

Numpunct facet id.

Definition at line 948 of file locale\_facets\_nonio.h.

**5.555.5.2** `template<typename _CharT, bool _Intl> const bool  
std::moneypunct<_CharT, _Intl >::intl [static]`

This value is provided by the standard, but no reason for its `///` existence.

Reimplemented in [std::moneypunct\\_byname<\\_CharT, \\_Intl >](#).

Definition at line 946 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

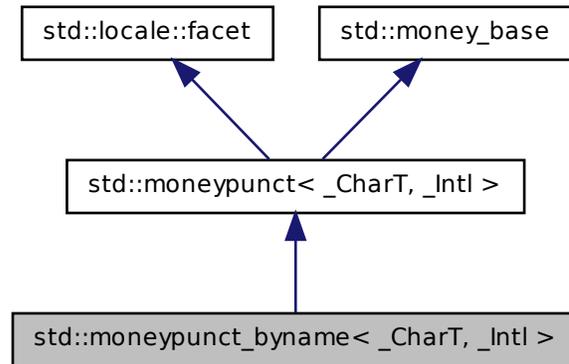
- [locale\\_facets\\_nonio.h](#)

## 5.556 `std::moneypunct_byname<_CharT, _Intl >` Class Template Reference

class [moneypunct\\_byname](#) [22.2.6.4].

## 5.556 `std::moneypunct_byname<_CharT, _Intl >` Class Template Reference 2651

Inheritance diagram for `std::moneypunct_byname<_CharT, _Intl >`:



### Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache<_CharT, _Intl >` `__cache_type`
- typedef `_CharT` `char_type`
- enum `part` {  
    `none`, `space`, `symbol`, `sign`,  
    `value` }
- typedef `basic_string<_CharT >` `string_type`

### Public Member Functions

- `moneypunct_byname` (const char \*`__s`, size\_t `__refs=0`)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- int `frac_digits` () const
- `string grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const

- pattern [pos\\_format](#) () const
- pattern [neg\\_format](#) () const

### Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **`_S_construct_pattern`** (char `__precedes`, char `__space`, char `__posn`) throw ()

### Static Public Attributes

- static const char \* **`_S_atoms`**
- static const pattern **`_S_default_pattern`**
- static [locale::id](#) `id`
- static const bool [intl](#)

### Protected Member Functions

- template<>  
void **`_M_initialize_moneypunct`** (`__c_locale`, const char \*)
- void **`_M_initialize_moneypunct`** (`__c_locale` `__cloc=NULL`, const char \*`__name=NULL`)
- template<>  
void **`_M_initialize_moneypunct`** (`__c_locale`, const char \*)
- template<>  
void **`_M_initialize_moneypunct`** (`__c_locale`, const char \*)
- template<>  
void **`_M_initialize_moneypunct`** (`__c_locale`, const char \*)
- virtual [string\\_type](#) **`do_curr_symbol`** () const
- virtual [char\\_type](#) **`do_decimal_point`** () const
- virtual int **`do_frac_digits`** () const
- virtual [string](#) **`do_grouping`** () const
- virtual pattern **`do_neg_format`** () const
- virtual [string\\_type](#) **`do_negative_sign`** () const
- virtual pattern **`do_pos_format`** () const
- virtual [string\\_type](#) **`do_positive_sign`** () const
- virtual [char\\_type](#) **`do_thousands_sep`** () const

### Static Protected Member Functions

- static `__c_locale` **`_S_clone_c_locale`** (`__c_locale` &`__cloc`) throw ()
- static void **`_S_create_c_locale`** (`__c_locale` &`__cloc`, const char \*`__s`, `__c_locale` `__old=0`)

## 5.556 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference2653

- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_GLIBCXX\_CONST const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class **locale::\_Impl**

### 5.556.1 Detailed Description

**template<typename \_CharT, bool \_Intl> class std::moneypunct\_byname< \_CharT, \_Intl >**

class [moneypunct\\_byname](#) [22.2.6.4].

Definition at line 1318 of file locale\_facets\_nonio.h.

### 5.556.2 Member Typedef Documentation

**5.556.2.1 template<typename \_CharT, bool \_Intl> typedef \_CharT  
std::moneypunct\_byname<\_CharT, \_Intl >::char\_type**

Public typedefs.

Reimplemented from [std::moneypunct<\\_CharT, \\_Intl >](#).

Definition at line 1321 of file locale\_facets\_nonio.h.

**5.556.2.2 template<typename \_CharT, bool \_Intl> typedef  
basic\_string<\_CharT> std::moneypunct\_byname<\_CharT, \_Intl  
>::string\_type**

Public typedefs.

Reimplemented from [std::moneypunct<\\_CharT, \\_Intl >](#).

Definition at line 1322 of file locale\_facets\_nonio.h.

### 5.556.3 Member Function Documentation

**5.556.3.1** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl >::curr_symbol ( ) const  
[inline, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1055 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_curr_symbol()`.

**5.556.3.2** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct<_CharT, _Intl >::decimal_point ( ) const  
[inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

Definition at line 999 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_decimal_point()`.

**5.556.3.3** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct<_CharT, _Intl >::do_curr_symbol ( ) const  
[inline, protected, virtual, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See also

`curr_symbol()` for details.

## 5.556 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference 2655

### Returns

*string\_type* representing a currency symbol.

Definition at line 1201 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::curr\_symbol().

**5.556.3.4** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl >::do_decimal_point ( ) const  
[inline, protected, virtual, inherited]`

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

### Returns

*char\_type* representing a decimal point.

Definition at line 1163 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::decimal\_point().

**5.556.3.5** `template<typename _CharT, bool _Intl> virtual int  
std::moneypunct<_CharT, _Intl >::do_frac_digits ( ) const  
[inline, protected, virtual, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

### See also

[frac\\_digits\(\)](#) for details.

### Returns

Number of digits in amount fraction.

Definition at line 1241 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::frac\_digits().

**5.556.3.6** `template<typename _CharT, bool _Intl> virtual string  
std::moneypunct< _CharT, _Intl >::do_grouping ( ) const  
[inline, protected, virtual, inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1188 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::grouping()`.

**5.556.3.7** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const  
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[neg\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1269 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct< _CharT, _Intl >::neg_format()`.

**5.556.3.8** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const  
[inline, protected, virtual, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

## 5.556 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference 2657

### See also

[negative\\_sign\(\)](#) for details.

### Returns

*string\_type* representing a negative sign.

Definition at line 1227 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::negative\_sign().

**5.556.3.9** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct<_CharT, _Intl >::do_pos_format ( ) const  
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

### See also

[pos\\_format\(\)](#) for details.

### Returns

Pattern for money values.

Definition at line 1255 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl >::pos\_format().

**5.556.3.10** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct<_CharT, _Intl >::do_positive_sign ( ) const  
[inline, protected, virtual, inherited]`

Return positive sign string.

This function returns a *string\_type* to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

### See also

[positive\\_sign\(\)](#) for details.

### Returns

*string\_type* representing a positive sign.

Definition at line 1214 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::positive_sign()`.

**5.556.3.11** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl >::do_thousands_sep ( ) const  
[inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1175 of file locale\_facets\_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::thousands_sep()`.

**5.556.3.12** `template<typename _CharT, bool _Intl> int std::moneypunct<  
_CharT, _Intl >::frac_digits ( ) const [inline, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

#### Returns

Number of digits in amount fraction.

Definition at line 1105 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

**5.556.3.13** `template<typename _CharT, bool _Intl> string std::moneypunct<  
_CharT, _Intl >::grouping ( ) const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

## **5.556 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference**

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the [grouping\(\)](#) returns `32` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling [moneypunct<char\\_type>::do\\_grouping\(\)](#).

### **Returns**

string representing grouping specification.

Definition at line 1042 of file `locale_facets_nonio.h`.

References [std::moneypunct<\\_CharT, \\_Intl >::do\\_grouping\(\)](#).

### **5.556.3.14 template<typename \_CharT, bool \_Intl> pattern std::moneypunct<\_CharT, \_Intl >::neg\_format ( ) const [inline, inherited]**

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning [moneypunct<char\\_type>::do\\_pos\\_format\(\)](#) or [moneypunct<char\\_type>::do\\_neg\\_format\(\)](#).

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of [curr\\_symbol\(\)](#) may be present. The sign field indicates that the value of [positive\\_sign\(\)](#) or [negative\\_sign\(\)](#) must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and [pos\\_format\(\)](#) pattern {symbol,sign,value,none}, [curr\\_symbol\(\) == '\\$'](#) [positive\\_sign\(\) == '+'](#), and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

### **Returns**

Pattern for money values.

Definition at line 1145 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_neg_format()`.

**5.556.3.15** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl >::negative_sign ( ) const  
[inline, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1089 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_negative_sign()`.

**5.556.3.16** `template<typename _CharT, bool _Intl> pattern std::moneypunct<  
_CharT, _Intl >::pos_format ( ) const [inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

## 5.556 std::moneypunct\_byname<\_CharT, \_Intl > Class Template Reference 2661

### Returns

Pattern for money values.

Definition at line 1141 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_pos_format()`.

**5.556.3.17** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl >::positive_sign ( ) const  
[inline, inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `money_punct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

### Returns

*string\_type* representing a positive sign.

Definition at line 1072 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_positive_sign()`.

**5.556.3.18** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct<_CharT, _Intl >::thousands_sep ( ) const  
[inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `money_punct<char_type>::do_thousands_sep()`.

### Returns

`char_type` representing a thousands separator.

Definition at line 1012 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

## 5.556.4 Member Data Documentation

**5.556.4.1** `template<typename _CharT, bool _Intl> locale::id`  
`std::moneypunct<_CharT, _Intl>::id` [`static`, `inherited`]

Numpunct facet id.

Definition at line 948 of file `locale_facets_nonio.h`.

**5.556.4.2** `template<typename _CharT, bool _Intl> const bool`  
`std::moneypunct_byname<_CharT, _Intl>::intl` [`static`]

This value is provided by the standard, but no reason for its /// existence.

Reimplemented from `std::moneypunct<_CharT, _Intl>`.

Definition at line 1324 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.557 `std::move_iterator<_Iterator>` Class Template Reference

### Public Types

- `typedef __traits_type::difference_type` **difference\_type**
- `typedef __traits_type::iterator_category` **iterator\_category**
- `typedef _Iterator` **iterator\_type**
- `typedef _Iterator` **pointer**
- `typedef value_type &&` **reference**
- `typedef __traits_type::value_type` **value\_type**

### Public Member Functions

- `move_iterator` (`iterator_type __i`)
- `template<typename _Iter >`  
`move_iterator` (`const` `move_iterator<_Iter > &__i`)
- `iterator_type` **base** () `const`
- `reference` **operator\*** () `const`
- `move_iterator` **operator+** (`difference_type __n`) `const`
- `move_iterator` **operator++** (`int`)

## 5.558 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

- `move_iterator` & `operator++` ()
- `move_iterator` & `operator+=` (difference\_type \_\_n)
- `move_iterator` `operator-` (difference\_type \_\_n) const
- `move_iterator` & `operator--` ()
- `move_iterator` `operator--` (int)
- `move_iterator` & `operator-=` (difference\_type \_\_n)
- pointer `operator->` () const
- reference `operator[]` (difference\_type \_\_n) const

### Protected Types

- typedef `iterator_traits<_Iterator >` `__traits_type`

### Protected Attributes

- `_Iterator _M_current`

### 5.557.1 Detailed Description

`template<typename _Iterator> class std::move_iterator<_Iterator >`

Class template `move_iterator` is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 916 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- `stl_iterator.h`

## 5.558 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const\_pointer**
- typedef `_Pair_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const_Key, _Tp >` **value\_type**

## Public Member Functions

- `multimap` ()
- `multimap` (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `multimap` (`multimap` &&\_\_x)
- `multimap` (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `multimap` (const `multimap` &\_\_x)
- template<typename `_InputIterator` >  
`multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
`multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- iterator `begin` ()
- `const_iterator` `begin` () const
- `const_iterator` `cbegin` () const
- `const_iterator` `end` () const
- void `clear` ()
- `size_type` `count` (const `key_type` &\_\_x) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- bool `empty` () const
- iterator `end` ()
- `const_iterator` `end` () const

## 5.558 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

- `std::pair< const_iterator, const_iterator >` `equal_range` (const key\_type &\_\_x) const
- `std::pair< iterator, iterator >` `equal_range` (const key\_type &\_\_x)
- iterator `erase` (iterator \_\_first, iterator \_\_last)
- size\_type `erase` (const key\_type &\_\_x)
- iterator `erase` (iterator \_\_position)
- iterator `find` (const key\_type &\_\_x)
- const\_iterator `find` (const key\_type &\_\_x) const
- allocator\_type `get_allocator` () const
- iterator `insert` (const value\_type &\_\_x)
- iterator `insert` (iterator \_\_position, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `insert` (initializer\_list< value\_type > \_\_l)
- key\_compare `key_comp` () const
- const\_iterator `lower_bound` (const key\_type &\_\_x) const
- iterator `lower_bound` (const key\_type &\_\_x)
- size\_type `max_size` () const
- `multimap` & `operator=` (const `multimap` &\_\_x)
- `multimap` & `operator=` (`multimap` &&\_\_x)
- `multimap` & `operator=` (initializer\_list< value\_type > \_\_l)
- reverse\_iterator `rbegin` ()
- const\_reverse\_iterator `rbegin` () const
- reverse\_iterator `rend` ()
- const\_reverse\_iterator `rend` () const
- size\_type `size` () const
- void `swap` (`multimap` &\_\_x)
- const\_iterator `upper_bound` (const key\_type &\_\_x) const
- iterator `upper_bound` (const key\_type &\_\_x)
- value\_compare `value_comp` () const

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool `operator<` (const `multimap`< \_K1, \_T1, \_C1, \_A1 > &, const `multimap`< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool `operator==` (const `multimap`< \_K1, \_T1, \_C1, \_A1 > &, const `multimap`< \_K1, \_T1, \_C1, \_A1 > &)

### 5.558.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class
std::multimap< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_multimap.h`.

### 5.558.2 Constructor & Destructor Documentation

**5.558.2.1** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap( ) [inline]`

Default constructor creates no elements.

Definition at line 148 of file `stl_multimap.h`.

**5.558.2.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap( const _Compare & _comp, const allocator_type & _a = allocator_type() ) [inline, explicit]`

Creates a `multimap` with no elements.

#### Parameters

*comp* A comparison object.

*a* An allocator object.

Definition at line 157 of file `stl_multimap.h`.

## 5.558 `std::multimap<_Key, _Tp, _Compare, _Alloc>` Class Template Reference

**5.558.2.3** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const multimap<_Key, _Tp, _Compare, _Alloc> & _x ) [inline]`

Multimap copy constructor.

### Parameters

**x** A multimap of identical element and allocator types.

The newly-created multimap uses a copy of the allocation object used by *x*.

Definition at line 168 of file `stl_multimap.h`.

**5.558.2.4** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( multimap<_Key, _Tp, _Compare, _Alloc> && _x ) [inline]`

Multimap move constructor.

### Parameters

**x** A multimap of identical element and allocator types.

The newly-created multimap contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multimap.

Definition at line 179 of file `stl_multimap.h`.

**5.558.2.5** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( initializer_list<value_type> __l, const _Compare & __comp = _Compare(), const allocator_type & __a = allocator_type() ) [inline]`

Builds a multimap from an [initializer\\_list](#).

### Parameters

**l** An [initializer\\_list](#).

**comp** A comparison functor.

*a* An allocator object.

Create a multimap consisting of copies of the elements from the [initializer\\_list](#). This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 192 of file `stl_multimap.h`.

```
5.558.2.6  template<typename _Key, typename _Tp, typename _Compare =
           std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
           _Key, _Tp>>> template<typename _InputIterator >
           std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
           _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a multimap from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 209 of file `stl_multimap.h`.

```
5.558.2.7  template<typename _Key, typename _Tp, typename _Compare =
           std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
           _Key, _Tp>>> template<typename _InputIterator >
           std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
           _InputIterator __first, _InputIterator __last, const _Compare &
           __comp, const allocator_type & __a = allocator_type() )
           [inline]
```

Builds a multimap from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 225 of file `stl_multimap.h`.

### 5.558.3 Member Function Documentation

**5.558.3.1** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::begin ( ) [inline]`

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 304 of file `stl_multimap.h`.

**5.558.3.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 313 of file `stl_multimap.h`.

**5.558.3.3** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 377 of file `stl_multimap.h`.

**5.558.3.4** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 386 of file `stl_multimap.h`.

**5.558.3.5** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear ( ) [inline]`

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 601 of file `stl_multimap.h`.

**5.558.3.6** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::count ( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

#### Returns

Number of elements with specified key.

Definition at line 658 of file `stl_multimap.h`.

**5.558.3.7** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 395 of file `stl_multimap.h`.

**5.558.3.8** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 404 of file `stl_multimap.h`.

## 5.558 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.558.3.9** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline]`

Returns true if the multimap is empty.

Definition at line 411 of file stl\_multimap.h.

**5.558.3.10** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl\_multimap.h.

**5.558.3.11** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl\_multimap.h.

**5.558.3.12** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range ( const key_type & __x ) [inline]`

Finds a subsequence matching given key.

### Parameters

*x* Key of (key, value) pairs to be located.

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 725 of file stl\_multimap.h.

**5.558.3.13** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::multimap<_Key, _Tp, _Compare, _Alloc>::equal_range ( const key_type & __x ) const [inline]`

Finds a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 742 of file stl\_multimap.h.

**5.558.3.14** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::erase ( iterator __position ) [inline]`

Erases an element from a multimap.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

## 5.558 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 509 of file stl\_multimap.h.

```
5.558.3.15 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,  
_Alloc >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

### Parameters

*x* Key of element to be erased.

### Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 539 of file stl\_multimap.h.

```
5.558.3.16 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::erase ( iterator __first, iterator __last ) [inline]
```

Erases a [first,last) range of elements from a multimap.

### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

### Returns

The iterator *last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 558 of file stl\_multimap.h.

```
5.558.3.17 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::find ( const key_type & __x ) const
[inline]
```

Tries to locate an element in a multimap.

#### Parameters

**x** Key of (key, value) pair to be located.

#### Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 649 of file stl\_multimap.h.

```
5.558.3.18 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::find ( const key_type & __x ) [inline]
```

Tries to locate an element in a multimap.

#### Parameters

**x** Key of (key, value) pair to be located.

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 634 of file stl\_multimap.h.

## 5.558 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.558.3.19** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::multimap<_Key, _Tp, _Compare, _Alloc >::get_allocator( ) const [inline]`

Get a copy of the memory allocation object.

Definition at line 294 of file stl\_multimap.h.

**5.558.3.20** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert( const value_type & __x ) [inline]`

Inserts a [std::pair](#) into the multimap.

### Parameters

**x** Pair to be inserted (see `std::make_pair` for easy creation of pairs).

### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 438 of file stl\_multimap.h.

**5.558.3.21** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert( iterator __position, const value_type & __x ) [inline]`

Inserts a [std::pair](#) into the multimap.

### Parameters

**position** An iterator that serves as a hint as to where the pair should be inserted.

**x** Pair to be inserted (see `std::make_pair` for easy creation of pairs).

### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 462 of file `stl_multimap.h`.

```
5.558.3.22 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> template<typename _InputIterator > void
std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 476 of file `stl_multimap.h`.

```
5.558.3.23 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> void std::multimap< _Key, _Tp, _Compare, _Alloc
>::insert ( initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of `std::pairs` into the multimap.

#### Parameters

*list* A `std::initializer_list<value_type>` of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 488 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

## 5.558 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

**5.558.3.24** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> key_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]`

Returns the key comparison object out of which the multimap was constructed.

Definition at line 610 of file stl\_multimap.h.

**5.558.3.25** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound ( const key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to `end()`.

Definition at line 688 of file stl\_multimap.h.

**5.558.3.26** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 673 of file `stl_multimap.h`.

```
5.558.3.27 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare,
_Alloc>::max_size ( ) const [inline]
```

Returns the maximum size of the multimap.

Definition at line 421 of file `stl_multimap.h`.

```
5.558.3.28 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare,
_Alloc>::operator= ( multimap<_Key, _Tp, _Compare, _Alloc >
&& __x ) [inline]
```

Multimap move assignment operator.

#### Parameters

*x* A multimap of identical element and allocator types.

The contents of *x* are moved into this multimap (without copying). *x* is a valid, but unspecified multimap.

Definition at line 263 of file `stl_multimap.h`.

References `std::swap()`.

```
5.558.3.29 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare,
_Alloc>::operator= ( const multimap<_Key, _Tp, _Compare,
_Alloc > & __x ) [inline]
```

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.558 `std::multimap<_Key, _Tp, _Compare, _Alloc >` Class Template Reference

### Parameters

*x* A multimap of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 248 of file `stl_multimap.h`.

```
5.558.3.30 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare,  
_Alloc >::operator= ( initializer_list< value_type > __l )  
[inline]
```

Multimap list assignment operator.

### Parameters

*l* An [initializer\\_list](#).

This function fills a multimap with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 284 of file `stl_multimap.h`.

```
5.558.3.31 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> reverse_iterator std::multimap<_Key, _Tp,  
_Compare, _Alloc >::rbegin ( ) [inline]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 340 of file `stl_multimap.h`.

```
5.558.3.32 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp,  
_Compare, _Alloc >::rbegin ( ) const [inline]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 349 of file `stl_multimap.h`.

**5.558.3.33** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 358 of file `stl_multimap.h`.

**5.558.3.34** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 367 of file `stl_multimap.h`.

**5.558.3.35** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::size ( ) const [inline]`

Returns the size of the multimap.

Definition at line 416 of file `stl_multimap.h`.

**5.558.3.36** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::swap ( multimap<_Key, _Tp, _Compare, _Alloc> & __x ) [inline]`

Swaps data with another multimap.

#### Parameters

- `x` A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

## 5.558 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

Definition at line 591 of file stl\_multimap.h.

Referenced by std::swap().

```
5.558.3.37 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::upper_bound ( const key_type & __x ) const  
[inline]
```

Finds the end of a subsequence matching given key.

### Parameters

**x** Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 708 of file stl\_multimap.h.

```
5.558.3.38 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

### Parameters

**x** Key of (key, value) pair to be located.

### Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 698 of file stl\_multimap.h.

```
5.558.3.39 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> value_compare std::multimap< _Key, _Tp,  
_Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

Definition at line 618 of file `stl_multimap.h`.

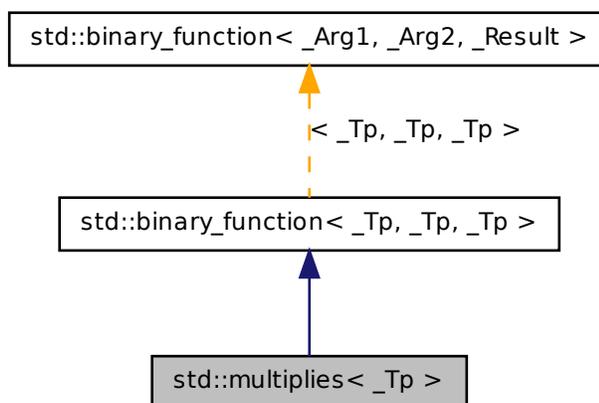
The documentation for this class was generated from the following file:

- [stl\\_multimap.h](#)

## 5.559 `std::multiplies< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::multiplies< _Tp >`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

### 5.559.1 Detailed Description

`template<typename _Tp> struct std::multiplies<_Tp >`

One of the [math functors](#).

Definition at line 153 of file `stl_function.h`.

### 5.559.2 Member Typedef Documentation

**5.559.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp  
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.559.2.2** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp >::result_type  
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.559.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.560 `std::multiset<_Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of elements, which can be retrieved in logarithmic time.

### Public Types

- `typedef _Alloc allocator_type`

- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Key_alloc_type::const_pointer` **const\_pointer**
- typedef `_Key_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::const_iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Key_alloc_type::pointer` **pointer**
- typedef `_Key_alloc_type::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- [multiset](#) ()
- [multiset](#) (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const [multiset](#) &\_\_x)
- `template<typename _InputIterator >`  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- [multiset](#) ([multiset](#) &&\_\_x)
- [multiset](#) (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- iterator [begin](#) () const
- iterator [cbegin](#) () const
- iterator [cend](#) () const
- void [clear](#) ()
- `size_type` [count](#) (const `key_type` &\_\_x) const
- `reverse_iterator` [crbegin](#) () const
- `reverse_iterator` [crend](#) () const
- bool [empty](#) () const
- iterator [end](#) () const
- iterator [erase](#) (iterator \_\_first, iterator \_\_last)
- `size_type` [erase](#) (const `key_type` &\_\_x)
- iterator [erase](#) (iterator \_\_position)
- `allocator_type` [get\\_allocator](#) () const

## 5.560 `std::multiset<_Key, _Compare, _Alloc>` Class Template Reference 2685

- iterator `insert` (const value\_type &\_\_x)
  - iterator `insert` (iterator \_\_position, const value\_type &\_\_x)
  - template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void `insert` (initializer\_list< value\_type > \_\_l)
  - key\_compare `key_comp` () const
  - size\_type `max_size` () const
  - `multiset` & `operator=` (const `multiset` &\_\_x)
  - `multiset` & `operator=` (`multiset` &&\_\_x)
  - `multiset` & `operator=` (initializer\_list< value\_type > \_\_l)
  - `reverse_iterator` `rbegin` () const
  - `reverse_iterator` `rend` () const
  - size\_type `size` () const
  - void `swap` (`multiset` &\_\_x)
  - value\_compare `value_comp` () const
- 
- iterator `find` (const key\_type &\_\_x)
  - const\_iterator `find` (const key\_type &\_\_x) const
- 
- iterator `lower_bound` (const key\_type &\_\_x)
  - const\_iterator `lower_bound` (const key\_type &\_\_x) const
- 
- iterator `upper_bound` (const key\_type &\_\_x)
  - const\_iterator `upper_bound` (const key\_type &\_\_x) const
- 
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)
  - `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x)  
const
  - template<typename \_K1, typename \_C1, typename \_A1 >  
bool `operator==` (const `multiset`< \_K1, \_C1, \_A1 > &, const `multiset`< \_K1, \_C1, \_A1 > &)
  - template<typename \_K1, typename \_C1, typename \_A1 >  
bool `operator<` (const `multiset`< \_K1, \_C1, \_A1 > &, const `multiset`< \_K1, \_C1, \_A1 > &)

### 5.560.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::multiset<_Key, _Compare, _Alloc >`

A standard container made up of elements, which can be retrieved in logarithmic time. Meets the requirements of a `container`, a `reversible container`, and an

*associative container* (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 84 of file `stl_multiset.h`.

## 5.560.2 Constructor & Destructor Documentation

**5.560.2.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( ) [inline]`

Default constructor creates no elements.

Definition at line 129 of file `stl_multiset.h`.

**5.560.2.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( const _Compare & __comp, const  
allocator_type & __a = allocator_type() ) [inline,  
explicit]`

Creates a multiset with no elements.

### Parameters

*comp* Comparator to use.

*a* An allocator object.

Definition at line 138 of file `stl_multiset.h`.

**5.560.2.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset  
( _InputIterator __first, _InputIterator __last ) [inline]`

Builds a multiset from a range.

### Parameters

*first* An input iterator.

## 5.560 `std::multiset<_Key, _Compare, _Alloc >` Class Template Reference 2687

*last* An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 152 of file `stl_multiset.h`.

```
5.560.2.4 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> template<typename
                _InputIterator > std::multiset<_Key, _Compare, _Alloc >::multiset
                ( _InputIterator __first, _InputIterator __last, const _Compare &
                __comp, const allocator_type & __a = allocator_type() )
                [inline]
```

Builds a multiset from a range.

### Parameters

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 168 of file `stl_multiset.h`.

```
5.560.2.5 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> std::multiset<_Key,
                _Compare, _Alloc >::multiset ( const multiset<_Key, _Compare,
                _Alloc > & __x ) [inline]
```

Multiset copy constructor.

### Parameters

*x* A multiset of identical element and allocator types.

The newly-created multiset uses a copy of the allocation object used by *x*.

Definition at line 181 of file `stl_multiset.h`.

**5.560.2.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( multiset< _Key, _Compare, _Alloc >  
&& _x ) [inline]`

Multiset move constructor.

#### Parameters

*x* A multiset of identical element and allocator types.

The newly-created multiset contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multiset.

Definition at line 192 of file `stl_multiset.h`.

**5.560.2.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( initializer_list< value_type > __l,  
const _Compare & __comp = _Compare(), const allocator_type &  
__a = allocator_type() ) [inline]`

Builds a multiset from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#).

*comp* A comparison functor.

*a* An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in *N* if the list is already sorted, and *N*log*N* otherwise (where *N* is *l.size()*).

Definition at line 205 of file `stl_multiset.h`.

### 5.560.3 Member Function Documentation

**5.560.3.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 285 of file `stl_multiset.h`.

## 5.560 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 2689

**5.560.3.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl\_multiset.h.

**5.560.3.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl\_multiset.h.

**5.560.3.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::multiset< _Key,  
_Compare, _Alloc >::clear ( ) [inline]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 541 of file stl\_multiset.h.

**5.560.3.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::count ( const key_type & __x ) const  
[inline]`

Finds the number of elements with given key.

### **Parameters**

*x* Key of elements to be located.

### **Returns**

Number of elements with specified key.

Definition at line 552 of file stl\_multiset.h.

**5.560.3.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset< _Key, _Compare, _Alloc >::crbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 340 of file `stl_multiset.h`.

**5.560.3.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset< _Key, _Compare, _Alloc >::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 349 of file `stl_multiset.h`.

**5.560.3.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> bool std::multiset< _Key,  
_Compare, _Alloc >::empty ( ) const [inline]`

Returns true if the set is empty.

Definition at line 355 of file `stl_multiset.h`.

**5.560.3.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 294 of file `stl_multiset.h`.

**5.560.3.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (  
const key_type & __x ) [inline]`

Finds a subsequence matching given key.

## 5.560 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 2691

### Parameters

*x* Key to be located.

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 632 of file stl\_multiset.h.

```
5.560.3.11 template<typename _Key, typename _Compare = std::less<_Key>,  
                 typename _Alloc = std::allocator<_Key>>> std::pair<const_iterator,  
                 const_iterator> std::multiset< _Key, _Compare, _Alloc  
                 >::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

### Parameters

*x* Key to be located.

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 636 of file stl\_multiset.h.

```
5.560.3.12 template<typename _Key, typename _Compare = std::less<_Key>,  
                 typename _Alloc = std::allocator<_Key>>> iterator std::multiset<  
                 _Key, _Compare, _Alloc >::erase ( iterator __position )  
                 [inline]
```

Erases an element from a multiset.

**Parameters**

*position* An iterator pointing to the element to be erased.

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 466 of file `stl_multiset.h`.

```
5.560.3.13 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::erase ( const key_type & __x )
[inline]
```

Erases elements according to the provided key.

**Parameters**

*x* Key of element to be erased.

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 496 of file `stl_multiset.h`.

```
5.560.3.14 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::erase ( iterator __first, iterator __last
) [inline]
```

Erases a `[first,last)` range of elements from a multiset.

**Parameters**

*first* Iterator pointing to the start of the range to be erased.

## 5.560 `std::multiset<_Key, _Compare, _Alloc >` Class Template Reference 2693

*last* Iterator pointing to the end of the range to be erased.

### Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 515 of file `stl_multiset.h`.

```
5.560.3.15 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::find ( const key_type & __x )
[inline]
```

Tries to locate an element in a set.

### Parameters

*x* Element to be located.

### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 570 of file `stl_multiset.h`.

```
5.560.3.16 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator
std::multiset<_Key, _Compare, _Alloc >::find ( const key_type &
__x ) const [inline]
```

Tries to locate an element in a set.

### Parameters

*x* Element to be located.

### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 574 of file `stl_multiset.h`.

```
5.560.3.17 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> allocator_type
std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const
[inline]
```

Returns the memory allocation object.

Definition at line 276 of file `stl_multiset.h`.

```
5.560.3.18 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::insert ( const value_type & __x )
[inline]
```

Inserts an element into the multiset.

#### Parameters

*x* Element to be inserted.

#### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 396 of file `stl_multiset.h`.

```
5.560.3.19 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::insert ( iterator __position, const
value_type & __x ) [inline]
```

Inserts an element into the multiset.

#### Parameters

*position* An iterator that serves as a hint as to where the element should be inserted.

## 5.560 `std::multiset<_Key, _Compare, _Alloc >` Class Template Reference 2695

*x* Element to be inserted.

### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 420 of file `stl_multiset.h`.

```
5.560.3.20 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > void std::multiset<_Key, _Compare, _Alloc
>::insert ( _InputIterator __first, _InputIterator __last )
[inline]
```

A template function that tries to insert a range of elements.

### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 433 of file `stl_multiset.h`.

```
5.560.3.21 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::multiset<
_Key, _Compare, _Alloc >::insert ( initializer_list< value_type >
__l ) [inline]
```

Attempts to insert a list of elements into the multiset.

### Parameters

*list* A `std::initializer_list<value_type>` of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 445 of file `stl_multiset.h`.

References `std::multiset<_Key, _Compare, _Alloc >::insert()`.

Referenced by `std::multiset<_Key, _Compare, _Alloc >::insert()`.

```
5.560.3.22 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> key_compare
std::multiset<_Key, _Compare, _Alloc >::key_comp ( ) const
[inline]
```

Returns the comparison object.

Definition at line 268 of file `stl_multiset.h`.

```
5.560.3.23 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::lower_bound ( const key_type & __x )
[inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

`x` Key to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 591 of file `stl_multiset.h`.

```
5.560.3.24 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator
std::multiset<_Key, _Compare, _Alloc >::lower_bound ( const
key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

#### Parameters

`x` Key to be located.

## 5.560 `std::multiset<_Key, _Compare, _Alloc >` Class Template Reference 2697

### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 595 of file `stl_multiset.h`.

**5.560.3.25** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::max_size( ) const [inline]`

Returns the maximum size of the set.

Definition at line 365 of file `stl_multiset.h`.

**5.560.3.26** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> multiset&  
std::multiset<_Key, _Compare, _Alloc >::operator= ( multiset<  
_Key, _Compare, _Alloc > && __x ) [inline]`

Multiset move assignment operator.

### Parameters

*x* A multiset of identical element and allocator types.

The contents of *x* are moved into this multiset (without copying). *x* is a valid, but unspecified multiset.

Definition at line 235 of file `stl_multiset.h`.

References `std::swap()`.

**5.560.3.27** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> multiset&  
std::multiset<_Key, _Compare, _Alloc >::operator= (   
initializer_list<value_type > __l ) [inline]`

Multiset list assignment operator.

### Parameters

*l* An `initializer_list`.

This function fills a multiset with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 256 of file `stl_multiset.h`.

```
5.560.3.28 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset< _Key, _Compare, _Alloc >::operator= ( const
multiset< _Key, _Compare, _Alloc > & __x ) [inline]
```

Multiset assignment operator.

#### Parameters

*x* A multiset of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 220 of file `stl_multiset.h`.

```
5.560.3.29 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const
[inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 303 of file `stl_multiset.h`.

```
5.560.3.30 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rend ( ) const
[inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 312 of file `stl_multiset.h`.

## 5.560 std::multiset<\_Key, \_Compare, \_Alloc> Class Template Reference 2699

**5.560.3.31** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc>::size ( ) const [inline]`

Returns the size of the set.

Definition at line 360 of file stl\_multiset.h.

**5.560.3.32** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::multiset<  
_Key, _Compare, _Alloc>::swap ( multiset<_Key, _Compare,  
_Alloc> & __x ) [inline]`

Swaps data with another multiset.

### Parameters

*x* A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 380 of file stl\_multiset.h.

Referenced by `std::swap()`.

**5.560.3.33** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc>::upper_bound ( const key_type & __x )  
[inline]`

Finds the end of a subsequence matching given key.

### Parameters

*x* Key to be located.

### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 607 of file stl\_multiset.h.

**5.560.3.34** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator  
std::multiset<_Key, _Compare, _Alloc >::upper_bound ( const  
key_type & __x ) const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 611 of file `stl_multiset.h`.

**5.560.3.35** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> value_compare  
std::multiset<_Key, _Compare, _Alloc >::value_comp ( ) const  
[inline]`

Returns the comparison object.

Definition at line 272 of file `stl_multiset.h`.

## 5.560.4 Friends And Related Function Documentation

**5.560.4.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename _K1  
, typename _C1, typename _A1 > bool operator< ( const multiset<  
_K1, _C1, _A1 > &, const multiset<_K1, _C1, _A1 > & )  
[friend]`

Finds a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

```
5.560.4.2 template<typename _Key, typename _Compare = std::less<_Key>,
                 typename _Alloc = std::allocator<_Key>> template<typename
                 _K1, typename _C1, typename _A1 > bool operator==( const
                 multiset<_K1, _C1, _A1 > &, const multiset<_K1, _C1, _A1 > &
                 ) [friend]
```

Finds a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

## 5.561 std::mutex Class Reference

mutex

#### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

## Public Member Functions

- **mutex** (const [mutex](#) &)
- void **lock** ()
- native\_handle\_type **native\_handle** ()
- [mutex](#) & **operator=** (const [mutex](#) &)
- bool **try\_lock** ()
- void **unlock** ()

### 5.561.1 Detailed Description

`mutex`

Definition at line 63 of file `mutex`.

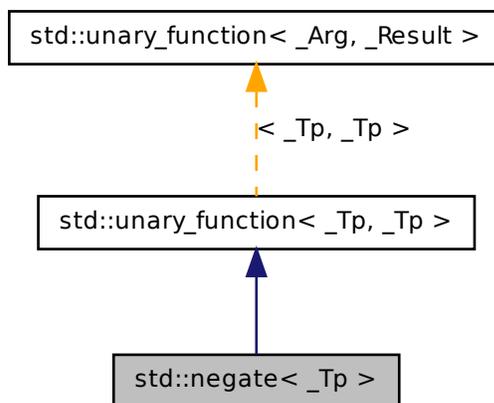
The documentation for this class was generated from the following file:

- [mutex](#)

## 5.562 `std::negate<_Tp>` Struct Template Reference

One of the [math functions](#).

Inheritance diagram for `std::negate<_Tp>`:



## Public Types

- typedef `_Tp` `argument_type`
- typedef `_Tp` `result_type`

## Public Member Functions

- `_Tp operator() (const _Tp &__x) const`

### 5.562.1 Detailed Description

`template<typename _Tp> struct std::negate< _Tp >`

One of the [math functors](#).

Definition at line 180 of file `stl_function.h`.

### 5.562.2 Member Typedef Documentation

#### 5.562.2.1 `typedef _Tp std::unary_function< _Tp , _Tp >::argument_type` [inherited]

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

#### 5.562.2.2 `typedef _Tp std::unary_function< _Tp , _Tp >::result_type` [inherited]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.563 `std::negative_binomial_distribution< _IntType >` Class Template Reference

A [negative\\_binomial\\_distribution](#) random number distribution.

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- [negative\\_binomial\\_distribution](#) (`_IntType __k=1, double __p=0.5`)
- [negative\\_binomial\\_distribution](#) (`const param\_type &__p`)
- `_IntType k` () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator](#)() (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator](#)() (`_UniformRandomNumberGenerator &__urng`)
- `double p` () const
- `void param` (`const param\_type &__param`)
- [param\\_type param](#) () const
- `void reset` ()

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::negative\_binomial\_distribution< _IntType1 > &)`
- `template<typename _IntType1 >`  
`bool operator== (const std::negative\_binomial\_distribution< _IntType1 > &__d1, const std::negative\_binomial\_distribution< _IntType1 > &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::negative\_binomial\_distribution< _IntType1 > &)`

### 5.563.1 Detailed Description

```
template<typename _IntType = int> class std::negative_binomial_distribution<
_IntType >
```

A [negative\\_binomial\\_distribution](#) random number distribution. The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3792 of file random.h.

### 5.563.2 Member Typedef Documentation

**5.563.2.1** `template<typename _IntType = int> typedef _IntType  
std::negative_binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3799 of file random.h.

### 5.563.3 Member Function Documentation

**5.563.3.1** `template<typename _IntType = int> _IntType  
std::negative_binomial_distribution< _IntType >::k ( ) const  
[inline]`

Return the  $k$  parameter of the distribution.

Definition at line 3848 of file random.h.

**5.563.3.2** `template<typename _IntType = int> result_type  
std::negative_binomial_distribution< _IntType >::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3884 of file random.h.

**5.563.3.3** `template<typename _IntType = int> result_type  
std::negative_binomial_distribution< _IntType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3877 of file random.h.

**5.563.3.4** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > negative_  
binomial_distribution< _IntType >::result_type  
std::negative_binomial_distribution< _IntType >::operator() (  
_UniformRandomNumberGenerator & __urng )`

Generating functions.

Definition at line 1041 of file random.tcc.

**5.563.3.5** `template<typename _IntType = int> double  
std::negative_binomial_distribution< _IntType >::p ( ) const  
[inline]`

Return the  $p$  parameter of the distribution.

Definition at line 3855 of file random.h.

**5.563.3.6** `template<typename _IntType = int> param_type  
std::negative_binomial_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3862 of file random.h.

**5.563.3.7** `template<typename _IntType = int> void std::negative_binomial_  
distribution< _IntType >::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3870 of file random.h.

**5.563.3.8** `template<typename _IntType = int> void  
std::negative_binomial_distribution< _IntType >::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 3841 of file random.h.

## 5.563 `std::negative_binomial_distribution< _IntType >` Class Template Reference

2707

References `std::gamma_distribution< _RealType >::reset()`.

### 5.563.4 Friends And Related Function Documentation

**5.563.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > &  
, const std::negative_binomial_distribution<_IntType1 > & )  
[friend]`

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `negative_binomial_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.563.4.2** `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::negative_binomial_distribution<  
_IntType1 > & __d1, const std::negative_binomial_distribution<  
_IntType1 > & __d2 ) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3906 of file `random.h`.

**5.563.4.3** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & ,  
std::negative_binomial_distribution<_IntType1 > & ) [friend]`

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `negative_binomial_distribution` random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.564 `std::negative_binomial_distribution< _IntType >::param_type` Struct Reference

**Public Types**

- typedef [negative\\_binomial\\_distribution< \\_IntType >](#) **distribution\_type**

**Public Member Functions**

- **param\_type** (`_IntType __k=1, double __p=0.5`)
- `_IntType k` () const
- `double p` () const

**Friends**

- `bool operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.564.1 Detailed Description

```
template<typename _IntType = int> struct std::negative_binomial_ -
distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3801 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.565 std::nested\_exception Class Reference

Exception class with exception\_ptr data member.

Inherited by std::\_Nested\_exception< \_Except >.

### Public Member Functions

- **nested\_exception** (const [nested\\_exception](#) &)
- exception\_ptr **nested\_ptr** () const
- [nested\\_exception](#) & **operator=** (const [nested\\_exception](#) &)
- void **rethrow\_nested** () const \_\_attribute\_\_((\_\_noreturn\_\_))

### 5.565.1 Detailed Description

Exception class with exception\_ptr data member.

Definition at line 55 of file nested\_exception.h.

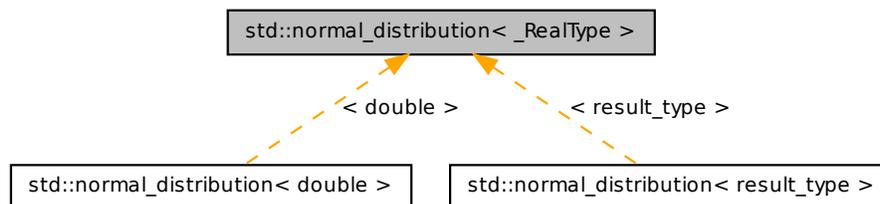
The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 5.566 std::normal\_distribution< \_RealType > Class Template Reference

A normal continuous distribution for random numbers.

Inheritance diagram for std::normal\_distribution< \_RealType >:



## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [normal\\_distribution](#) ([result\\_type](#) \_\_mean=[result\\_type](#)(0), [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- **normal\_distribution** (const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- `_RealType` mean () const
- [result\\_type](#) min () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) operator() (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) operator() (`_UniformRandomNumberGenerator` &\_\_urng)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type](#) param () const
- void reset ()
- `_RealType` stddev () const

## Friends

- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
`std::basic_ostream`< `_CharT`, `_Traits` > & operator<< (`std::basic_ostream`< `_CharT`, `_Traits` > &, const `std::normal_distribution`< `_RealType1` > &)
- template<typename `_RealType1` >  
bool operator== (const `std::normal_distribution`< `_RealType1` > &\_\_d1, const `std::normal_distribution`< `_RealType1` > &\_\_d2)
- template<typename `_RealType1` , typename `_CharT` , typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & operator>> (`std::basic_istream`< `_CharT`, `_Traits` > &, `std::normal_distribution`< `_RealType1` > &)

### 5.566.1 Detailed Description

**template<typename \_RealType = double> class std::normal\_distribution<\_RealType>**

A normal continuous distribution for random numbers. The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1990 of file random.h.

### 5.566.2 Member Typedef Documentation

**5.566.2.1 template<typename \_RealType = double> typedef \_RealType std::normal\_distribution<\_RealType>::result\_type**

The type of the range of the distribution.

Definition at line 1997 of file random.h.

### 5.566.3 Constructor & Destructor Documentation

**5.566.3.1 template<typename \_RealType = double> std::normal\_distribution<\_RealType>::normal\_distribution ( result\_type \_\_mean = result\_type (0), result\_type \_\_stddev = result\_type (1) ) [inline, explicit]**

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2035 of file random.h.

### 5.566.4 Member Function Documentation

**5.566.4.1 template<typename \_RealType = double> result\_type std::normal\_distribution<\_RealType>::max ( ) const [inline]**

Returns the least upper bound value of the distribution.

Definition at line 2092 of file random.h.

Referenced by std::normal\_distribution<result\_type>::max().

**5.566.4.2** `template<typename _RealType = double> _RealType  
std::normal_distribution< _RealType >::mean ( ) const  
[inline]`

Returns the mean of the distribution.

Definition at line 2056 of file random.h.

**5.566.4.3** `template<typename _RealType = double> result_type  
std::normal_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2085 of file random.h.

Referenced by `std::normal_distribution< result_type >::min()`.

**5.566.4.4** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::normal_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2100 of file random.h.

Referenced by `std::normal_distribution< result_type >::operator()`.

**5.566.4.5** `template<typename _RealType > template<typename  
_UniformRandomNumberGenerator > normal_distribution<  
_RealType >::result_type std::normal_distribution< _RealType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1607 of file random.tcc.

References `std::log()`, and `std::sqrt()`.

**5.566.4.6** `template<typename _RealType = double> void  
std::normal_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

**Parameters**

`__param` The new parameter set of the distribution.

Definition at line 2078 of file `random.h`.

**5.566.4.7** `template<typename _RealType = double> param_type  
std::normal_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2070 of file `random.h`.

Referenced by `std::normal_distribution<result_type>::operator()`.

**5.566.4.8** `template<typename _RealType = double> void  
std::normal_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2049 of file `random.h`.

Referenced by `std::poisson_distribution<_IntType>::reset()`, `std::binomial_distribution<_IntType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::gamma_distribution<result_type>::reset()`, and `std::lognormal_distribution<_RealType>::reset()`.

**5.566.4.9** `template<typename _RealType = double> _RealType  
std::normal_distribution<_RealType>::stddev ( ) const  
[inline]`

Returns the standard deviation of the distribution.

Definition at line 2063 of file `random.h`.

## 5.566.5 Friends And Related Function Documentation

**5.566.5.1** `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & , const std::normal_distribution< _RealType1 > & ) [friend]`

Inserts a normal\_distribution random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A normal\_distribution random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.566.5.2** `template<typename _RealType = double> template<typename _RealType1 > bool operator==( const std::normal_distribution< _RealType1 > & __d1, const std::normal_distribution< _RealType1 > & __d2 ) [friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

**5.566.5.3** `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & , std::normal_distribution< _RealType1 > & ) [friend]`

Extracts a normal\_distribution random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A normal\_distribution random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

## 5.567 std::normal\_distribution<\_RealType>::param\_type Struct Reference

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.567 std::normal\_distribution<\_RealType>::param\_type Struct Reference

### Public Types

- typedef [normal\\_distribution<\\_RealType>](#) **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_mean=\_RealType(0), \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.567.1 Detailed Description

**template<typename \_RealType = double> struct std::normal\_distribution<\_RealType>::param\_type**

Parameter type.

Definition at line 1999 of file [random.h](#).

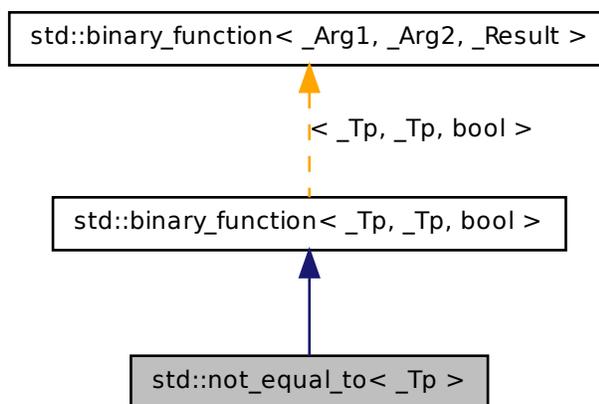
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.568 std::not\_equal\_to<\_Tp> Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::not_equal_to<_Tp >`:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.568.1 Detailed Description

`template<typename _Tp> struct std::not_equal_to<_Tp >`

One of the [comparison functors](#).

Definition at line 208 of file `stl_function.h`.

## 5.568.2 Member Typedef Documentation

**5.568.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.568.2.2** `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.568.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

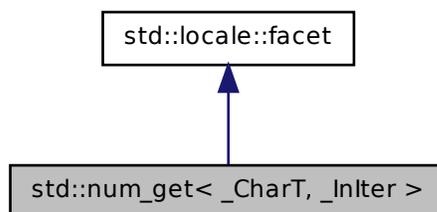
- [stl\\_function.h](#)

## 5.569 `std::num_get< _CharT, _InIter >` Class Template Reference

Primary class template [num\\_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

Inheritance diagram for `std::num_get<_CharT, _InIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

## Public Member Functions

- `num_get` (`size_t __refs=0`)
- `template<typename _ValueT >`  
`_InIter _M_extract_int` (`_InIter __beg, _InIter __end, ios_base &__io, ios_base::iostate &__err, _ValueT &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void *&__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v`) `const`

- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `~num_get ()`
- `iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT > iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if<!__is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 *__zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, float &) const`

- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, void *&) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.569.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::num_get< _CharT, _InIter >`

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators. The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

Definition at line 1909 of file `locale_facets.h`.

### 5.569.2 Member Typedef Documentation

**5.569.2.1** `template<typename _CharT , typename _InIter > typedef _CharT std::num_get< _CharT, _InIter >::char_type`

Public typedefs.

Definition at line 1915 of file `locale_facets.h`.

### 5.569.2.2 `template<typename _CharT, typename _InIter > typedef _InIter std::num_get<_CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1916 of file `locale_facets.h`.

## 5.569.3 Constructor & Destructor Documentation

### 5.569.3.1 `template<typename _CharT, typename _InIter > std::num_get< _CharT, _InIter >::num_get ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 1930 of file `locale_facets.h`.

### 5.569.3.2 `template<typename _CharT, typename _InIter > virtual std::num_get<_CharT, _InIter >::~~num_get ( ) [inline, protected, virtual]`

Destructor.

Definition at line 2099 of file `locale_facets.h`.

## 5.569.4 Member Function Documentation

### 5.569.4.1 `template<typename _CharT, typename _InIter > _InIter std::num_get<_CharT, _InIter >::do_get ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 686 of file locale\_facets.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.

**5.569.4.2** `template<typename _CharT, typename _InIter > _InIter  
std::num_get<_CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, bool &  
__v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 590 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get<_CharT, _InIter >::get()`.

```
5.569.4.3 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long &
__v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2167 of file locale\_facets.h.

```
5.569.4.4 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned short & __v ) const [inline, protected,
virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2172 of file locale\_facets.h.

```
5.569.4.5 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned int & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

### See also

[get\(\)](#) for more details.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2177 of file locale\_facets.h.

```
5.569.4.6 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned long & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2182 of file `locale_facets.h`.

```
5.569.4.7 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get<_CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
long & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2188 of file `locale_facets.h`.

```
5.569.4.8 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned long long & __v ) const [inline, protected,
virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2193 of file locale\_facets.h.

```
5.569.4.9 template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long
double & __v ) const [protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.  
*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 733 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

```
5.569.4.10 template<typename _CharT , typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
double & __v ) const [protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

### See also

[get\(\)](#) for more details.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 701 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

**5.569.4.11** `template<typename _CharT , typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, void  
*& __v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 748 of file locale\_facets.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.

**5.569.4.12** `template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, long & __v )  
const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets err to [ios\\_base::eofbit](#) if the stream is emptied.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 1992 of file locale\_facets.h.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

```
5.569.4.13 template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, double & __v )  
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

The input characters are parsed like the scanf *g* specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal\\_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets err to [ios\\_base::eofbit](#) if the stream is emptied.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.  
*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2056 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.14 template<typename _CharT, typename _InIter > iter_type  
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, void *& __v )  
const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` *p* specifier.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

### Parameters

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2093 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.15 template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios\\_base::basefield](#). If equal to [ios\\_base::oct](#), parses like the `scanf` `o` specifier. Else if equal to [ios\\_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets `err` to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets `err` to [ios\\_base::eofbit](#) if the stream is emptied.

#### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2007 of file `locale_facets.h`.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

```
5.569.4.16 template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, bool & __v )
const [inline]
```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 1956 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

Referenced by `std::basic_istream<_CharT, _Traits >::operator>>()`.

```
5.569.4.17 template<typename _CharT , typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for  $v$ ,  $v$  is set. Otherwise, sets `err` to `ios_base::failbit` and leaves  $v$  unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2018 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.18 template<typename _CharT , typename _InIter > iter_type  
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, float & __v )  
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable  $v$ . It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for  $v$ ,  $v$  is set. Otherwise, sets `err` to `ios_base::failbit` and leaves  $v$  unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2051 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.19 template<typename _CharT , typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned int &
__v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2002 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.20 template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, long double &  
__v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

The input characters are parsed like the scanf *g* specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal\\_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets *err* to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets *err* to [ios\\_base::eofbit](#) if the stream is emptied.

#### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2061 of file locale\_facets.h.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

```
5.569.4.21 template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, long long & __v  
) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2013 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.569.4.22 template<typename _CharT , typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned short
& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets err to [ios\\_base::eofbit](#) if the stream is emptied.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 1997 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

## 5.569.5 Member Data Documentation

### 5.569.5.1 `template<typename _CharT, typename _InIter > locale::id` `std::num_get< _CharT, _InIter >::id` [static]

Numpunct facet id.

Definition at line 1920 of file `locale_facets.h`.

The documentation for this class was generated from the following files:

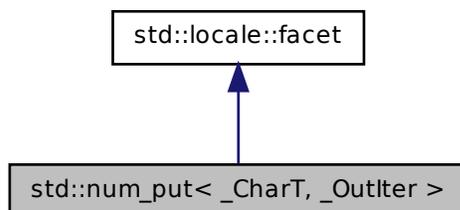
- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

## 5.570 `std::num_put< _CharT, _OutIter >` Class Template Reference

Primary class template [num\\_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

Inheritance diagram for `std::num_put<_CharT, _OutIter >`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

## Public Member Functions

- [num\\_put](#) (`size_t __refs=0`)
- `template<typename _ValueT >`  
`_OutIter \_M\_insert\_float (_OutIter __s, ios\_base &__io, _CharT __fill, char __mod, _ValueT __v) const`
- `template<typename _ValueT >`  
`_OutIter \_M\_insert\_int (_OutIter __s, ios\_base &__io, _CharT __fill, _ValueT __v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, const void *__v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, bool __v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, long __v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, unsigned long __v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, long long __v) const`
- `iter\_type put (iter\_type __s, ios\_base &__f, char_type __fill, unsigned long long __v) const`

- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, double __v) const`
- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, long double __v) const`

## Static Public Attributes

- static `locale::id id`

## Protected Member Functions

- virtual `~num_put ()`
- void `_M_group_float (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int &__len) const`
- void `_M_group_int (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type *__new, char_type *__cs, int &__len) const`
- `template<typename _ValueT >`  
`iter_type _M_insert_float (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const`
- `template<typename _ValueT >`  
`iter_type _M_insert_int (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const`
- void `_M_pad (char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, bool __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, long double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, const void *__v) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.570.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::num_put< _CharT, _OutIter >`

Primary class template `num_put`.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The `num_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_put` facet.

Definition at line 2247 of file `locale_facets.h`.

### 5.570.2 Member Typedef Documentation

**5.570.2.1** `template<typename _CharT , typename _OutIter > typedef _CharT std::num_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 2253 of file `locale_facets.h`.

**5.570.2.2** `template<typename _CharT , typename _OutIter > typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 2254 of file `locale_facets.h`.

### 5.570.3 Constructor & Destructor Documentation

**5.570.3.1** `template<typename _CharT, typename _OutIter> std::num_put<_CharT, _OutIter>::num_put( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 2268 of file `locale_facets.h`.

**5.570.3.2** `template<typename _CharT, typename _OutIter> virtual std::num_put<_CharT, _OutIter>::~~num_put( ) [inline, protected, virtual]`

Destructor.

Definition at line 2447 of file `locale_facets.h`.

### 5.570.4 Member Function Documentation

**5.570.4.1** `template<typename _CharT, typename _OutIter> virtual iter_type std::num_put<_CharT, _OutIter>::do_put( iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const [inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* `Char_type` to use for filling.

*v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2471 of file `locale_facets.h`.

```
5.570.4.2 template<typename _CharT , typename _OutIter > virtual
iter_type std::num_put< _CharT, _OutIter >::do_put ( iter_type
__s, ios_base & __io, char_type __fill, long long __v ) const
[inline, protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2477 of file locale\_facets.h.

```
5.570.4.3 template<typename _CharT , typename _OutIter > virtual iter_type
std::num_put< _CharT, _OutIter >::do_put ( iter_type __s,
ios_base & __io, char_type __fill, long __v ) const [inline,
protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2467 of file locale\_facets.h.

**5.570.4.4** `template<typename _CharT, typename _OutIter> virtual iter_type  
std::num_put<_CharT, _OutIter>::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, unsigned long long __v ) const  
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2482 of file locale\_facets.h.

**5.570.4.5** `template<typename _CharT, typename _OutIter> _OutIter  
std::num_put<_CharT, _OutIter>::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, bool __v ) const [protected,  
virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1089 of file locale\_facets.tcc.

References `std::ios_base::M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

Referenced by `std::num_put<_CharT, _OutIter>::put()`.

```
5.570.4.6 template<typename _CharT, typename _OutIter > _OutIter  
std::num_put< _CharT, _OutIter >::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, long double __v ) const  
[protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1155 of file locale\_facets.tcc.

```
5.570.4.7 template<typename _CharT, typename _OutIter > _OutIter  
std::num_put< _CharT, _OutIter >::do_put ( iter_type  
__s, ios_base & __io, char_type __fill, double __v ) const  
[protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1141 of file locale\_facets.tcc.

**5.570.4.8** `template<typename _CharT, typename _OutIter > _OutIter  
std::num_put<_CharT, _OutIter >::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, const void * __v ) const  
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* `Char_type` to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1162 of file `locale_facets.tcc`.

References `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::uppercase`.

**5.570.4.9** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base &  
__f, char_type __fill, long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2328 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

```
5.570.4.10 template<typename _CharT, typename _OutIter > iter_type
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base
& __f, char_type __fill, long long __v ) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2338 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

```
5.570.4.11 template<typename _CharT , typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, unsigned long __v ) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.

*v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2332 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**5.570.4.12** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, bool __v ) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats *v* as an int.

### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2286 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**5.570.4.13** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, long double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2395 of file `locale_facets.h`.

References `std::num_put< _CharT, _OutIter >::do_put()`.

```
5.570.4.14 template<typename _CharT , typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base
& __f, char_type __fill, const void * __v ) const [inline]
```

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats *v* as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2416 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

```
5.570.4.15 template<typename _CharT, typename _OutIter > iter_type
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base
& __f, char_type __fill, unsigned long long __v ) const
[inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2342 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**5.570.4.16** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put<_CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on `uppercase`. Note that if both `fixed` and `scientific` are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

*s* Stream to write to.

*io* Source of locale and flags.

*fill* Char\_type to use for filling.

*v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2391 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

## 5.570.5 Member Data Documentation

### 5.570.5.1 `template<typename _CharT, typename _OutIter > locale::id` `std::num_put<_CharT, _OutIter >::id` [static]

Numpunct facet id.

Definition at line 2258 of file locale\_facets.h.

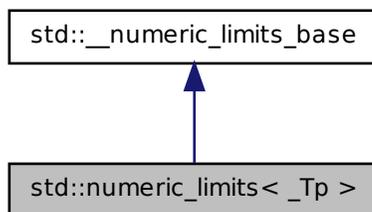
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

## 5.571 `std::numeric_limits<_Tp >` Struct Template Reference

Properties of fundamental types.

Inheritance diagram for `std::numeric_limits<_Tp >`:



## Static Public Member Functions

- static `_Tp` [denorm\\_min](#) () throw ()
- static `_Tp` [epsilon](#) () throw ()
- static `_Tp` [infinity](#) () throw ()
- static `_Tp` [lowest](#) () throw ()
- static `_Tp` [max](#) () throw ()
- static `_Tp` [min](#) () throw ()
- static `_Tp` [quiet\\_NaN](#) () throw ()
- static `_Tp` [round\\_error](#) () throw ()
- static `_Tp` [signaling\\_NaN](#) () throw ()

## Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const `float_denorm_style` [has\\_denorm](#)
- static const bool [has\\_denorm\\_loss](#)
- static const bool [has\\_infinity](#)
- static const bool [has\\_quiet\\_NaN](#)
- static const bool [has\\_signaling\\_NaN](#)
- static const bool [is\\_bounded](#)
- static const bool [is\\_exact](#)
- static const bool [is\\_iec559](#)
- static const bool [is\\_integer](#)
- static const bool [is\\_modulo](#)
- static const bool [is\\_signed](#)
- static const bool [is\\_specialized](#)
- static const int [max\\_digits10](#)
- static const int [max\\_exponent](#)
- static const int [max\\_exponent10](#)
- static const int [min\\_exponent](#)
- static const int [min\\_exponent10](#)
- static const int [radix](#)
- static const `float_round_style` [round\\_style](#)
- static const bool [tinyness\\_before](#)
- static const bool [traps](#)

### 5.571.1 Detailed Description

**template<typename \_Tp> struct std::numeric\_limits<\_Tp >**

Properties of fundamental types. This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS`: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 284 of file `limits`.

### 5.571.2 Member Function Documentation

**5.571.2.1 template<typename \_Tp > static \_Tp std::numeric\_limits<\_Tp >::denorm\_min( ) throw () [inline, static]**

The minimum positive denormalized value. For types where `has_denorm` is `false`, this is the minimum positive normalized value.

Definition at line 313 of file `limits`.

**5.571.2.2 template<typename \_Tp > static \_Tp std::numeric\_limits<\_Tp >::epsilon( ) throw () [inline, static]**

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 298 of file `limits`.

**5.571.2.3 template<typename \_Tp > static \_Tp std::numeric\_limits<\_Tp >::infinity( ) throw () [inline, static]**

The representation of positive infinity, if `has_infinity`.

Definition at line 302 of file `limits`.

**5.571.2.4 template<typename \_Tp > static \_Tp std::numeric\_limits<\_Tp >::lowest( ) throw () [inline, static]**

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 294 of file `limits`.

**5.571.2.5** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::max( ) throw () [inline, static]`

The maximum finite value.

Definition at line 290 of file limits.

**5.571.2.6** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::min( ) throw () [inline, static]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 288 of file limits.

**5.571.2.7** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::quiet_NaN( ) throw () [inline, static]`

The representation of a quiet *Not a Number*, if has\_quiet\_NaN.

Definition at line 306 of file limits.

**5.571.2.8** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::round_error( ) throw () [inline, static]`

The maximum rounding error measurement (see LIA-1).

Definition at line 300 of file limits.

**5.571.2.9** `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::signaling_NaN( ) throw () [inline, static]`

The representation of a signaling *Not a Number*, if has\_signaling\_NaN.

Definition at line 309 of file limits.

### 5.571.3 Member Data Documentation

**5.571.3.1** `const int std::_numeric_limits_base::digits [static, inherited]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file limits.

**5.571.3.2** `const int std::__numeric_limits_base::digits10` [**static, inherited**]

The number of base 10 digits that can be represented without change.

Definition at line 201 of file limits.

**5.571.3.3** `const float_denorm_style std::__numeric_limits_base::has_denorm` [**static, inherited**]

See [std::float\\_denorm\\_style](#) for more information.

Definition at line 244 of file limits.

**5.571.3.4** `const bool std::__numeric_limits_base::has_denorm_loss` [**static, inherited**]

*True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42*

Definition at line 247 of file limits.

**5.571.3.5** `const bool std::__numeric_limits_base::has_infinity` [**static, inherited**]

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

**5.571.3.6** `const bool std::__numeric_limits_base::has_quiet_NaN` [**static, inherited**]

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

**5.571.3.7** `const bool std::__numeric_limits_base::has_signaling_NaN` [**static, inherited**]

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

**5.571.3.8 const bool std::\_\_numeric\_limits\_base::is\_bounded [static, inherited]**

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54

Definition at line 255 of file limits.

**5.571.3.9 const bool std::\_\_numeric\_limits\_base::is\_exact [static, inherited]**

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15

Definition at line 216 of file limits.

**5.571.3.10 const bool std::\_\_numeric\_limits\_base::is\_iec559 [static, inherited]**

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

**5.571.3.11 const bool std::\_\_numeric\_limits\_base::is\_integer [static, inherited]**

True if the type is integer. Is this supposed to be *if the type is integral?*

Definition at line 211 of file limits.

**5.571.3.12 const bool std::\_\_numeric\_limits\_base::is\_modulo [static, inherited]**

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

**5.571.3.13 const bool std::\_\_numeric\_limits\_base::is\_signed [static, inherited]**

True if the type is signed.

Definition at line 208 of file limits.

**5.571.3.14** `const bool std::__numeric_limits_base::is_specialized [static, inherited]`

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

**5.571.3.15** `const int std::__numeric_limits_base::max_digits10 [static, inherited]`

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

**5.571.3.16** `const int std::__numeric_limits_base::max_exponent [static, inherited]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file limits.

**5.571.3.17** `const int std::__numeric_limits_base::max_exponent10 [static, inherited]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file limits.

**5.571.3.18** `const int std::__numeric_limits_base::min_exponent [static, inherited]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file limits.

**5.571.3.19** `const int std::__numeric_limits_base::min_exponent10` [`static`, `inherited`]

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file `limits`.

**5.571.3.20** `const int std::__numeric_limits_base::radix` [`static`, `inherited`]

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file `limits`.

**5.571.3.21** `const float_round_style std::__numeric_limits_base::round_style` [`static`, `inherited`]

See [std::float\\_round\\_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file `limits`.

**5.571.3.22** `const bool std::__numeric_limits_base::tinyness_before` [`static`, `inherited`]

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file `limits`.

**5.571.3.23** `const bool std::__numeric_limits_base::traps` [`static`, `inherited`]

True if trapping is implemented for this type.

Definition at line 263 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.572 `std::numeric_limits< bool >` Struct Template Reference

[numeric\\_limits<bool>](#) specialization.

### Static Public Member Functions

- static bool **denorm\_min** () throw ()
- static bool **epsilon** () throw ()
- static bool **infinity** () throw ()
- static bool **lowest** () throw ()
- static bool **max** () throw ()
- static bool **min** () throw ()
- static bool **quiet\_NaN** () throw ()
- static bool **round\_error** () throw ()
- static bool **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.572.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

[numeric\\_limits<bool>](#) specialization.

Definition at line 335 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.573 `std::numeric_limits< char >` Struct Template Reference

[numeric\\_limits<char>](#) specialization.

### Static Public Member Functions

- static char **denorm\_min** () throw ()
- static char **epsilon** () throw ()
- static char **infinity** () throw ()
- static char **lowest** () throw ()
- static char **max** () throw ()
- static char **min** () throw ()
- static char **quiet\_NaN** () throw ()
- static char **round\_error** () throw ()
- static char **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.573.1 Detailed Description

`template<> struct std::numeric_limits< char >`

[numeric\\_limits<char>](#) specialization.

Definition at line 395 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.574 `std::numeric_limits< char16_t >` Struct Template Reference

[numeric\\_limits<char16\\_t>](#) specialization.

### Static Public Member Functions

- static `char16_t` **denorm\_min** () throw ()
- static `char16_t` **epsilon** () throw ()
- static `char16_t` **infinity** () throw ()
- static `char16_t` **lowest** () throw ()
- static `char16_t` **max** () throw ()
- static `char16_t` **min** () throw ()
- static `char16_t` **quiet\_NaN** () throw ()
- static `char16_t` **round\_error** () throw ()
- static `char16_t` **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.574.1 Detailed Description

`template<> struct std::numeric_limits< char16_t >`

[numeric\\_limits<char16\\_t>](#) specialization.

Definition at line 628 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.575 `std::numeric_limits< char32_t >` Struct Template Reference

[numeric\\_limits<char32\\_t>](#) specialization.

## Static Public Member Functions

- static char32\_t **denorm\_min** () throw ()
- static char32\_t **epsilon** () throw ()
- static char32\_t **infinity** () throw ()
- static char32\_t **lowest** () throw ()
- static char32\_t **max** () throw ()
- static char32\_t **min** () throw ()
- static char32\_t **quiet\_NaN** () throw ()
- static char32\_t **round\_error** () throw ()
- static char32\_t **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.575.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

[numeric\\_limits<char32\\_t>](#) specialization.

Definition at line 686 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.576 `std::numeric_limits< double >` Struct Template Reference

[numeric\\_limits<double>](#) specialization.

### Static Public Member Functions

- static double **denorm\_min** () throw ()
- static double **epsilon** () throw ()
- static double **infinity** () throw ()
- static double **lowest** () throw ()
- static double **max** () throw ()
- static double **min** () throw ()
- static double **quiet\_NaN** () throw ()
- static double **round\_error** () throw ()
- static double **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.576.1 Detailed Description

`template<> struct std::numeric_limits< double >`

[numeric\\_limits<double>](#) specialization.

Definition at line 1274 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.577 `std::numeric_limits< float >` Struct Template Reference

[numeric\\_limits<float>](#) specialization.

### Static Public Member Functions

- static float **denorm\_min** () throw ()
- static float **epsilon** () throw ()
- static float **infinity** () throw ()
- static float **lowest** () throw ()
- static float **max** () throw ()
- static float **min** () throw ()
- static float **quiet\_NaN** () throw ()
- static float **round\_error** () throw ()
- static float **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.577.1 Detailed Description

`template<> struct std::numeric_limits< float >`

[numeric\\_limits<float>](#) specialization.

Definition at line 1209 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.578 `std::numeric_limits< int >` Struct Template Reference

[numeric\\_limits<int>](#) specialization.

## Static Public Member Functions

- static int **denorm\_min** () throw ()
- static int **epsilon** () throw ()
- static int **infinity** () throw ()
- static int **lowest** () throw ()
- static int **max** () throw ()
- static int **min** () throw ()
- static int **quiet\_NaN** () throw ()
- static int **round\_error** () throw ()
- static int **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.578.1 Detailed Description

`template<> struct std::numeric_limits< int >`

[numeric\\_limits<int>](#) specialization.

Definition at line 861 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.579 `std::numeric_limits< long >` Struct Template Reference

[numeric\\_limits<long>](#) specialization.

### Static Public Member Functions

- static long **denorm\_min** () throw ()
- static long **epsilon** () throw ()
- static long **infinity** () throw ()
- static long **lowest** () throw ()
- static long **max** () throw ()
- static long **min** () throw ()
- static long **quiet\_NaN** () throw ()
- static long **round\_error** () throw ()
- static long **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.579.1 Detailed Description

`template<> struct std::numeric_limits< long >`

[numeric\\_limits<long>](#) specialization.

Definition at line 977 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.580 `std::numeric_limits< long double >` Struct Template Reference

[numeric\\_limits<long double>](#) specialization.

### Static Public Member Functions

- static long double **denorm\_min** () throw ()
- static long double **epsilon** () throw ()
- static long double **infinity** () throw ()
- static long double **lowest** () throw ()
- static long double **max** () throw ()
- static long double **min** () throw ()
- static long double **quiet\_NaN** () throw ()
- static long double **round\_error** () throw ()
- static long double **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.580.1 Detailed Description

`template<> struct std::numeric_limits< long double >`

[numeric\\_limits<long double>](#) specialization.

Definition at line 1339 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.581 `std::numeric_limits< long long >` Struct Template Reference

[numeric\\_limits<long long>](#) specialization.

## Static Public Member Functions

- static long long **denorm\_min** () throw ()
- static long long **epsilon** () throw ()
- static long long **infinity** () throw ()
- static long long **lowest** () throw ()
- static long long **max** () throw ()
- static long long **min** () throw ()
- static long long **quiet\_NaN** () throw ()
- static long long **round\_error** () throw ()
- static long long **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.581.1 Detailed Description

`template<> struct std::numeric_limits< long long >`

[numeric\\_limits<long long>](#) specialization.

Definition at line 1093 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.582 `std::numeric_limits< short >` Struct Template Reference

[numeric\\_limits<short>](#) specialization.

### Static Public Member Functions

- static short `denorm_min` () throw ()
- static short `epsilon` () throw ()
- static short `infinity` () throw ()
- static short `lowest` () throw ()
- static short `max` () throw ()
- static short `min` () throw ()
- static short `quiet_NaN` () throw ()
- static short `round_error` () throw ()
- static short `signaling_NaN` () throw ()

### Static Public Attributes

- static const int `digits`
- static const int `digits10`
- static const `float_denorm_style` `has_denorm`
- static const bool `has_denorm_loss`
- static const bool `has_infinity`
- static const bool `has_quiet_NaN`
- static const bool `has_signaling_NaN`
- static const bool `is_bounded`
- static const bool `is_exact`
- static const bool `is_iec559`
- static const bool `is_integer`

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.582.1 Detailed Description

`template<> struct std::numeric_limits< short >`

[numeric\\_limits<short>](#) specialization.

Definition at line 745 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.583 `std::numeric_limits< signed char >` Struct Template Reference

[numeric\\_limits<signed char>](#) specialization.

#### Static Public Member Functions

- static signed char **denorm\_min** () throw ()
- static signed char **epsilon** () throw ()
- static signed char **infinity** () throw ()
- static signed char **lowest** () throw ()
- static signed char **max** () throw ()
- static signed char **min** () throw ()
- static signed char **quiet\_NaN** () throw ()
- static signed char **round\_error** () throw ()
- static signed char **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int `digits`
- static const int `digits10`
- static const `float_denorm_style` `has_denorm`
- static const bool `has_denorm_loss`
- static const bool `has_infinity`
- static const bool `has_quiet_NaN`
- static const bool `has_signaling_NaN`
- static const bool `is_bounded`
- static const bool `is_exact`
- static const bool `is_iec559`
- static const bool `is_integer`
- static const bool `is_modulo`
- static const bool `is_signed`
- static const bool `is_specialized`
- static const int `max_digits10`
- static const int `max_exponent`
- static const int `max_exponent10`
- static const int `min_exponent`
- static const int `min_exponent10`
- static const int `radix`
- static const `float_round_style` `round_style`
- static const bool `tinyness_before`
- static const bool `traps`

### 5.583.1 Detailed Description

`template<> struct std::numeric_limits< signed char >`

[numeric\\_limits<signed char>](#) specialization.

Definition at line 453 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.584 `std::numeric_limits< unsigned char >` Struct Template Reference

[numeric\\_limits<unsigned char>](#) specialization.

## Static Public Member Functions

- static unsigned char **denorm\_min** () throw ()
- static unsigned char **epsilon** () throw ()
- static unsigned char **infinity** () throw ()
- static unsigned char **lowest** () throw ()
- static unsigned char **max** () throw ()
- static unsigned char **min** () throw ()
- static unsigned char **quiet\_NaN** () throw ()
- static unsigned char **round\_error** () throw ()
- static unsigned char **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.584.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

[numeric\\_limits<unsigned char>](#) specialization.

Definition at line 511 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.585 `std::numeric_limits< unsigned int >` Struct Template Reference

[numeric\\_limits<unsigned int>](#) specialization.

### Static Public Member Functions

- static unsigned int **denorm\_min** () throw ()
- static unsigned int **epsilon** () throw ()
- static unsigned int **infinity** () throw ()
- static unsigned int **lowest** () throw ()
- static unsigned int **max** () throw ()
- static unsigned int **min** () throw ()
- static unsigned int **quiet\_NaN** () throw ()
- static unsigned int **round\_error** () throw ()
- static unsigned int **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.585.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

[numeric\\_limits<unsigned int>](#) specialization.

Definition at line 919 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.586 `std::numeric_limits< unsigned long >` Struct Template Reference

[numeric\\_limits<unsigned long>](#) specialization.

#### Static Public Member Functions

- static unsigned long **denorm\_min** () throw ()
- static unsigned long **epsilon** () throw ()
- static unsigned long **infinity** () throw ()
- static unsigned long **lowest** () throw ()
- static unsigned long **max** () throw ()
- static unsigned long **min** () throw ()
- static unsigned long **quiet\_NaN** () throw ()
- static unsigned long **round\_error** () throw ()
- static unsigned long **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.586.1 Detailed Description

**template<> struct std::numeric\_limits< unsigned long >**

[numeric\\_limits<unsigned long>](#) specialization.

Definition at line 1035 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## **5.587 std::numeric\_limits< unsigned long long > Struct Template Reference**

[numeric\\_limits<unsigned long long>](#) specialization.

## Static Public Member Functions

- static unsigned long long **denorm\_min** () throw ()
- static unsigned long long **epsilon** () throw ()
- static unsigned long long **infinity** () throw ()
- static unsigned long long **lowest** () throw ()
- static unsigned long long **max** () throw ()
- static unsigned long long **min** () throw ()
- static unsigned long long **quiet\_NaN** () throw ()
- static unsigned long long **round\_error** () throw ()
- static unsigned long long **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.587.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

[numeric\\_limits<unsigned long long>](#) specialization.

Definition at line 1151 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.588 `std::numeric_limits< unsigned short >` Struct Template Reference

[numeric\\_limits<unsigned short>](#) specialization.

### Static Public Member Functions

- static unsigned short **denorm\_min** () throw ()
- static unsigned short **epsilon** () throw ()
- static unsigned short **infinity** () throw ()
- static unsigned short **lowest** () throw ()
- static unsigned short **max** () throw ()
- static unsigned short **min** () throw ()
- static unsigned short **quiet\_NaN** () throw ()
- static unsigned short **round\_error** () throw ()
- static unsigned short **signaling\_NaN** () throw ()

### Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**

- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.588.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

[numeric\\_limits<unsigned short>](#) specialization.

Definition at line 803 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.589 `std::numeric_limits< wchar_t >` Struct Template Reference

[numeric\\_limits<wchar\\_t>](#) specialization.

### Static Public Member Functions

- static `wchar_t` **denorm\_min** () throw ()
- static `wchar_t` **epsilon** () throw ()
- static `wchar_t` **infinity** () throw ()
- static `wchar_t` **lowest** () throw ()
- static `wchar_t` **max** () throw ()
- static `wchar_t` **min** () throw ()
- static `wchar_t` **quiet\_NaN** () throw ()
- static `wchar_t` **round\_error** () throw ()
- static `wchar_t` **signaling\_NaN** () throw ()

## Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float\\_denorm\\_style](#) **has\_denorm**
- static const bool **has\_denorm\_loss**
- static const bool **has\_infinity**
- static const bool **has\_quiet\_NaN**
- static const bool **has\_signaling\_NaN**
- static const bool **is\_bounded**
- static const bool **is\_exact**
- static const bool **is\_iec559**
- static const bool **is\_integer**
- static const bool **is\_modulo**
- static const bool **is\_signed**
- static const bool **is\_specialized**
- static const int **max\_digits10**
- static const int **max\_exponent**
- static const int **max\_exponent10**
- static const int **min\_exponent**
- static const int **min\_exponent10**
- static const int **radix**
- static const [float\\_round\\_style](#) **round\_style**
- static const bool **tinyness\_before**
- static const bool **traps**

### 5.589.1 Detailed Description

`template<> struct std::numeric_limits<wchar_t>`

[numeric\\_limits<wchar\\_t>](#) specialization.

Definition at line 569 of file `limits`.

The documentation for this struct was generated from the following file:

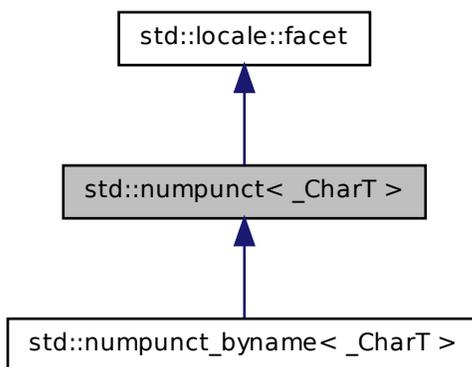
- [limits](#)

## 5.590 `std::num_punct<_CharT>` Class Template Reference

Primary class template `num_punct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

Inheritance diagram for `std::numpunct<_CharT >`:



## Public Types

- typedef `__numpunct_cache<_CharT > __cache_type`
- typedef `_CharT char_type`
- typedef `basic_string<_CharT > string_type`

## Public Member Functions

- `numpunct` (`size_t __refs=0`)
- `numpunct` (`__cache_type *__cache, size_t __refs=0`)
- `numpunct` (`__c_locale __cloc, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~num\\_punct](#) ()
- template<>  
void [\\_M\\_initialize\\_num\\_punct](#) (\_\_c\_locale \_\_cloc)
- template<>  
void [\\_M\\_initialize\\_num\\_punct](#) (\_\_c\_locale \_\_cloc)
- void [\\_M\\_initialize\\_num\\_punct](#) (\_\_c\_locale \_\_cloc=NULL)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static \_GLIBCXX\_CONST const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- [\\_\\_cache\\_type](#) \* [\\_M\\_data](#)

## Friends

- class [locale::\\_Impl](#)

### 5.590.1 Detailed Description

**template<typename \_CharT> class std::numpunct< \_CharT >**

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers. The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

Definition at line 1636 of file locale\_facets.h.

### 5.590.2 Member Typedef Documentation

**5.590.2.1 template<typename \_CharT > typedef \_CharT std::numpunct< \_CharT >::char\_type**

Public typedefs.

Reimplemented in [std::numpunct\\_byname< \\_CharT >](#).

Definition at line 1642 of file locale\_facets.h.

**5.590.2.2 template<typename \_CharT > typedef basic\_string<\_CharT> std::numpunct< \_CharT >::string\_type**

Public typedefs.

Reimplemented in [std::numpunct\\_byname< \\_CharT >](#).

Definition at line 1643 of file locale\_facets.h.

### 5.590.3 Constructor & Destructor Documentation

**5.590.3.1 template<typename \_CharT > std::numpunct< \_CharT >::numpunct ( size\_t \_\_refs = 0 ) [inline, explicit]**

Numpunct constructor.

#### Parameters

*refs* Refcount to pass to the base class.

Definition at line 1660 of file locale\_facets.h.

```
5.590.3.2 template<typename _CharT > std::num_punct< _CharT  
>::num_punct ( __cache_type * __cache, size_t __refs = 0 )  
[inline, explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

#### Parameters

*cache* \_\_num\_punct\_cache object.

*refs* Refcount to pass to the base class.

Definition at line 1673 of file locale\_facets.h.

```
5.590.3.3 template<typename _CharT > std::num_punct< _CharT  
>::num_punct ( __c_locale __cloc, size_t __refs = 0 ) [inline,  
explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

*cloc* The C locale.

*refs* Refcount to pass to the base class.

Definition at line 1687 of file locale\_facets.h.

```
5.590.3.4 template<typename _CharT > virtual std::num_punct< _CharT  
>::~~num_punct ( ) [protected, virtual]
```

Destructor.

## 5.590.4 Member Function Documentation

```
5.590.4.1 template<typename _CharT > char_type std::num_punct< _CharT  
>::decimal_point ( ) const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `numpunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1701 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_decimal_point()`.

**5.590.4.2** `template<typename _CharT > virtual char_type std::numpunct<_CharT >::do_decimal_point ( ) const [inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1788 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::decimal_point()`.

**5.590.4.3** `template<typename _CharT > virtual string_type std::numpunct<_CharT >::do_falsename ( ) const [inline, protected, virtual]`

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1839 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::falsename()`.

**5.590.4.4** `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping( ) const [inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1813 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::grouping()`.

**5.590.4.5** `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep( ) const [inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1800 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.

**5.590.4.6** `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_truename( ) const [inline, protected, virtual]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

`string_type` representing printed form of true.

Definition at line 1826 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::true_name()`.

#### **5.590.4.7** `template<typename _CharT> string_type std::num_punct<_CharT>::false_name ( ) const [inline]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_false_name()`.

#### **Returns**

`string_type` representing printed form of false.

Definition at line 1771 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_false_name()`.

#### **5.590.4.8** `template<typename _CharT> string std::num_punct<_CharT>::grouping ( ) const [inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

#### **Returns**

string representing grouping specification.

Definition at line 1745 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_grouping()`.

---

**5.590.4.9** `template<typename _CharT> char_type std::num_punct<_CharT>::thousands_sep( ) const [inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

Definition at line 1714 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_thousands_sep()`.

**5.590.4.10** `template<typename _CharT> string_type std::num_punct<_CharT>::true_name( ) const [inline]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `num_punct<char_type>::do_true_name()`.

**Returns**

`string_type` representing printed form of true.

Definition at line 1758 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_true_name()`.

**5.590.5 Member Data Documentation****5.590.5.1** `template<typename _CharT> locale::id std::num_punct<_CharT>::id [static]`

Num\_punct facet id.

Definition at line 1652 of file `locale_facets.h`.

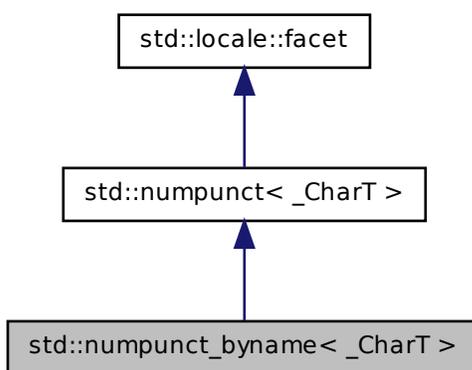
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.591 `std::numpunct_byname<_CharT>` Class Template Reference

class `numpunct_byname` [22.2.3.2].

Inheritance diagram for `std::numpunct_byname<_CharT>`:



### Public Types

- typedef `__numpunct_cache<_CharT>` `__cache_type`
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `numpunct_byname` (`const char *__s`, `size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string_type grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- `template<>`  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- `template<>`  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc=NULL)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

## Static Protected Member Functions

- static `__c_locale` [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static `__c_locale` [\\_S\\_get\\_c\\_locale](#) ()
- static `_GLIBCXX_CONST` const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static `__c_locale` [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__cache_type` \* [\\_M\\_data](#)

## Friends

- class [locale::\\_Impl](#)

### 5.591.1 Detailed Description

`template<typename _CharT> class std::numpunct_byname<_CharT>`

class [numpunct\\_byname](#) [22.2.3.2].

Definition at line 1868 of file [locale\\_facets.h](#).

## 5.591.2 Member Typedef Documentation

### 5.591.2.1 `template<typename _CharT > typedef _CharT std::num_punct_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::num\\_punct< \\_CharT >](#).

Definition at line 1871 of file locale\_facets.h.

### 5.591.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::num_punct_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::num\\_punct< \\_CharT >](#).

Definition at line 1872 of file locale\_facets.h.

## 5.591.3 Member Function Documentation

### 5.591.3.1 `template<typename _CharT > char_type std::num_punct< _CharT >::decimal_point( ) const [inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning [num\\_punct<char\\_type>::do\\_decimal\\_point\(\)](#).

#### Returns

*char\_type* representing a decimal point.

Definition at line 1701 of file locale\_facets.h.

References [std::num\\_punct< \\_CharT >::do\\_decimal\\_point\(\)](#).

### 5.591.3.2 `template<typename _CharT > virtual char_type std::num_punct< _CharT >::do_decimal_point( ) const [inline, protected, virtual, inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1788 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::decimal\_point().

**5.591.3.3** `template<typename _CharT > virtual string_type std::num_punct< _CharT >::do_falsename ( ) const [inline, protected, virtual, inherited]`

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1839 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::falsename().

**5.591.3.4** `template<typename _CharT > virtual string std::num_punct< _CharT >::do_grouping ( ) const [inline, protected, virtual, inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1813 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::grouping().

**5.591.3.5** `template<typename _CharT > virtual char_type std::num_punct<_CharT >::do_thousands_sep( ) const [inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1800 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT >::thousands_sep()`.

**5.591.3.6** `template<typename _CharT > virtual string_type std::num_punct<_CharT >::do_truename( ) const [inline, protected, virtual, inherited]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of true.

Definition at line 1826 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT >::truename()`.

**5.591.3.7** `template<typename _CharT > string_type std::num_punct<_CharT >::do_falsename( ) const [inline, inherited]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1771 of file `locale_facets.h`.

References `std::num_punct<_CharT >::do_falsename()`.

**5.591.3.8** `template<typename _CharT > string std::num_punct< _CharT >::grouping( ) const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the [grouping\(\)](#) returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling [num\\_punct<char\\_type>::do\\_grouping\(\)](#).

**Returns**

string representing grouping specification.

Definition at line 1745 of file locale\_facets.h.

References [std::num\\_punct< \\_CharT >::do\\_grouping\(\)](#).

**5.591.3.9** `template<typename _CharT > char_type std::num_punct< _CharT >::thousands_sep( ) const [inline, inherited]`

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning [num\\_punct<char\\_type>::do\\_thousands\\_sep\(\)](#).

**Returns**

char\_type representing a thousands separator.

Definition at line 1714 of file locale\_facets.h.

References [std::num\\_punct< \\_CharT >::do\\_thousands\\_sep\(\)](#).

**5.591.3.10** `template<typename _CharT > string_type std::num_punct< _CharT >::truename( ) const [inline, inherited]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

**Returns**

`string_type` representing printed form of true.

Definition at line 1758 of file `locale_facets.h`.

References `std::numpunct<_CharT >::do_truename()`.

## 5.591.4 Member Data Documentation

### 5.591.4.1 `template<typename _CharT > locale::id std::numpunct<_CharT >::id [static, inherited]`

Numpunct facet id.

Definition at line 1652 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.592 `std::once_flag` Struct Reference

[once\\_flag](#)

**Public Member Functions**

- `once_flag` (const [once\\_flag](#) &)
- `once_flag` & `operator=` (const [once\\_flag](#) &)

**Friends**

- `template<typename _Callable, typename... _Args>`  
void `call_once` ([once\\_flag](#) &\_\_once, `_Callable` \_\_f, `_Args` &&... \_\_args)

### 5.592.1 Detailed Description

[once\\_flag](#)

Definition at line 676 of file `mutex`.

## 5.593 `std::ostream_iterator< _Tp, _CharT, _Traits >` Class Template Reference 2799

### 5.592.2 Friends And Related Function Documentation

5.592.2.1 `template<typename _Callable, typename... _Args> void call_once`  
( `once_flag & __once, _Callable __f, _Args &&... __args` )  
[`friend`]

`call_once`

Definition at line 722 of file `mutex`.

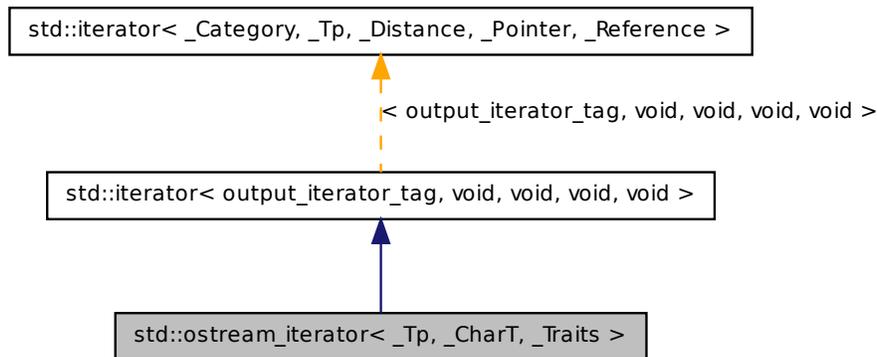
The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.593 `std::ostream_iterator< _Tp, _CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streams.

Inheritance diagram for `std::ostream_iterator< _Tp, _CharT, _Traits >`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) `iterator_category`

- typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

## Public Member Functions

- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s)
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c)
- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj)
- [ostream\\_iterator](#) & **operator\*** ()
- [ostream\\_iterator](#) & **operator++** (int)
- [ostream\\_iterator](#) & **operator++** ()
- [ostream\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_value)

### 5.593.1 Detailed Description

`template<typename \_Tp, typename \_CharT = char, typename \_Traits = char\_traits<\_CharT>> class std::ostream\_iterator< \_Tp, \_CharT, \_Traits >`

Provides output iterator semantics for streams. This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

#### Parameters

***Tp*** The type to write to the ostream.

***CharT*** The ostream `char_type`.

***Traits*** The ostream `char_traits`.

Definition at line 152 of file `stream_iterator.h`.

### 5.593.2 Member Typedef Documentation

**5.593.2.1** `template<typename \_Tp , typename \_CharT = char, typename \_Traits = char\_traits<\_CharT>> typedef \_CharT std::ostream\_iterator< \_Tp, \_CharT, \_Traits >::char\_type`

Public typedef.

Definition at line 158 of file `stream_iterator.h`.

## 5.593 std::ostream\_iterator< \_Tp, \_CharT, \_Traits > Class Template Reference

**5.593.2.2** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 121 of file stl\_iterator\_base\_types.h.

**5.593.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 117 of file stl\_iterator\_base\_types.h.

**5.593.2.4** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 160 of file stream\_iterator.h.

**5.593.2.5** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file stl\_iterator\_base\_types.h.

**5.593.2.6** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

**5.593.2.7** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 159 of file stream\_iterator.h.

**5.593.2.8** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

### 5.593.3 Constructor & Destructor Documentation

**5.593.3.1** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s ) [inline]`

Construct from an ostream.

Definition at line 169 of file `stream_iterator.h`.

**5.593.3.2** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s, const _CharT * __c ) [inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

#### Parameters

*s* Underlying ostream to write to.

*c* CharT delimiter string to insert.

Definition at line 181 of file `stream_iterator.h`.

**5.593.3.3** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( const ostream_iterator< _Tp, _CharT, _Traits > & __obj ) [inline]`

Copy constructor.

Definition at line 185 of file `stream_iterator.h`.

## 5.594 `std::ostreambuf_iterator<_CharT, _Traits >` Class Template Reference 2803

### 5.593.4 Member Function Documentation

**5.593.4.1** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator& std::ostream_iterator<_Tp, _CharT, _Traits >::operator=( const _Tp & __value ) [inline]`

Writes *value* to underlying ostream using operator<<. If /// constructed with delimiter string, writes delimiter to ostream.

Definition at line 191 of file stream\_iterator.h.

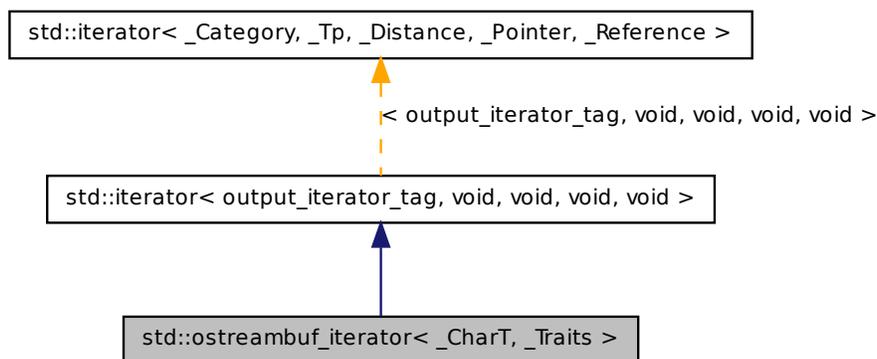
The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.594 `std::ostreambuf_iterator<_CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streambufs.

Inheritance diagram for `std::ostreambuf_iterator<_CharT, _Traits >`:



## Public Types

- typedef void [difference\\_type](#)
  - typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
  - typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- 
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [streambuf\\_type](#)
  - typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

## Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) throw ()
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) throw ()
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const [\\_CharT](#) \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const throw ()
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

## Friends

- template<typename [\\_CharT2](#) >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_char](#)< [\\_CharT2](#) >::\_\_value, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >::\_\_type **copy** ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >)

### 5.594.1 Detailed Description

```
template<typename \_CharT, typename \_Traits> class std::ostreambuf\_iterator< \_CharT, \_Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 204 of file [streambuf\\_iterator.h](#).

## 5.594 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

### 5.594.2 Member Typedef Documentation

**5.594.2.1** `template<typename _CharT , typename _Traits > typedef _CharT  
std::ostreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 211 of file streambuf\_iterator.h.

**5.594.2.2** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 121 of file stl\_iterator\_base\_types.h.

**5.594.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void  
, void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 117 of file stl\_iterator\_base\_types.h.

**5.594.2.4** `template<typename _CharT , typename _Traits > typedef  
basic_ostream<_CharT, _Traits> std::ostreambuf_iterator<  
_CharT, _Traits >::ostream_type`

Public typedefs.

Definition at line 214 of file streambuf\_iterator.h.

**5.594.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 123 of file stl\_iterator\_base\_types.h.

**5.594.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

**5.594.2.7** `template<typename _CharT , typename _Traits > typedef  
basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator<  
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 213 of file streambuf\_iterator.h.

**5.594.2.8** `template<typename _CharT , typename _Traits > typedef _Traits  
std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 212 of file streambuf\_iterator.h.

**5.594.2.9** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file stl\_iterator\_base\_types.h.

### 5.594.3 Constructor & Destructor Documentation

**5.594.3.1** `template<typename _CharT , typename _Traits >  
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (  
ostream_type & __s ) throw () [inline]`

Construct output iterator from ostream.

Definition at line 229 of file streambuf\_iterator.h.

**5.594.3.2** `template<typename _CharT , typename _Traits >  
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (  
streambuf_type * __s ) throw () [inline]`

Construct output iterator from streambuf.

Definition at line 233 of file streambuf\_iterator.h.

## 5.594.4 Member Function Documentation

**5.594.4.1** `template<typename _CharT, typename _Traits > bool  
std::ostreambuf_iterator< _CharT, _Traits >::failed ( ) const throw  
( ) [inline]`

Return true if previous `operator=()` failed.

Definition at line 263 of file `streambuf_iterator.h`.

**5.594.4.2** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator*( ) [inline]`

Return `*this`.

Definition at line 248 of file `streambuf_iterator.h`.

**5.594.4.3** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator++( ) [inline]`

Return `*this`.

Definition at line 258 of file `streambuf_iterator.h`.

**5.594.4.4** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator++( int ) [inline]`

Return `*this`.

Definition at line 253 of file `streambuf_iterator.h`.

**5.594.4.5** `template<typename _CharT, typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator=( _CharT __c ) [inline]`

Write character to `streambuf`. Calls `streambuf.sputc()`.

Definition at line 238 of file `streambuf_iterator.h`.

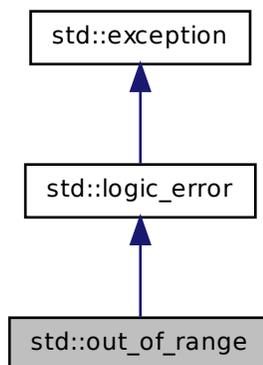
References `std::basic_streambuf<_CharT, _Traits>::sputc()`.

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

## 5.595 std::out\_of\_range Class Reference

Inheritance diagram for std::out\_of\_range:



### Public Member Functions

- `out_of_range` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.595.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in [basic\\_string](#)).

Definition at line 96 of file `stdexcept`.

#### 5.595.2 Member Function Documentation

##### 5.595.2.1 virtual const char\* std::logic\_error::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## **5.596 std::output\_iterator\_tag Struct Reference**

Marking output iterators.

### **5.596.1 Detailed Description**

Marking output iterators.

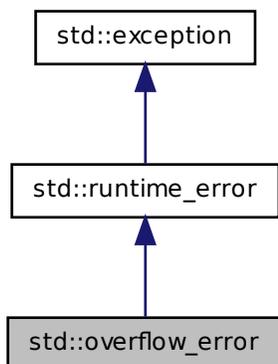
Definition at line 88 of file [stl\\_iterator\\_base\\_types.h](#).

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.597 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- `overflow_error` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.597.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 133 of file `stdexcept`.

#### 5.597.2 Member Function Documentation

##### 5.597.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

## **5.598 std::owner\_less< shared\_ptr< \_Tp > > Struct Template Reference 2811**

The documentation for this class was generated from the following file:

- [stdexcept](#)

## **5.598 std::owner\_less< shared\_ptr< \_Tp > > Struct Template Reference**

Partial specialization of `owner_less` for [shared\\_ptr](#).

Inherits `_Sp_owner_less< shared_ptr< _Tp >, weak_ptr< _Tp > >`.

### **5.598.1 Detailed Description**

```
template<typename _Tp> struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for [shared\\_ptr](#).

Definition at line 384 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## **5.599 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference**

Partial specialization of `owner_less` for [weak\\_ptr](#).

Inherits `_Sp_owner_less< weak_ptr< _Tp >, shared_ptr< _Tp > >`.

### **5.599.1 Detailed Description**

```
template<typename _Tp> struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for [weak\\_ptr](#).

Definition at line 390 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 5.600 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

`packaged_task`

### Public Types

- typedef `_Res` `result_type`

### Public Member Functions

- `template<typename _Fn >`  
`packaged_task` (`const _Fn &__fn`)
- `packaged_task` (`_Res(*__fn)(_ArgTypes...)`)
- `packaged_task` (`packaged_task &&__other`)
- `template<typename _Fn >`  
`packaged_task` (`_Fn &&__fn`)
- `packaged_task` (`packaged_task &`)
- `future<_Res >` `get_future` ()
- `operator bool` () `const`
- `void operator()` (`_ArgTypes...__args`)
- `packaged_task &` `operator=` (`packaged_task &&__other`)
- `packaged_task &` `operator=` (`packaged_task &`)
- `void reset` ()
- `void swap` (`packaged_task &__other`)

### 5.600.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::packaged_task<_Res(_ArgTypes...)>`

`packaged_task`

Definition at line 1183 of file `future`.

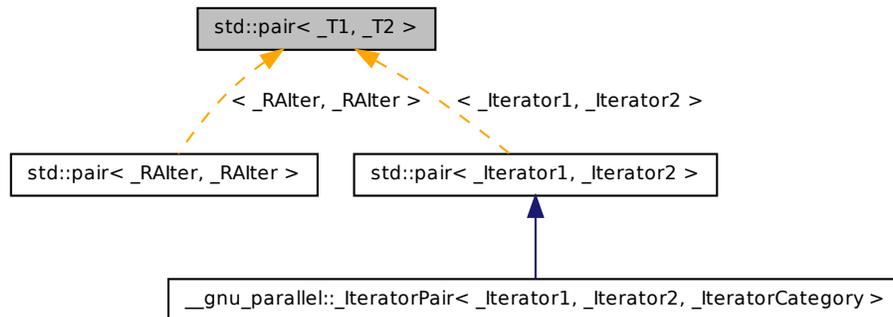
The documentation for this class was generated from the following file:

- `future`

## 5.601 `std::pair< _T1, _T2 >` Struct Template Reference

`pair` holds two objects of arbitrary type.

Inheritance diagram for `std::pair< _T1, _T2 >`:



### Public Types

- typedef `_T1` `first_type`
- typedef `_T2` `second_type`

### Public Member Functions

- `pair` ()
- `pair` (const `_T1` &\_\_a, const `_T2` &\_\_b)
- template<class `_U2` , class = typename `std::enable_if<std::is_convertible<_U2, _T2>::value>::type`>  
`pair` (const `_T1` &\_\_x, `_U2` &&\_\_y)
- template<class `_U1` , class `_U2` , class = typename `std::enable_if<std::is_convertible<_U1, _T1>::value && std::is_convertible<_U2, _T2>::value>::type`>  
`pair` (`_U1` &&\_\_x, `_U2` &&\_\_y)
- template<class `_U1` , class `_U2` >  
`pair` (`pair`< `_U1`, `_U2` > &&\_\_p)
- template<class `_U1` , class `_U2` >  
`pair` (const `pair`< `_U1`, `_U2` > &\_\_p)

- `template<class _U1 , class = typename std::enable_if<std::is_convertible<_U1, _T1>::value>::type>`  
`pair (_U1 &&__x, const _T2 &__y)`
- `pair & operator= (pair &&__p)`
- `template<class _U1 , class _U2 >`  
`pair & operator= (pair< _U1, _U2 > &&__p)`
- `void swap (pair &__p)`

## Public Attributes

- `_T1` *first*
- `_T2` *second*

### 5.601.1 Detailed Description

`template<class _T1, class _T2> struct std::pair< _T1, _T2 >`

`pair` holds two objects of arbitrary type.

Definition at line 71 of file `stl_pair.h`.

### 5.601.2 Member Typedef Documentation

**5.601.2.1** `template<class _T1, class _T2> typedef _T1 std::pair< _T1, _T2 >::first_type`

`first_type` is the first bound type

Definition at line 73 of file `stl_pair.h`.

**5.601.2.2** `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`second_type` is the second bound type

Definition at line 74 of file `stl_pair.h`.

### 5.601.3 Constructor & Destructor Documentation

**5.601.3.1** `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair ( )`  
`[inline]`

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 83 of file `stl_pair.h`.

**5.601.3.2** `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair ( const`  
`_T1 & __a, const _T2 & __b ) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 87 of file `stl_pair.h`.

**5.601.3.3** `template<class _T1, class _T2> template<class _U1, class _U2 >`  
`std::pair< _T1, _T2 >::pair ( const pair< _U1, _U2 > & __p )`  
`[inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 114 of file `stl_pair.h`.

### 5.601.4 Member Data Documentation

**5.601.4.1** `template<class _T1, class _T2> _T1 std::pair< _T1, _T2 >::first`

`first` is a copy of the first object

Definition at line 76 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set< _Key, _Compare, _Alloc >::insert()`, and `std::operator==( )`.

**5.601.4.2** `template<class _T1, class _T2> _T2 std::pair< _T1, _T2 >::second`

`second` is a copy of the second object

Definition at line 77 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set< _Key, _Compare, _Alloc >::insert()`, and `std::operator==( )`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.602 `std::piecewise_constant_distribution`< `_RealType` > Class Template Reference \_-

A [piecewise\\_constant\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**piecewise\_constant\_distribution** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**piecewise\_constant\_distribution** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **piecewise\_constant\_distribution** (`const param_type &__p`)
- `template<typename _Func >`  
**piecewise\_constant\_distribution** (`initializer_list<_RealType > __bl, _Func __fw`)
- `std::vector< double >` [densities](#) () const
- `std::vector< _RealType >` [intervals](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) [operator](#)() (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) [operator](#)() (`_UniformRandomNumberGenerator &__urng`)
- [param\\_type](#) [param](#) () const
- void [param](#) (`const param_type &__param`)
- void [reset](#) ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits > & operator<<< (std::basic_ostream<_CharT, _Traits > &, const std::piecewise_constant_distribution<_RealType1 > &)`
- `template<typename _RealType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits > & operator>>> (std::basic_istream<_CharT, _Traits > &, std::piecewise_constant_distribution<_RealType1 > &)`

### 5.602.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_constant_distribution<_RealType >`

A `piecewise_constant_distribution` random number distribution. The formula for the piecewise constant probability mass function is

Definition at line 4874 of file `random.h`.

### 5.602.2 Member Typedef Documentation

**5.602.2.1** `template<typename _RealType = double> typedef _RealType std::piecewise_constant_distribution<_RealType >::result_type`

The type of the range of the distribution.

Definition at line 4881 of file `random.h`.

### 5.602.3 Member Function Documentation

**5.602.3.1** `template<typename _RealType = double> std::vector<double> std::piecewise_constant_distribution<_RealType >::densities ( ) const [inline]`

Returns a vector of the probability densities.

Definition at line 4973 of file `random.h`.

**5.602.3.2** `template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution<_RealType >::intervals ( ) const [inline]`

Returns a vector of the intervals.

Definition at line 4966 of file random.h.

**5.602.3.3** `template<typename _RealType = double> result_type  
std::piecewise_constant_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5002 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`.

**5.602.3.4** `template<typename _RealType = double> result_type  
std::piecewise_constant_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4995 of file random.h.

References `std::vector<_Tp, _Alloc>::front()`.

**5.602.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::piecewise_constant_distribution<_RealType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5010 of file random.h.

References `std::piecewise_constant_distribution<_RealType>::operator()()`, and `std::piecewise_constant_distribution<_RealType>::param()`.

Referenced by `std::piecewise_constant_distribution<_RealType>::operator()()`.

**5.602.3.6** `template<typename _RealType = double> void  
std::piecewise_constant_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

*\_\_param* The new parameter set of the distribution.

Definition at line 4988 of file random.h.

## 5.602 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

2819

**5.602.3.7** `template<typename _RealType = double> param_type  
std::piecewise_constant_distribution<_RealType>::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4980 of file `random.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::operator()()`, and `std::operator==( )`.

**5.602.3.8** `template<typename _RealType = double> void  
std::piecewise_constant_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 4959 of file `random.h`.

## 5.602.4 Friends And Related Function Documentation

**5.602.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::piecewise_constant_distribution<_RealType1> & )  
[friend]`

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `piecewise_constant_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.602.4.2** `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > &, std::piecewise_constant_distribution<_RealType1 > & ) [friend]`

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `piecewise_constant_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.603 `std::piecewise_constant_distribution<_RealType >::param_type` Struct Reference

### Public Types

- typedef `piecewise_constant_distribution<_RealType > distribution_type`

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW > param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)`
- `template<typename _Func > param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)`
- `template<typename _Func > param_type (initializer_list<_RealType > __bi, _Func __fw)`
- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`

## Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **piecewise\_constant\_distribution**< [\\_RealType](#) >

### 5.603.1 Detailed Description

**template**<typename [\\_RealType](#) = double> **struct** **std::piecewise\_constant\_distribution**< [\\_RealType](#) >::[param\\_type](#)

Parameter type.

Definition at line 4883 of file [random.h](#).

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.604 std::piecewise\_linear\_distribution< \_RealType > Class Template Reference

A [piecewise\\_linear\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef [\\_RealType](#) [result\\_type](#)

### Public Member Functions

- **template**<typename [\\_InputIteratorB](#) , typename [\\_InputIteratorW](#) >  
**piecewise\_linear\_distribution** ([\\_InputIteratorB](#) \_\_bfirst, [\\_InputIteratorB](#) \_\_bend, [\\_InputIteratorW](#) \_\_wbegin)
- **template**<typename [\\_Func](#) >  
**piecewise\_linear\_distribution** (size\_t \_\_nw, [\\_RealType](#) \_\_xmin, [\\_RealType](#) \_\_xmax, [\\_Func](#) \_\_fw)
- **piecewise\_linear\_distribution** (const [param\\_type](#) &\_\_p)

- `template<typename _Func >`  
**piecewise\_linear\_distribution** (`initializer_list`< `_RealType` > `__bl`, `_Func` `__fw`)
- `std::vector`< `double` > `densities` () const
- `std::vector`< `_RealType` > `intervals` () const
- `result_type` `max` () const
- `result_type` `min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type` `operator`() (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type` `operator`() (`_UniformRandomNumberGenerator` &`__urng`)
- `param_type` `param` () const
- void `param` (const `param_type` &`__param`)
- void `reset` ()

## Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream`< `_CharT`, `_Traits` > & `operator<<` (`std::basic_ostream`< `_CharT`, `_Traits` > &, const `std::piecewise_linear_distribution`< `_RealType1` > &)
- `template<typename _RealType1 , typename _CharT , typename _Traits >`  
`std::basic_istream`< `_CharT`, `_Traits` > & `operator>>` (`std::basic_istream`< `_CharT`, `_Traits` > &, `std::piecewise_linear_distribution`< `_RealType1` > &)

### 5.604.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_linear_distribution< _RealType >`

A `piecewise_linear_distribution` random number distribution. The formula for the piecewise linear probability mass function is

Definition at line 5082 of file `random.h`.

### 5.604.2 Member Typedef Documentation

**5.604.2.1** `template<typename _RealType = double> typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 5089 of file `random.h`.

### 5.604.3 Member Function Documentation

**5.604.3.1** `template<typename _RealType = double> std::vector<double>  
std::piecewise_linear_distribution<_RealType>::densities ( ) const  
[inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5184 of file `random.h`.

**5.604.3.2** `template<typename _RealType = double> std::vector<_RealType>  
std::piecewise_linear_distribution<_RealType>::intervals ( ) const  
[inline]`

Return the intervals of the distribution.

Definition at line 5176 of file `random.h`.

**5.604.3.3** `template<typename _RealType = double> result_type  
std::piecewise_linear_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5213 of file `random.h`.

References `std::vector<_Tp, _Alloc>::back()`.

**5.604.3.4** `template<typename _RealType = double> result_type  
std::piecewise_linear_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5206 of file `random.h`.

References `std::vector<_Tp, _Alloc>::front()`.

**5.604.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::piecewise_linear_distribution<_RealType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5221 of file `random.h`.

References `std::piecewise_linear_distribution<_RealType>::operator()`, and `std::piecewise_linear_distribution<_RealType>::param()`.

Referenced by `std::piecewise_linear_distribution<_RealType>::operator()`.

**5.604.3.6** `template<typename _RealType = double> void  
std::piecewise_linear_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 5199 of file `random.h`.

**5.604.3.7** `template<typename _RealType = double> param_type  
std::piecewise_linear_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 5191 of file `random.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::operator()`, and `std::operator==( )`.

**5.604.3.8** `template<typename _RealType = double> void  
std::piecewise_linear_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 5169 of file `random.h`.

## 5.604.4 Friends And Related Function Documentation

**5.604.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits > & , const  
std::piecewise_linear_distribution<_RealType1 > & ) [friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

## 5.605 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference 2825

---

### Parameters

- `__os` An output stream.
- `__x` A `piecewise_linear_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

```
5.604.4.2 template<typename _RealType = double> template<typename
    _RealType1, typename _CharT, typename _Traits >
    std::basic_istream<_CharT, _Traits>& operator>>
    ( std::basic_istream<_CharT, _Traits > & ,
      std::piecewise_linear_distribution<_RealType1 > & ) [friend]
```

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

- `__is` An input stream.
- `__x` A `piecewise_linear_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.605 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `piecewise_linear_distribution<_RealType>` `distribution_type`

### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
`param_type` (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)

- `template<typename _Func >`  
**param\_type** (`size_t __nw`, `_RealType __xmin`, `_RealType __xmax`, `_Func __fw`)
- `template<typename _Func >`  
**param\_type** (`initializer_list<_RealType > __bl`, `_Func __fw`)
- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`

## Friends

- `bool operator== (const param_type &__p1, const param_type &__p2)`
- `class piecewise_linear_distribution< _RealType >`

### 5.605.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_linear_distribution< _RealType >::param_type`

Parameter type.

Definition at line 5091 of file `random.h`.

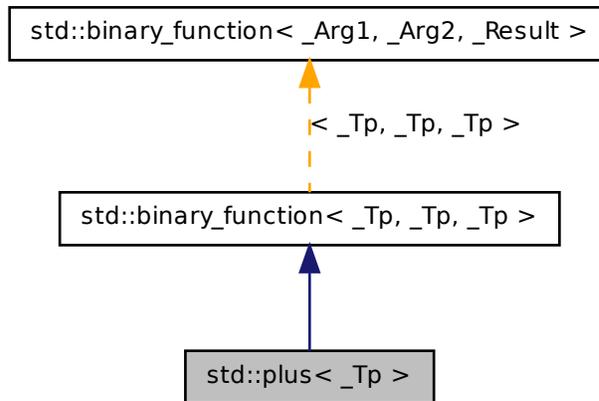
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

### 5.606 `std::plus< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::plus< \_Tp >:



## Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

## Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

### 5.606.1 Detailed Description

```
template<typename _Tp> struct std::plus< _Tp >
```

One of the [math functors](#).

Definition at line 135 of file `stl_function.h`.

## 5.606.2 Member Typedef Documentation

### 5.606.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

### 5.606.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` [inherited]

type of the return type

Definition at line 118 of file `stl_function.h`.

### 5.606.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

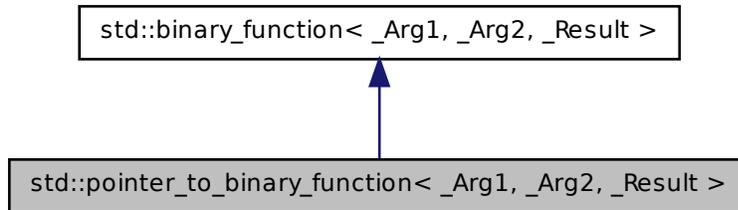
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.607 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- `pointer_to_binary_function` (`_Result`(\*`__x`)(`_Arg1`, `_Arg2`))
- `_Result operator()` (`_Arg1` `__x`, `_Arg2` `__y`) const

## Protected Attributes

- `_Result`(\* `_M_ptr`)(`_Arg1`, `_Arg2`)

### 5.607.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> class  
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 442 of file `stl_function.h`.

## 5.607.2 Member Typedef Documentation

**5.607.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result  
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

**5.607.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Result std::binary_function< _Arg1, _Arg2, _Result  
>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

**5.607.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result  
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

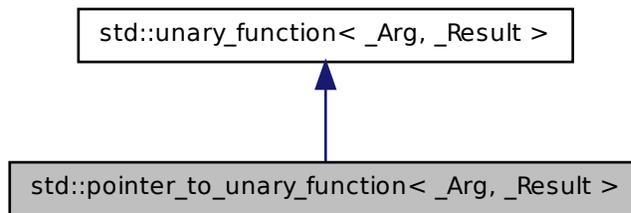
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.608 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_unary_function<_Arg, _Result>`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- `pointer_to_unary_function` (`_Result(*__x)(_Arg)`)
- `_Result operator()` (`_Arg __x`) const

## Protected Attributes

- `_Result(*_M_ptr)(_Arg)`

### 5.608.1 Detailed Description

```
template<typename _Arg, typename _Result> class std::pointer_to_unary_function<_Arg, _Result>
```

One of the [adaptors for function pointers](#).

Definition at line 417 of file `stl_function.h`.

## 5.608.2 Member Typedef Documentation

**5.608.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.608.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.609 `std::poisson_distribution< _IntType >` Class Template Reference

A discrete Poisson random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- `poisson_distribution` (double `__mean=1.0`)
- `poisson_distribution` (const [param\\_type](#) &`__p`)
- [result\\_type](#) `max` () const
- double `mean` () const
- [result\\_type](#) `min` () const

- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

## Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::poisson_distribution< _IntType1 > &)`
- `template<typename _IntType1 >`  
`bool operator== (const std::poisson_distribution< _IntType1 > &__d1, const std::poisson_distribution< _IntType1 > &__d2)`
- `template<typename _IntType1 , typename _CharT , typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::poisson_distribution< _IntType1 > &)`

### 5.609.1 Detailed Description

`template<typename _IntType = int> class std::poisson_distribution< _IntType >`

A discrete Poisson random number distribution. The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 3973 of file random.h.

### 5.609.2 Member Typedef Documentation

**5.609.2.1** `template<typename _IntType = int> typedef _IntType std::poisson_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3980 of file random.h.

### 5.609.3 Member Function Documentation

**5.609.3.1** `template<typename _IntType = int> result_type  
std::poisson_distribution< _IntType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4067 of file random.h.

Referenced by `std::poisson_distribution< _IntType >::operator()`.

**5.609.3.2** `template<typename _IntType = int> double  
std::poisson_distribution< _IntType >::mean ( ) const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4038 of file random.h.

**5.609.3.3** `template<typename _IntType = int> result_type  
std::poisson_distribution< _IntType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4060 of file random.h.

**5.609.3.4** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > poisson_distribution<  
_IntType >::result_type std::poisson_distribution< _IntType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

A rejection algorithm when `mean >= 12` and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1158 of file random.tcc.

References `std::abs()`, `std::log()`, and `std::poisson_distribution< _IntType >::max()`.

**5.609.3.5** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::poisson_distribution<_IntType >::operator() (  
_UniformRandomNumberGenerator & _urng ) [inline]`

Generating functions.

Definition at line 4075 of file `random.h`.

References `std::poisson_distribution<_IntType >::operator()`, and `std::poisson_distribution<_IntType >::param()`.

Referenced by `std::poisson_distribution<_IntType >::operator()`.

**5.609.3.6** `template<typename _IntType = int> param_type  
std::poisson_distribution<_IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4045 of file `random.h`.

Referenced by `std::poisson_distribution<_IntType >::operator()`.

**5.609.3.7** `template<typename _IntType = int> void std::poisson_distribution<  
_IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

*\_\_param* The new parameter set of the distribution.

Definition at line 4053 of file `random.h`.

**5.609.3.8** `template<typename _IntType = int> void std::poisson_distribution<  
_IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4031 of file `random.h`.

References `std::normal_distribution<_RealType >::reset()`.

## 5.609.4 Friends And Related Function Documentation

**5.609.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > & ,  
const std::poisson_distribution<_IntType1 > & ) [friend]`

Inserts a `poisson_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `poisson_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.609.4.2** `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::poisson_distribution<_IntType1 >  
& __d1, const std::poisson_distribution<_IntType1 > & __d2 )  
[friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4090 of file `random.h`.

**5.609.4.3** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & ,  
std::poisson_distribution<_IntType1 > & ) [friend]`

Extracts a `poisson_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A `poisson_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

## 5.610 std::poisson\_distribution< \_IntType >::param\_type Struct Reference 2837

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## **5.610 std::poisson\_distribution< \_IntType >::param\_type Struct Reference**

### **Public Types**

- typedef [poisson\\_distribution< \\_IntType >](#) **distribution\_type**

### **Public Member Functions**

- **param\_type** (double \_\_mean=1.0)
- double **mean** () const

### **Friends**

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class [poisson\\_distribution< \\_IntType >](#)

### **5.610.1 Detailed Description**

**template<typename \_IntType = int> struct std::poisson\_distribution< \_IntType >::param\_type**

Parameter type.

Definition at line 3982 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## **5.611 std::priority\_queue< \_Tp, \_Sequence, \_Compare > Class Template Reference**

A standard container automatically sorting its contents.

## Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

## Public Member Functions

- [priority\\_queue](#) (`const _Compare &__x, const _Sequence &__s`)
- **priority\_queue** (`const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence()`)
- `template<typename _InputIterator >`  
**priority\_queue** (`_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence()`)
- **priority\_queue** ([priority\\_queue](#) &&\_\_pq)
- `template<typename _InputIterator >`  
[priority\\_queue](#) (`_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s`)
- `template<typename... _Args>`  
void **emplace** (`_Args &&... __args`)
- bool **empty** () const
- [priority\\_queue](#) & **operator=** ([priority\\_queue](#) &&\_\_pq)
- void **pop** ()
- void **push** (`const value_type &__x`)
- void **push** (`value_type &&__x`)
- `size_type` **size** () const
- void **swap** ([priority\\_queue](#) &\_\_pq)
- `const_reference` **top** () const

## Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

### 5.611.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> class std::priority_queue<_Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents. This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that con-

## 5.611 `std::priority_queue<_Tp, _Sequence, _Compare>` Class Template Reference

2839

tainer. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

### Note

No equality/comparison operators are provided for `priority_queue`. Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 359 of file `stl_queue.h`.

## 5.611.2 Constructor & Destructor Documentation

**5.611.2.1** `template<typename _Tp, typename _Sequence = vector<_Tp>,  
typename _Compare = less<typename _Sequence::value_type>>  
std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue(  
const _Compare & __x, const _Sequence & __s ) [inline,  
explicit]`

Default constructor creates no elements.

Definition at line 394 of file `stl_queue.h`.

References `std::make_heap()`.

```

5.611.2.2 template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator > std::priority_queue<_Tp,
_Sequence, _Compare >::priority_queue ( _InputIterator __first,
_InputIterator __last, const _Compare & __x, const _Sequence &
__s ) [inline]

```

Builds a queue from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*x* A comparison functor describing a strict weak ordering.

*s* An initial sequence with which to start.

Begins by copying *s*, inserting a copy of the elements from [first,last) into the copy of *s*, then ordering the copy according to *x*.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 434 of file `stl_queue.h`.

References `std::make_heap()`.

### 5.611.3 Member Function Documentation

```

5.611.3.1 template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
bool std::priority_queue<_Tp, _Sequence, _Compare >::empty ( )
const [inline]

```

Returns true if the queue is empty.

Definition at line 471 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```

5.611.3.2 template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
void std::priority_queue<_Tp, _Sequence, _Compare >::pop ( )
[inline]

```

Removes first element.

## 5.611 `std::priority_queue<_Tp, _Sequence, _Compare >` Class Template Reference 2841

---

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 534 of file `stl_queue.h`.

References `std::pop_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```
5.611.3.3 template<typename _Tp, typename _Sequence = vector<_Tp>,
                typename _Compare = less<typename _Sequence::value_type>>
void std::priority_queue<_Tp, _Sequence, _Compare >::push (
    const value_type & __x ) [inline]
```

Add data to the queue.

### Parameters

`x` Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 499 of file `stl_queue.h`.

References `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```
5.611.3.4 template<typename _Tp, typename _Sequence = vector<_Tp>,
                typename _Compare = less<typename _Sequence::value_type>>
size_type std::priority_queue<_Tp, _Sequence, _Compare >::size (
    ) const [inline]
```

Returns the number of elements in the queue.

Definition at line 476 of file `stl_queue.h`.

```
5.611.3.5 template<typename _Tp, typename _Sequence = vector<_Tp>,
                typename _Compare = less<typename _Sequence::value_type>>
const_reference std::priority_queue<_Tp, _Sequence, _Compare
>::top ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 484 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.612 `std::promise<_Res>` Class Template Reference

Primary template for `promise`.

### Public Member Functions

- `promise` (`promise` &&\_\_rhs)
- `promise` (const `promise` &)
- `future<_Res>` `get_future` ()
- `promise` & `operator=` (`promise` &&\_\_rhs)
- `promise` & `operator=` (const `promise` &)
- void `set_exception` (exception\_ptr \_\_p)
- void `set_value` (const \_Res &\_\_r)
- void `set_value` (\_Res &&\_\_r)
- void `swap` (`promise` &\_\_rhs)

### Friends

- class `_State::Setter`

#### 5.612.1 Detailed Description

`template<typename _Res> class std::promise<_Res>`

Primary template for `promise`.

Definition at line 825 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.613 `std::promise<_Res &>` Class Template Reference

Partial specialization for `promise<R&>`

### Public Member Functions

- `promise` (`promise` &&\_\_rhs)
- `promise` (const `promise` &)
- `future<_Res &>` `get_future` ()
- `promise` & `operator=` (`promise` &&\_\_rhs)
- `promise` & `operator=` (const `promise` &)
- void `set_exception` (exception\_ptr \_\_p)
- void `set_value` (\_Res &\_\_r)
- void `swap` (`promise` &\_\_rhs)

### Friends

- class `_State::_Setter`

#### 5.613.1 Detailed Description

`template<typename _Res> class std::promise<_Res &>`

Partial specialization for `promise<R&>`

Definition at line 915 of file `future`.

The documentation for this class was generated from the following file:

- `future`

## 5.614 `std::promise<void>` Class Template Reference

Explicit specialization for `promise<void>`

### Public Member Functions

- `promise` (`promise` &&\_\_rhs)
- `promise` (const `promise` &)

- [future](#)< void > **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs)
- [promise](#) & **operator=** (const [promise](#) &)
- void **set\_exception** (exception\_ptr \_\_p)
- void **set\_value** ()
- void **swap** ([promise](#) &\_\_rhs)

## Friends

- class `_State::_Setter`

### 5.614.1 Detailed Description

`template<> class std::promise< void >`

Explicit specialization for [promise<void>](#)

Definition at line 993 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.615 `std::queue< _Tp, _Sequence >` Class Template Reference

A standard container giving FIFO behavior.

### Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

### Public Member Functions

- [queue](#) (const `_Sequence` &\_\_c)
- [queue](#) (`_Sequence` &&\_\_c=\_Sequence())

- `queue` (`queue` &&\_\_q)
- reference `back` ()
- const\_reference `back` () const
- `template<typename... _Args>`  
void `emplace` (\_Args &&...\_\_args)
- bool `empty` () const
- const\_reference `front` () const
- reference `front` ()
- `queue` & `operator=` (`queue` &&\_\_q)
- void `pop` ()
- void `push` (value\_type &&\_\_x)
- void `push` (const value\_type &\_\_x)
- size\_type `size` () const
- void `swap` (`queue` &\_\_q)

### Protected Attributes

- `_Sequence` `c`

### Friends

- `template<typename _Tp1, typename _Seq1 >`  
bool `operator` (const `queue`<\_Tp1, \_Seq1 > &, const `queue`<\_Tp1, \_Seq1 > &)
- `template<typename _Tp1, typename _Seq1 >`  
bool `operator==` (const `queue`<\_Tp1, \_Seq1 > &, const `queue`<\_Tp1, \_Seq1 > &)

### 5.615.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>> class std::queue<_Tp, _Sequence >`

A standard container giving FIFO behavior. Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/-FIFO operations.

Definition at line 89 of file `stl_queue.h`.

## 5.615.2 Constructor & Destructor Documentation

**5.615.2.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
std::queue<_Tp, _Sequence>::queue ( const _Sequence & __c )  
[inline, explicit]`

Default constructor creates no elements.

Definition at line 134 of file `stl_queue.h`.

## 5.615.3 Member Function Documentation

**5.615.3.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
reference std::queue<_Tp, _Sequence>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 191 of file `stl_queue.h`.

**5.615.3.2** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
const_reference std::queue<_Tp, _Sequence>::back ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 202 of file `stl_queue.h`.

**5.615.3.3** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
bool std::queue<_Tp, _Sequence>::empty ( ) const [inline]`

Returns true if the queue is empty.

Definition at line 156 of file `stl_queue.h`.

**5.615.3.4** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
reference std::queue<_Tp, _Sequence>::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 169 of file `stl_queue.h`.

**5.615.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`const_reference std::queue<_Tp, _Sequence >::front ( ) const`  
`[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 180 of file `stl_queue.h`.

**5.615.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`void std::queue<_Tp, _Sequence >::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 244 of file `stl_queue.h`.

**5.615.3.7** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`void std::queue<_Tp, _Sequence >::push ( const value_type & __x`  
`) [inline]`

Add data to the end of the queue.

#### Parameters

`x` Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 218 of file `stl_queue.h`.

**5.615.3.8** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`size_type std::queue<_Tp, _Sequence >::size ( ) const [inline]`

Returns the number of elements in the queue.

Definition at line 161 of file `stl_queue.h`.

## 5.615.4 Member Data Documentation

### 5.615.4.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> _Sequence std::queue<_Tp, _Sequence>::c` [protected]

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 122 of file `stl_queue.h`.

Referenced by `std::operator==(())`.

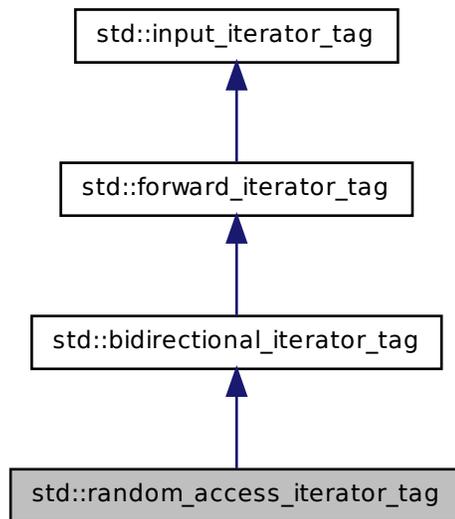
The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.616 `std::random_access_iterator_tag` Struct Reference

Random-access iterators support a superset of bidirectional `///` iterator operations.

Inheritance diagram for `std::random_access_iterator_tag`:



### 5.616.1 Detailed Description

Random-access iterators support a superset of bidirectional `///` iterator operations.

Definition at line 99 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.617 `std::random_device` Class Reference

### Public Types

- typedef unsigned int `result_type`

## Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token="mt19937")
- **random\_device** (const [random\\_device](#) &)
- double **entropy** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- [result\\_type](#) **operator**() ()
- void **operator=** (const [random\\_device](#) &)

### 5.617.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1521 of file random.h.

### 5.617.2 Member Typedef Documentation

#### 5.617.2.1 `typedef unsigned int std::random_device::result_type`

The type of the generated random value.

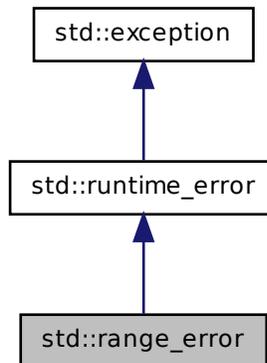
Definition at line 1525 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.618 `std::range_error` Class Reference

Inheritance diagram for `std::range_error`:



### Public Member Functions

- `range_error` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.618.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 126 of file `stdexcept`.

#### 5.618.2 Member Function Documentation

##### 5.618.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.619 `std::ratio< _Num, _Den >` Struct Template Reference

Provides compile-time rational arithmetic.

### Public Member Functions

- **static\_assert** (`_Den!=0`, "denominator cannot be zero")
- **static\_assert** (`_Num >= __INTMAX_MAX__ && _Den >= __INTMAX_MAX__`, "out of range")

### Static Public Attributes

- static const `intmax_t` **den**
- static const `intmax_t` **num**

#### 5.619.1 Detailed Description

`template<intmax_t _Num, intmax_t _Den = 1> struct std::ratio< _Num, _Den >`

Provides compile-time rational arithmetic. This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

Definition at line 151 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.620 `std::ratio_add< _R1, _R2 >` Struct Template Reference

[ratio\\_add](#)

### Public Types

- typedef `ratio< __safe_add< __safe_multiply< _R1::num,(_R2::den/___gcd)>::value, __safe_multiply< _R2::num,(_R1::den/___gcd)>::value >::value, __safe_multiply< _R1::den,(_R2::den/___gcd)>::value > type`

### 5.620.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_add< _R1, _R2 >
```

[ratio\\_add](#)

Definition at line 173 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.621 `std::ratio_divide< _R1, _R2 >` Struct Template Reference

[ratio\\_divide](#)

### Public Types

- typedef `ratio_multiply< _R1, ratio< _R2::den, _R2::num > >::type type`

### Public Member Functions

- `static_assert` (`_R2::num!=0,"division by 0")`

### 5.621.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_divide< _R1, _R2 >`

[ratio\\_divide](#)

Definition at line 216 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.622 `std::ratio_equal< _R1, _R2 >` Struct Template Reference

[ratio\\_equal](#)

Inherits `integral_constant< bool, _R1::num==_R2::num &&_R1::den==_R2::den >`.

### 5.622.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_equal< _R1, _R2 >`

[ratio\\_equal](#)

Definition at line 227 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.623 `std::ratio_greater< _R1, _R2 >` Struct Template Reference

[ratio\\_greater](#)

Inherits `integral_constant< bool, ratio_less< _R2, _R1 >::value >`.

### 5.623.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_greater< _R1, _R2 >`

[ratio\\_greater](#)

## **5.624 std::ratio\_greater\_equal< \_R1, \_R2 > Struct Template Reference** 2855

Definition at line 269 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## **5.624 std::ratio\_greater\_equal< \_R1, \_R2 > Struct Template Reference**

[ratio\\_greater\\_equal](#)

Inherits `integral_constant< bool, !ratio_less< _R1, _R2 >::value >`.

### **5.624.1 Detailed Description**

```
template<typename _R1, typename _R2> struct std::ratio_greater_equal< _R1,
_R2 >
```

[ratio\\_greater\\_equal](#)

Definition at line 275 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## **5.625 std::ratio\_less< \_R1, \_R2 > Struct Template Reference**

[ratio\\_less](#)

Inherits `__ratio_less_impl::type< _R1, _R2 >`.

### **5.625.1 Detailed Description**

```
template<typename _R1, typename _R2> struct std::ratio_less< _R1, _R2 >
```

[ratio\\_less](#)

Definition at line 257 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.626 `std::ratio_less_equal< _R1, _R2 >` Struct Template Reference

[ratio\\_less\\_equal](#)

Inherits `integral_constant< bool, !ratio_less< _R2, _R1 >::value >`.

### 5.626.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_less_equal< _R1, _R2
>
```

[ratio\\_less\\_equal](#)

Definition at line 263 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.627 `std::ratio_multiply< _R1, _R2 >` Struct Template Reference

[ratio\\_multiply](#)

### Public Types

- typedef `ratio< __safe_multiply<(_R1::num/__gcd1),(_R2::num/_-__gcd2)>::value, __safe_multiply<(_R1::den/__gcd2),(_R2::den/_-gcd1)>::value >` **type**

### 5.627.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_multiply< _R1, _R2
>
```

[ratio\\_multiply](#)

Definition at line 198 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.628 `std::ratio_not_equal< _R1, _R2 >` Struct Template Reference

[ratio\\_not\\_equal](#)

Inherits `integral_constant< bool,!ratio_equal< _R1, _R2 >::value >`.

### 5.628.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2 >
```

[ratio\\_not\\_equal](#)

Definition at line 233 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.629 `std::ratio_subtract< _R1, _R2 >` Struct Template Reference

[ratio\\_subtract](#)

### Public Types

- typedef [ratio\\_add< \\_R1, ratio<-\\_R2::num, \\_R2::den > >::type](#) `type`

### 5.629.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_subtract< _R1, _R2 >
```

[ratio\\_subtract](#)

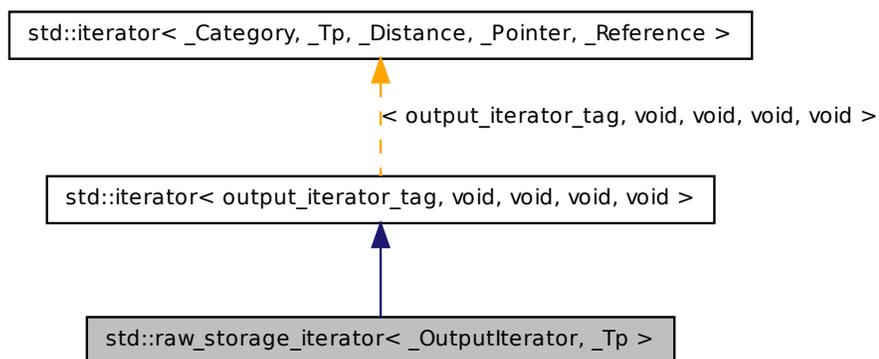
Definition at line 189 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.630 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **`raw_storage_iterator`** (`_OutputIterator __x`)
- **`raw_storage_iterator`** & **`operator*`** ()
- **`raw_storage_iterator`**< `_OutputIterator, _Tp` > & **`operator++`** ()
- **`raw_storage_iterator`**< `_OutputIterator, _Tp` > **`operator++`** (int)
- **`raw_storage_iterator`** & **`operator=`** (const `_Tp` &\_\_element)

## Protected Attributes

- `_OutputIterator_M_iter`

### 5.630.1 Detailed Description

`template<class _OutputIterator, class _Tp> class std::raw_storage_iterator<_OutputIterator, _Tp>`

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 67 of file `stl_raw_storage_iter.h`.

### 5.630.2 Member Typedef Documentation

**5.630.2.1** `typedef void std::iterator<output_iterator_tag, void, void, void, void>::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 121 of file `stl_iterator_base_types.h`.

**5.630.2.2** `typedef output_iterator_tag std::iterator<output_iterator_tag, void, void, void, void>::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

**5.630.2.3** `typedef void std::iterator<output_iterator_tag, void, void, void, void>::pointer [inherited]`

This type represents a `pointer-to-value_type`.

Definition at line 123 of file `stl_iterator_base_types.h`.

**5.630.2.4** `typedef void std::iterator<output_iterator_tag, void, void, void, void>::reference [inherited]`

This type represents a `reference-to-value_type`.

Definition at line 125 of file `stl_iterator_base_types.h`.

### 5.630.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 5.631 `std::recursive_mutex` Class Reference

[recursive\\_mutex](#)

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `recursive_mutex` (const [recursive\\_mutex](#) &)
- `void lock ()`
- `native_handle_type native_handle ()`
- `recursive_mutex & operator=` (const [recursive\\_mutex](#) &)
- `bool try_lock ()`
- `void unlock ()`

### 5.631.1 Detailed Description

[recursive\\_mutex](#)

Definition at line 115 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.632 `std::recursive_timed_mutex` Class Reference

[recursive\\_timed\\_mutex](#)

---

## Public Types

- `typedef __native_type * native_handle_type`

## Public Member Functions

- `recursive_timed_mutex` (const [recursive\\_timed\\_mutex](#) &)
- `void lock ()`
- `native_handle_type native_handle ()`
- `recursive_timed_mutex & operator=` (const [recursive\\_timed\\_mutex](#) &)
- `bool try_lock ()`
- `template<class _Rep, class _Period >`  
`bool try_lock_for` (const [chrono::duration](#)<\_Rep, \_Period > &\_\_rtime)
- `template<class _Clock, class _Duration >`  
`bool try_lock_until` (const [chrono::time\\_point](#)<\_Clock, \_Duration > &\_\_-  
atime)
- `void unlock ()`

### 5.632.1 Detailed Description

[recursive\\_timed\\_mutex](#)

Definition at line 271 of file `mutex`.

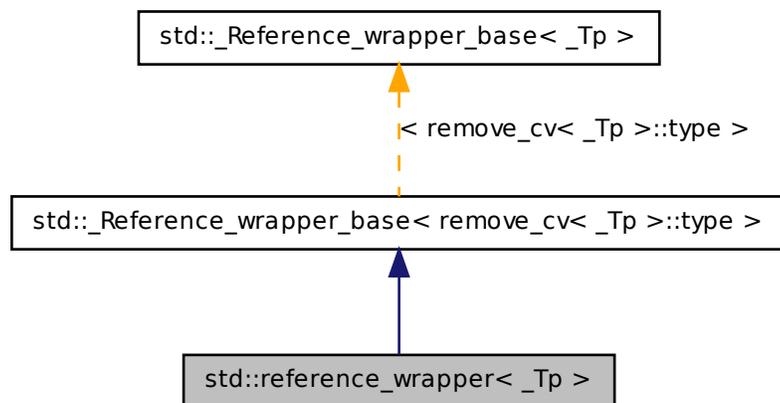
The documentation for this class was generated from the following file:

- [mutex](#)

## 5.633 `std::reference_wrapper<_Tp>` Class Template Reference

Primary class template for [reference\\_wrapper](#).

Inheritance diagram for `std::reference_wrapper<_Tp>`:



## Public Types

- `typedef _Tp type`

## Public Member Functions

- `reference_wrapper` (`_Tp &__indata`)
- `reference_wrapper` (`_Tp &&`)
- `reference_wrapper` (`const reference_wrapper<_Tp> &__inref`)
- `_Tp & get` () const
- `operator _Tp &` () const
- `template<typename... _Args> result_of<_M_func_type(_Args...)>::type operator` () (`_Args &&..._args`) const
- `reference_wrapper & operator=` (`const reference_wrapper<_Tp> &__inref`)

### 5.633.1 Detailed Description

`template<typename _Tp> class std::reference_wrapper< _Tp >`

Primary class template for [reference\\_wrapper](#).

Definition at line 392 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.634 `std::remove_reference< _Tp >` Struct Template Reference

[remove\\_reference](#)

### Public Types

- `typedef _Tp type`

### 5.634.1 Detailed Description

`template<typename _Tp> struct std::remove_reference< _Tp >`

[remove\\_reference](#)

Definition at line 98 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.635 `std::reverse_iterator< _Iterator >` Class Template Reference

Inheritance diagram for `std::reverse_iterator< _Iterator >`:



## Public Types

- typedef `__traits_type::difference_type` [difference\\_type](#)
- typedef `iterator_traits<_Iterator>::iterator_category` [iterator\\_category](#)
- typedef `_Iterator` **iterator\_type**
- typedef `__traits_type::pointer` [pointer](#)
- typedef `__traits_type::reference` [reference](#)
- typedef `iterator_traits<_Iterator>::value_type` [value\\_type](#)

## Public Member Functions

- [reverse\\_iterator](#) ()
- [reverse\\_iterator](#) (iterator\_type \_\_x)
- template<typename \_Iter >  
  [reverse\\_iterator](#) (const [reverse\\_iterator](#)<\_Iter > &\_\_x)
- [reverse\\_iterator](#) (const [reverse\\_iterator](#) &\_\_x)
- iterator\_type [base](#) () const
- [reference](#) [operator\\*](#) () const
- [reverse\\_iterator](#) [operator+](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) & [operator++](#) ()
- [reverse\\_iterator](#) [operator++](#) (int)
- [reverse\\_iterator](#) & [operator+=](#) (difference\_type \_\_n)
- [reverse\\_iterator](#) [operator-](#) (difference\_type \_\_n) const
- [reverse\\_iterator](#) & [operator--](#) ()
- [reverse\\_iterator](#) [operator--](#) (int)
- [reverse\\_iterator](#) & [operator-=](#) (difference\_type \_\_n)
- [pointer](#) [operator->](#) () const
- [reference](#) [operator\[\]](#) (difference\_type \_\_n) const

## Protected Types

- typedef `iterator_traits<_Iterator> __traits_type`

## Protected Attributes

- `_Iterator` **current**

### 5.635.1 Detailed Description

**template<typename `_Iterator`> class `std::reverse_iterator<_Iterator>` >**

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
*(reverse_iterator(i)) == *(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array.* [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 95 of file `stl_iterator.h`.

### 5.635.2 Member Typedef Documentation

**5.635.2.1 `template<typename _Iterator> typedef __traits_difference_type std::reverse_iterator<_Iterator>::difference_type`**

Distance between iterators is represented as this type.

Reimplemented from [`std::iterator< iterator\_traits<\_Iterator>::iterator\_category, iterator\_traits<\_Iterator>::value\_type, iterator\_traits<\_Iterator>::difference\_type, iterator\_traits<\_Iterator>::pointer, iterator\_traits<\_Iterator>::reference >`](#).

Definition at line 109 of file `stl_iterator.h`.

**5.635.2.2 `typedef iterator_traits<_Iterator>::iterator_category std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference >::iterator_category [inherited]`**

One of the [tag types](#).

Definition at line 117 of file `stl_iterator_base_types.h`.

### 5.635.2.3 `template<typename _Iterator> typedef __traits_type::pointer std::reverse_iterator< _Iterator >::pointer`

This type represents a pointer-to-value\_type.

Reimplemented from `std::iterator< iterator_traits< _Iterator >::iterator_category,  
iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type,  
iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >`.

Definition at line 110 of file `stl_iterator.h`.

### 5.635.2.4 `template<typename _Iterator> typedef __traits_type::reference std::reverse_iterator< _Iterator >::reference`

This type represents a reference-to-value\_type.

Reimplemented from `std::iterator< iterator_traits< _Iterator >::iterator_category,  
iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type,  
iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >`.

Definition at line 111 of file `stl_iterator.h`.

### 5.635.2.5 `typedef iterator_traits< _Iterator >::value_type std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 119 of file `stl_iterator_base_types.h`.

## 5.635.3 Constructor & Destructor Documentation

### 5.635.3.1 `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( ) [inline]`

The default constructor default-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 119 of file `stl_iterator.h`.

**5.635.3.2** `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( iterator_type __x ) [inline, explicit]`

This iterator will move in the opposite direction that `x` does.

Definition at line 125 of file `stl_iterator.h`.

**5.635.3.3** `template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( const reverse_iterator< _Iterator > & __x ) [inline]`

The copy constructor is normal.

Definition at line 130 of file `stl_iterator.h`.

**5.635.3.4** `template<typename _Iterator> template<typename _Iter > std::reverse_iterator< _Iterator >::reverse_iterator ( const reverse_iterator< _Iter > & __x ) [inline]`

A [reverse\\_iterator](#) across other types can be copied in the normal fashion.

Definition at line 138 of file `stl_iterator.h`.

## 5.635.4 Member Function Documentation

**5.635.4.1** `template<typename _Iterator> iterator_type std::reverse_iterator< _Iterator >::base ( ) const [inline]`

### Returns

`current`, the iterator used for underlying work.

Definition at line 145 of file `stl_iterator.h`.

Referenced by `std::operator==( )`.

**5.635.4.2** `template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator* ( ) const [inline]`

### Returns

TODO

### Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 154 of file stl\_iterator.h.

**5.635.4.3** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator<_Iterator >::operator+( difference_type __n  
) const [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 225 of file stl\_iterator.h.

**5.635.4.4** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator<_Iterator >::operator++( ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 175 of file stl\_iterator.h.

**5.635.4.5** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator<_Iterator >::operator++( int ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 187 of file stl\_iterator.h.

**5.635.4.6** `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator+=( difference_type __n ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 234 of file stl\_iterator.h.

**5.635.4.7** `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator-( difference_type __n ) const [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 246 of file stl\_iterator.h.

**5.635.4.8** `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator--( int ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 212 of file stl\_iterator.h.

**5.635.4.9** `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-- ( ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 200 of file stl\_iterator.h.

**5.635.4.10** `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-= ( difference_type __n ) [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 255 of file stl\_iterator.h.

**5.635.4.11** `template<typename _Iterator> pointer std::reverse_iterator< _Iterator >::operator-> ( ) const [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 166 of file stl\_iterator.h.

**5.635.4.12** `template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator[] ( difference_type __n ) const [inline]`

**Returns**

TODO

**Todo**

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 267 of file `stl_iterator.h`.

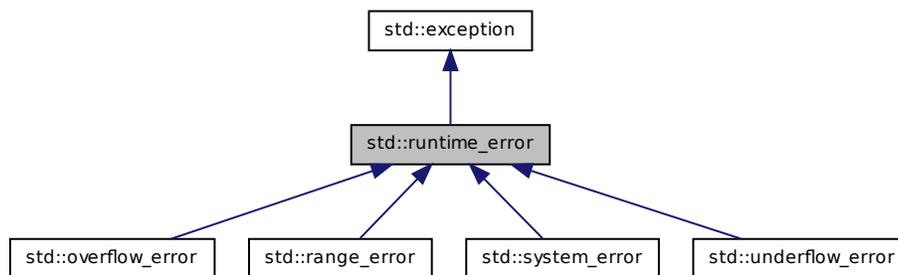
The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.636 `std::runtime_error` Class Reference

One of two subclasses of exception.

Inheritance diagram for `std::runtime_error`:



### Public Member Functions

- `runtime_error` (const `string` &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.636.1 Detailed Description

One of two subclasses of exception. Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 107 of file `stdexcept`.

## 5.636.2 Constructor & Destructor Documentation

### 5.636.2.1 `std::runtime_error::runtime_error ( const string & __arg )` [explicit]

Takes a character string describing the error.

## 5.636.3 Member Function Documentation

### 5.636.3.1 `virtual const char* std::runtime_error::what ( ) const throw ( )` [virtual]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.637 `std::seed_seq` Class Reference

The [seed\\_seq](#) class generates sequences of seeds for random number generators.

### Public Types

- typedef uint\_least32\_t [result\\_type](#)

### Public Member Functions

- [seed\\_seq](#) ()
- template<typename \_IntType >  
[seed\\_seq](#) ([std::initializer\\_list](#)< \_IntType > il)
- template<typename \_InputIterator >  
[seed\\_seq](#) (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- template<typename \_RandomAccessIterator >  
void [generate](#) (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_-  
end)
- template<typename OutputIterator >  
void [param](#) (OutputIterator \_\_dest) const
- [size\\_t size](#) () const

### 5.637.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 5300 of file `random.h`.

### 5.637.2 Member Typedef Documentation

#### 5.637.2.1 `typedef uint_least32_t std::seed_seq::result_type`

The type of the seed vales.

Definition at line 5305 of file `random.h`.

### 5.637.3 Constructor & Destructor Documentation

#### 5.637.3.1 `std::seed_seq::seed_seq ( ) [inline]`

Default constructor.

Definition at line 5308 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.638 `std::set< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of unique keys, which can be retrieved in logarithmic time.

### Public Types

- `typedef _Key key_type`
- `typedef _Key value_type`
- `typedef _Compare key_compare`
- `typedef _Compare value_compare`
- `typedef _Alloc allocator_type`
  
- `typedef _Key_alloc_type::pointer pointer`
- `typedef _Key_alloc_type::const_pointer const_pointer`

- typedef `_Key_alloc_type::reference` [reference](#)
- typedef `_Key_alloc_type::const_reference` [const\\_reference](#)
- typedef `_Rep_type::const_iterator` [iterator](#)
- typedef `_Rep_type::const_iterator` [const\\_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [reverse\\_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [const\\_reverse\\_iterator](#)
- typedef `_Rep_type::size_type` [size\\_type](#)
- typedef `_Rep_type::difference_type` [difference\\_type](#)

## Public Member Functions

- [set](#) ()
- [set](#) (const `_Compare` &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- `template<typename _InputIterator >`  
[set](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [set](#) &\_\_x)
- `template<typename _InputIterator >`  
[set](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- [set](#) ([set](#) &&\_\_x)
- [set](#) (`initializer_list`< [value\\_type](#) > \_\_l, const `_Compare` &\_\_comp=`_Compare`(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [iterator begin](#) () const
- [iterator cbegin](#) () const
- [iterator cend](#) () const
- `void clear` ()
- [size\\_type count](#) (const [key\\_type](#) &\_\_x) const
- [reverse\\_iterator crbegin](#) () const
- [reverse\\_iterator crend](#) () const
- `bool empty` () const
- [iterator end](#) () const
- [iterator erase](#) (`iterator` \_\_first, `iterator` \_\_last)
- [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
- [iterator erase](#) (`iterator` \_\_position)
- [allocator\\_type get\\_allocator](#) () const
- `std::pair`< `iterator`, `bool` > [insert](#) (const [value\\_type](#) &\_\_x)
- [iterator insert](#) (`iterator` \_\_position, const [value\\_type](#) &\_\_x)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void insert` (`initializer_list`< [value\\_type](#) > \_\_l)
- [key\\_compare key\\_comp](#) () const
- [size\\_type max\\_size](#) () const
- [set](#) & [operator=](#) (const [set](#) &\_\_x)
- [set](#) & [operator=](#) ([set](#) &&\_\_x)

- `set & operator= (initializer_list< value_type > __l)`
  - `reverse_iterator rbegin () const`
  - `reverse_iterator rend () const`
  - `size_type size () const`
  - `void swap (set & __x)`
  - `value_compare value_comp () const`
- 
- `iterator find (const key_type & __x)`
  - `const_iterator find (const key_type & __x) const`
- 
- `iterator lower_bound (const key_type & __x)`
  - `const_iterator lower_bound (const key_type & __x) const`
- 
- `iterator upper_bound (const key_type & __x)`
  - `const_iterator upper_bound (const key_type & __x) const`
- 
- `std::pair< iterator, iterator > equal_range (const key_type & __x)`
  - `std::pair< const_iterator, const_iterator > equal_range (const key_type & __x) const`

## Friends

- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

### 5.638.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set< _Key, _Compare, _Alloc >`

A standard container made up of unique keys, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

#### Parameters

*Key* Type of key objects.

*Compare* Comparison function object type, defaults to `less<Key>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 87 of file `stl_set.h`.

## 5.638.2 Member Typedef Documentation

**5.638.2.1** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<_Key, _Compare, _Alloc >::allocator_type`

Public typedefs.

Definition at line 104 of file `stl_set.h`.

**5.638.2.2** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 125 of file `stl_set.h`.

**5.638.2.3** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_pointer std::set<_Key, _Compare, _Alloc >::const_pointer`

Iterator-related typedefs.

Definition at line 118 of file `stl_set.h`.

**5.638.2.4** `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_reference std::set<_Key, _Compare, _Alloc >::const_reference`

Iterator-related typedefs.

Definition at line 120 of file `stl_set.h`.

**5.638.2.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc  
>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 127 of file stl\_set.h.

**5.638.2.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::difference_type std::set< _Key, _Compare, _Alloc  
>::difference_type`

Iterator-related typedefs.

Definition at line 129 of file stl\_set.h.

**5.638.2.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc  
>::iterator`

Iterator-related typedefs.

Definition at line 124 of file stl\_set.h.

**5.638.2.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Compare  
std::set< _Key, _Compare, _Alloc >::key_compare`

Public typedefs.

Definition at line 102 of file stl\_set.h.

**5.638.2.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<  
_Key, _Compare, _Alloc >::key_type`

Public typedefs.

Definition at line 100 of file stl\_set.h.

**5.638.2.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::pointer std::set< _Key, _Compare, _Alloc  
>::pointer`

Iterator-related typedefs.

Definition at line 117 of file `stl_set.h`.

**5.638.2.11** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::reference std::set< _Key, _Compare, _Alloc  
>::reference`

Iterator-related typedefs.

Definition at line 119 of file `stl_set.h`.

**5.638.2.12** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set< _Key, _Compare,  
_Alloc >::reverse_iterator`

Iterator-related typedefs.

Definition at line 126 of file `stl_set.h`.

**5.638.2.13** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::size_type std::set< _Key, _Compare, _Alloc  
>::size_type`

Iterator-related typedefs.

Definition at line 128 of file `stl_set.h`.

**5.638.2.14** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Compare  
std::set< _Key, _Compare, _Alloc >::value_compare`

Public typedefs.

Definition at line 103 of file `stl_set.h`.

**5.638.2.15** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<  
_Key, _Compare, _Alloc >::value_type`

Public typedefs.

Definition at line 101 of file stl\_set.h.

### 5.638.3 Constructor & Destructor Documentation

**5.638.3.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set< _Key,  
_Compare, _Alloc >::set ( ) [inline]`

Default constructor creates no elements.

Definition at line 136 of file stl\_set.h.

**5.638.3.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set< _Key,  
_Compare, _Alloc >::set ( const _Compare & __comp, const  
allocator_type & __a = allocator_type ( ) ) [inline,  
explicit]`

Creates a set with no elements.

#### Parameters

*comp* Comparator to use.

*a* An allocator object.

Definition at line 145 of file stl\_set.h.

**5.638.3.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator > std::set< _Key, _Compare, _Alloc >::set (   
_InputIterator __first, _InputIterator __last ) [inline]`

Builds a set from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 159 of file stl\_set.h.

```
5.638.3.4  template<typename _Key, typename _Compare = std::less<_Key>,
            typename _Alloc = std::allocator<_Key>> template<typename
            _InputIterator > std::set<_Key, _Compare, _Alloc >::set (
            _InputIterator __first, _InputIterator __last, const _Compare
            & __comp, const allocator_type & __a = allocator_type () )
            [inline]
```

Builds a set from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 175 of file stl\_set.h.

```
5.638.3.5  template<typename _Key, typename _Compare = std::less<_Key>,
            typename _Alloc = std::allocator<_Key>> std::set<_Key,
            _Compare, _Alloc >::set ( const set<_Key, _Compare, _Alloc > &
            __x ) [inline]
```

Set copy constructor.

#### Parameters

*x* A set of identical element and allocator types.

The newly-created set uses a copy of the allocation object used by *x*.

Definition at line 188 of file stl\_set.h.

```
5.638.3.6 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::set< _Key,
    _Compare, _Alloc >::set ( set< _Key, _Compare, _Alloc > && __x
    ) [inline]
```

Set move constructor

#### Parameters

*x* A set of identical element and allocator types.

The newly-created set contains the exact contents of *x*. The contents of *x* are a valid, but unspecified set.

Definition at line 199 of file stl\_set.h.

```
5.638.3.7 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::set< _Key,
    _Compare, _Alloc >::set ( initializer_list< value_type > __l, const
    _Compare & __comp = _Compare (), const allocator_type & __a =
    allocator_type () ) [inline]
```

Builds a set from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#).

*comp* A comparison functor.

*a* An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in *N* if the list is already sorted, and *N*log*N* otherwise (where *N* is *l.size()*).

Definition at line 212 of file stl\_set.h.

## 5.638.4 Member Function Documentation

```
5.638.4.1 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
    _Compare, _Alloc >::begin ( ) const [inline]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 292 of file stl\_set.h.

**5.638.4.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 329 of file `stl_set.h`.

**5.638.4.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 338 of file `stl_set.h`.

**5.638.4.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::set<_Key,  
_Compare, _Alloc >::clear ( ) [inline]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 552 of file `stl_set.h`.

**5.638.4.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc >::count ( const key_type & __x ) const  
[inline]`

Finds the number of elements.

#### Parameters

`x` Element to located.

#### Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 566 of file `stl_set.h`.

**5.638.4.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<  
_Key, _Compare, _Alloc >::crbegin( ) const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 347 of file stl\_set.h.

**5.638.4.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<  
_Key, _Compare, _Alloc >::crend( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 356 of file stl\_set.h.

**5.638.4.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> bool std::set< _Key,  
_Compare, _Alloc >::empty( ) const [inline]`

Returns true if the set is empty.

Definition at line 362 of file stl\_set.h.

**5.638.4.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,  
_Compare, _Alloc >::end( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 301 of file stl\_set.h.

**5.638.4.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
iterator> std::set< _Key, _Compare, _Alloc >::equal_range( const  
key_type & __x ) [inline]`

Finds a subsequence matching given key.

#### Parameters

**x** Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 646 of file stl\_set.h.

```
5.638.4.11 template<typename _Key, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::set<_Key,
_Compare, _Alloc >::equal_range ( const key_type & __x ) const
[inline]
```

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 650 of file stl\_set.h.

```
5.638.4.12 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::erase ( iterator __position ) [inline]
```

Erases an element from a set.

**Parameters**

*position* An iterator pointing to the element to be erased.

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 478 of file `stl_set.h`.

```
5.638.4.13 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> size_type std::set<_Key,
                _Compare, _Alloc >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

**Parameters**

*x* Key of element to be erased.

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 507 of file `stl_set.h`.

```
5.638.4.14 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
                _Compare, _Alloc >::erase ( iterator __first, iterator __last )
                [inline]
```

Erases a `[first,last)` range of elements from a set.

**Parameters**

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

**Returns**

The iterator *last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 526 of file `stl_set.h`.

```
5.638.4.15 template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc >::find ( const key_type & __x ) [inline]
```

Tries to locate an element in a set.

**Parameters**

**x** Element to be located.

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 584 of file `stl_set.h`.

```
5.638.4.16 template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc >::find ( const key_type & __x ) const  
[inline]
```

Tries to locate an element in a set.

**Parameters**

**x** Element to be located.

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 588 of file `stl_set.h`.

**5.638.4.17** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> allocator_type std::set<  
_Key, _Compare, _Alloc >::get_allocator ( ) const [inline]`

Returns the allocator object with which the set was constructed.

Definition at line 283 of file stl\_set.h.

**5.638.4.18** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
bool> std::set<_Key, _Compare, _Alloc >::insert ( const  
value_type & __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

*x* Element to be inserted.

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 405 of file stl\_set.h.

References std::pair< \_T1, \_T2 >::first, and std::pair< \_T1, \_T2 >::second.

**5.638.4.19** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc >::insert ( iterator __position, const value_type  
& __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

*position* An iterator that serves as a hint as to where the element should be inserted.

*x* Element to be inserted.

#### Returns

An iterator that points to the element with key of *x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 432 of file `stl_set.h`.

```
5.638.4.20 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> template<typename
                _InputIterator > void std::set<_Key, _Compare, _Alloc >::insert (
                _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 446 of file `stl_set.h`.

```
5.638.4.21 template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>> void std::set<_Key,
                _Compare, _Alloc >::insert ( initializer_list< value_type > __l )
                [inline]
```

Attempts to insert a list of elements into the set.

#### Parameters

*list* A `std::initializer_list<value_type>` of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 458 of file `stl_set.h`.

References `std::set<_Key, _Compare, _Alloc >::insert()`.

Referenced by `std::set<_Key, _Compare, _Alloc >::insert()`.

**5.638.4.22** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> key_compare std::set<  
_Key, _Compare, _Alloc >::key_comp ( ) const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 275 of file stl\_set.h.

**5.638.4.23** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,  
_Compare, _Alloc >::lower_bound ( const key_type & __x )  
[inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 605 of file stl\_set.h.

**5.638.4.24** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc >::lower_bound ( const key_type & __x )  
const [inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 609 of file stl\_set.h.

**5.638.4.25** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc >::max_size ( ) const [inline]`

Returns the maximum size of the set.

Definition at line 372 of file `stl_set.h`.

**5.638.4.26** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,  
_Compare, _Alloc >::operator= ( set< _Key, _Compare, _Alloc >  
&& __x ) [inline]`

Set move assignment operator.

#### Parameters

*x* A set of identical element and allocator types.

The contents of *x* are moved into this set (without copying). *x* is a valid, but unspecified set.

Definition at line 242 of file `stl_set.h`.

References `std::swap()`.

**5.638.4.27** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,  
_Compare, _Alloc >::operator= ( initializer_list< value_type > __l  
) [inline]`

Set list assignment operator.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a set with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 263 of file `stl_set.h`.

**5.638.4.28** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set<_Key,  
_Compare, _Alloc >::operator=( const set<_Key, _Compare,  
_Alloc > & _x ) [inline]`

Set assignment operator.

#### Parameters

*x* A set of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 227 of file stl\_set.h.

**5.638.4.29** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::set<_Key, _Compare, _Alloc >::rbegin( ) const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 310 of file stl\_set.h.

**5.638.4.30** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::set<_Key, _Compare, _Alloc >::rend( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 319 of file stl\_set.h.

**5.638.4.31** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc >::size( ) const [inline]`

Returns the size of the set.

Definition at line 367 of file stl\_set.h.

**5.638.4.32** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::set< _Key,  
_Compare, _Alloc >::swap ( set< _Key, _Compare, _Alloc > & __x  
) [inline]`

Swaps data with another set.

#### Parameters

**x** A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 387 of file `stl_set.h`.

Referenced by `std::swap()`.

**5.638.4.33** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,  
_Compare, _Alloc >::upper_bound ( const key_type & __x )  
[inline]`

Finds the end of a subsequence matching given key.

#### Parameters

**x** Key to be located.

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 621 of file `stl_set.h`.

**5.638.4.34** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc >::upper_bound ( const key_type & __x )  
const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

**x** Key to be located.

**Returns**

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 625 of file `stl_set.h`.

**5.638.4.35** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> value_compare std::set<  
_Key, _Compare, _Alloc >::value_comp( ) const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 279 of file `stl_set.h`.

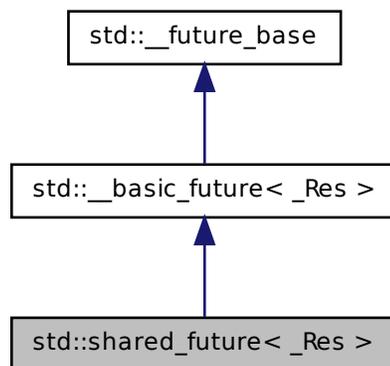
The documentation for this class was generated from the following file:

- [stl\\_set.h](#)

## 5.639 `std::shared_future<_Res>` Class Template Reference

Primary template for [shared\\_future](#).

Inheritance diagram for `std::shared_future<_Res>`:



## Public Member Functions

- [shared\\_future](#) (const [shared\\_future](#) &\_\_sf)
- [shared\\_future](#) ([shared\\_future](#) &&\_\_sf)
- [shared\\_future](#) ([future](#)< \_Res > &&\_\_uf)
- const \_Res & [get](#) ()
- [shared\\_future](#) & **operator=** (const [shared\\_future](#) &\_\_sf)
- [shared\\_future](#) & **operator=** ([shared\\_future](#) &&\_\_sf)
- bool **valid** () const
- void **wait** () const
- template<typename \_Rep, typename \_Period >  
bool **wait\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration >  
bool **wait\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs)  
const

## Protected Types

- typedef [\\_\\_future\\_base::Result](#)< \_Res > & **\_\_result\_type**
- typedef [shared\\_ptr](#)< \_State > **\_\_state\_type**

## Protected Member Functions

- [\\_\\_result\\_type](#) [\\_M\\_get\\_result](#) ()
- void [\\_M\\_swap](#) ([\\_\\_basic\\_future](#) &\_\_that)

### 5.639.1 Detailed Description

template<typename \_Res> class [std::shared\\_future](#)< \_Res >

Primary template for [shared\\_future](#).

Definition at line 683 of file future.

### 5.639.2 Constructor & Destructor Documentation

**5.639.2.1** template<typename \_Res > [std::shared\\_future](#)< \_Res >::[shared\\_future](#) ( const [shared\\_future](#)< \_Res > & \_\_sf )  
[**inline**]

Copy constructor.

Definition at line 691 of file future.

**5.639.2.2** `template<typename _Res > std::shared_future< _Res >::shared_future ( future< _Res > && __uf ) [inline]`

Construct from a future rvalue.

Definition at line 694 of file future.

**5.639.2.3** `template<typename _Res > std::shared_future< _Res >::shared_future ( shared_future< _Res > && __sf ) [inline]`

Construct from a [shared\\_future](#) rvalue.

Definition at line 699 of file future.

### 5.639.3 Member Function Documentation

**5.639.3.1** `template<typename _Res> __result_type std::__basic_future< _Res >::M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

**5.639.3.2** `template<typename _Res > const _Res& std::shared_future< _Res >::get ( ) [inline]`

Retrieving the value.

Definition at line 717 of file future.

References `std::__basic_future< _Res >::M_get_result()`.

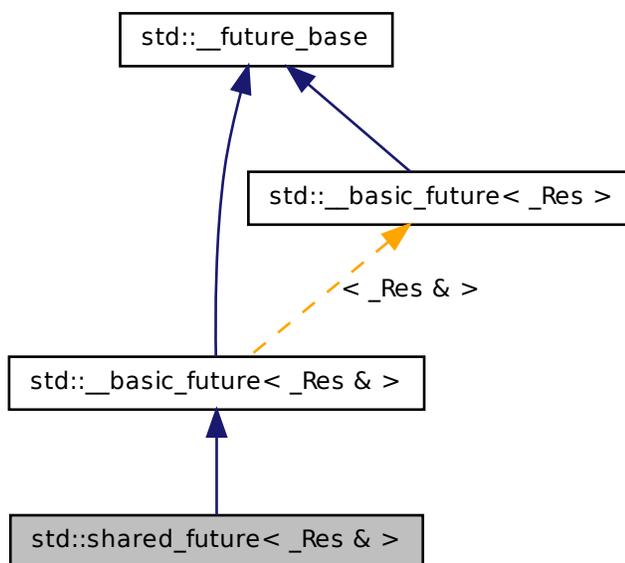
The documentation for this class was generated from the following file:

- [future](#)

## 5.640 `std::shared_future< _Res & >` Class Template Reference

Partial specialization for `shared_future<R&>`

Inheritance diagram for `std::shared_future<_Res & >`:



## Public Member Functions

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (`shared_future` &&\_\_sf)
- `shared_future` (`future<_Res & >` &&\_\_uf)
- `_Res & get ()`
- `shared_future & operator=` (const `shared_future` &\_\_sf)
- `shared_future & operator=` (`shared_future` &&\_\_sf)
- `bool valid () const`
- `void wait () const`
- `bool wait_for` (const `chrono::duration<_Rep, _Period >` &\_\_rel) const
- `bool wait_until` (const `chrono::time_point<_Clock, _Duration >` &\_\_abs) const

## Protected Types

- typedef `__future_base::Result<_Res &>` & `__result_type`
- typedef `shared_ptr<_State>` `__state_type`

## Protected Member Functions

- `__result_type M_get_result()`
- `void M_swap(__basic_future &__that)`

### 5.640.1 Detailed Description

`template<typename _Res> class std::shared_future<_Res &>`

Partial specialization for `shared_future<R&>`

Definition at line 727 of file `future`.

### 5.640.2 Constructor & Destructor Documentation

**5.640.2.1** `template<typename _Res > std::shared_future<_Res &>::shared_future ( const shared_future<_Res &> & __sf )`  
`[inline]`

Copy constructor.

Definition at line 735 of file `future`.

**5.640.2.2** `template<typename _Res > std::shared_future<_Res &>::shared_future ( future<_Res &> && __uf )` `[inline]`

Construct from a future rvalue.

Definition at line 738 of file `future`.

**5.640.2.3** `template<typename _Res > std::shared_future<_Res &>::shared_future ( shared_future<_Res &> && __sf )`  
`[inline]`

Construct from a `shared_future` rvalue.

Definition at line 743 of file `future`.

### 5.640.3 Member Function Documentation

#### 5.640.3.1 `__result_type std::__basic_future<_Res & >::_M_get_result ( )` `[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

References `std::rethrow_exception()`.

#### 5.640.3.2 `template<typename _Res > _Res& std::shared_future<_Res & >::get ( ) [inline]`

Retrieving the value.

Definition at line 761 of file future.

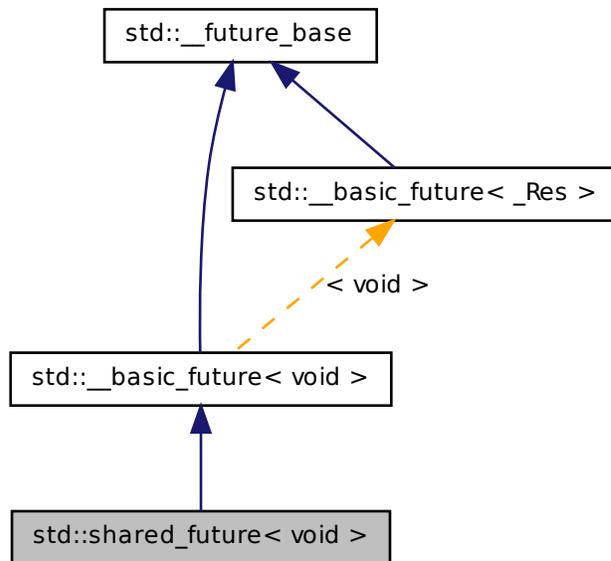
The documentation for this class was generated from the following file:

- [future](#)

## 5.641 `std::shared_future< void >` Class Template Reference

Explicit specialization for [shared\\_future<void>](#)

Inheritance diagram for `std::shared_future< void >`:



## Public Member Functions

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (`shared_future` &&\_\_sf)
- `shared_future` (`future< void >` &&\_\_uf)
- void `get` ()
- `shared_future` & `operator=` (const `shared_future` &\_\_sf)
- `shared_future` & `operator=` (`shared_future` &&\_\_sf)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration< _Rep, _Period >` &\_\_rel) const
- bool `wait_until` (const `chrono::time_point< _Clock, _Duration >` &\_\_abs) const

## Protected Types

- typedef `__future_base::_Result`< void > & `__result_type`
- typedef `shared_ptr`< `_State` > `__state_type`

## Protected Member Functions

- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` & `__that`)

### 5.641.1 Detailed Description

`template<> class std::shared_future< void >`

Explicit specialization for `shared_future<void>`

Definition at line 766 of file future.

### 5.641.2 Constructor & Destructor Documentation

**5.641.2.1** `std::shared_future< void >::shared_future ( const shared_future< void > & __sf ) [inline]`

Copy constructor.

Definition at line 774 of file future.

**5.641.2.2** `std::shared_future< void >::shared_future ( future< void > && __uf ) [inline]`

Construct from a future rvalue.

Definition at line 777 of file future.

**5.641.2.3** `std::shared_future< void >::shared_future ( shared_future< void > && __sf ) [inline]`

Construct from a `shared_future` rvalue.

Definition at line 782 of file future.

### 5.641.3 Member Function Documentation

#### 5.641.3.1 `__result_type std::__basic_future< void >::_M_get_result ( )` [inline, protected, inherited]

Wait for the state to be ready and rethrow any stored exception.

Definition at line 513 of file future.

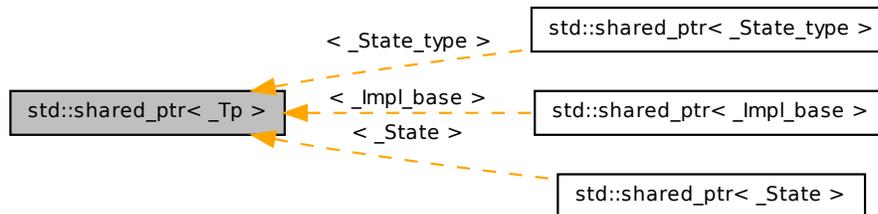
The documentation for this class was generated from the following file:

- [future](#)

## 5.642 std::shared\_ptr< \_Tp > Class Template Reference

A smart pointer with reference-counted copy semantics.

Inheritance diagram for std::shared\_ptr< \_Tp >:



### Public Member Functions

- [shared\\_ptr](#) ( )
- `template<typename _Tp1 >`  
[shared\\_ptr](#) (\_Tp1 \*\_\_p)
- `template<typename _Tp1, typename _Deleter, typename _Alloc >`  
[shared\\_ptr](#) (\_Tp1 \*\_\_p, \_Deleter \_\_d, const \_Alloc &\_\_a)
- `template<typename _Tp1 >`  
[shared\\_ptr](#) (shared\_ptr< \_Tp1 > &&\_\_r)

- `template<typename _Tp1 >`  
`shared_ptr` (const `weak_ptr`< `_Tp1` > &\_\_r)
- `template<typename _Tp1 >`  
`shared_ptr` (const `shared_ptr`< `_Tp1` > &\_\_r, `_Tp` \*\_\_p)
- `template<typename _Tp1 >`  
`shared_ptr` (`std::auto_ptr`< `_Tp1` > &&\_\_r)
- `template<typename _Tp1, typename _Del >`  
`shared_ptr` (`std::unique_ptr`< `_Tp1`, `_Del` > &&\_\_r)
- `template<typename _Tp1, typename _Deleter >`  
`shared_ptr` (`_Tp1` \*\_\_p, `_Deleter` \_\_d)
- `template<typename _Tp1 >`  
`shared_ptr` (const `shared_ptr`< `_Tp1` > &\_\_r)
- `shared_ptr` (`shared_ptr` &&\_\_r)
- `template<class _Tp1 >`  
`shared_ptr` & `operator=` (`shared_ptr`< `_Tp1` > &&\_\_r)
- `template<typename _Tp1 >`  
`shared_ptr` & `operator=` (const `shared_ptr`< `_Tp1` > &\_\_r)
- `shared_ptr` & `operator=` (`shared_ptr` &&\_\_r)
- `template<typename _Tp1, typename _Del >`  
`shared_ptr` & `operator=` (`std::unique_ptr`< `_Tp1`, `_Del` > &&\_\_r)
- `template<typename _Tp1 >`  
`shared_ptr` & `operator=` (`std::auto_ptr`< `_Tp1` > &&\_\_r)

## Friends

- `template<typename _Tp1, typename _Alloc, typename... _Args>`  
`shared_ptr`< `_Tp1` > `allocate_shared` (`_Alloc` \_\_a, `_Args` &&...\_\_args)

### 5.642.1 Detailed Description

`template<typename _Tp> class std::shared_ptr< _Tp >`

A smart pointer with reference-counted copy semantics. The object pointed to is deleted when the last `shared_ptr` pointing to it is destroyed or reset.

Definition at line 91 of file `shared_ptr.h`.

### 5.642.2 Constructor & Destructor Documentation

**5.642.2.1** `template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr ( )`  
[`inline`]

Construct an empty `shared_ptr`.

**Postcondition**

use\_count()==0 && get()==0

Definition at line 98 of file shared\_ptr.h.

**5.642.2.2** `template<typename _Tp> template<typename _Tp1 >  
std::shared_ptr< _Tp >::shared_ptr ( _Tp1 * __p ) [inline,  
explicit]`

Construct a shared\_ptr that owns the pointer `__p`.

**Parameters**

`__p` A pointer that is convertible to `element_type*`.

**Postcondition**

use\_count() == 1 && get() == \_\_p

**Exceptions**

*std::bad\_alloc*, *in* which case `delete __p` is called.

Definition at line 107 of file shared\_ptr.h.

**5.642.2.3** `template<typename _Tp> template<typename _Tp1 , typename  
_Deleter > std::shared_ptr< _Tp >::shared_ptr ( _Tp1 * __p,  
_Deleter __d ) [inline]`

Construct a shared\_ptr that owns the pointer `__p` and the deleter `__d`.

**Parameters**

`__p` A pointer.

`__d` A deleter.

**Postcondition**

use\_count() == 1 && get() == \_\_p

**Exceptions**

*std::bad\_alloc*, *in* which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 123 of file shared\_ptr.h.

**5.642.2.4** `template<typename _Tp> template<typename _Tp1, typename _Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr ( _Tp1 * __p, _Deleter __d, const _Alloc & __a ) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

#### Parameters

`__p` A pointer.  
`__d` A deleter.  
`__a` An allocator.

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

`std::bad_alloc`, in which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 141 of file `shared_ptr.h`.

**5.642.2.5** `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( const shared_ptr< _Tp1 > & __r, _Tp * __p ) [inline]`

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

#### Parameters

`__r` A `shared_ptr`.  
`__p` A pointer that will remain valid while `*__r` is valid.

#### Postcondition

`get() == __p && use_count() == __r.use_count()`

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 163 of file `shared_ptr.h`.

```
5.642.2.6 template<typename _Tp> template<typename _Tp1 >
std::shared_ptr<_Tp >::shared_ptr ( const shared_ptr<_Tp1 > &
__r ) [inline]
```

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

**Parameters**

`__r` A `shared_ptr`.

**Postcondition**

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 174 of file `shared_ptr.h`.

```
5.642.2.7 template<typename _Tp> std::shared_ptr<_Tp >::shared_ptr (
shared_ptr<_Tp > && __r ) [inline]
```

Move-constructs a `shared_ptr` instance from `__r`.

**Parameters**

`__r` A `shared_ptr` rvalue.

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 181 of file `shared_ptr.h`.

```
5.642.2.8 template<typename _Tp> template<typename _Tp1 >
std::shared_ptr<_Tp >::shared_ptr ( shared_ptr<_Tp1 > && __r
) [inline]
```

Move-constructs a `shared_ptr` instance from `__r`.

**Parameters**

`__r` A `shared_ptr` rvalue.

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 190 of file `shared_ptr.h`.

**5.642.2.9** `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr ( const weak_ptr< _Tp1 > & __r ) [inline, explicit]`

Constructs a `shared_ptr` that shares ownership with `__r` and stores a copy of the pointer stored in `__r`.

#### Parameters

`__r` A [weak\\_ptr](#).

#### Postcondition

`use_count() == __r.use_count()`

#### Exceptions

*bad\_weak\_ptr* when `__r.expired()`, in which case the constructor has no effect.

Definition at line 202 of file `shared_ptr.h`.

### 5.642.3 Friends And Related Function Documentation

**5.642.3.1** `template<typename _Tp> template<typename _Tp1 , typename _Alloc , typename... _Args> shared_ptr<_Tp1> allocate_shared ( _Alloc __a, _Args &&... __args ) [friend]`

Create an object that is owned by a [shared\\_ptr](#).

#### Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

#### Returns

A [shared\\_ptr](#) that owns the newly created object.

#### Exceptions

*An* exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared\\_ptr](#) and the new object.

Definition at line 452 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.643 `std::shuffle_order_engine<_RandomNumberEngine, __k >` Class Template Reference

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

### Public Types

- typedef `_RandomNumberEngine::result_type` [result\\_type](#)

### Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (const `_RandomNumberEngine &__rne`)
- [shuffle\\_order\\_engine](#) ([result\\_type \\_\\_s](#))
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>` [shuffle\\_order\\_engine](#) (`_Sseq &__q`)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine &&__rne`)
- const `_RandomNumberEngine &` [base](#) () const
- void [discard](#) (unsigned long long `__z`)
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- [result\\_type operator\(\)](#) ()
- void [seed](#) ([result\\_type \\_\\_s](#))
- `template<typename _Sseq >` void [seed](#) (`_Sseq &__q`)
- void [seed](#) ()

### Static Public Attributes

- static const `size_t` [table\\_size](#)

### Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits > &` [operator<<](#) (`std::basic_ostream<_CharT, _Traits > &`, const `std::shuffle_order_engine<_RandomNumberEngine1, __k1 >` &)

- `bool operator==(const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)`
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>>(std::basic_istream< _CharT, _Traits > &, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)`

### 5.643.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __k> class std::shuffle_order_engine< _RandomNumberEngine, __k >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1234 of file random.h.

### 5.643.2 Member Typedef Documentation

**5.643.2.1** `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1241 of file random.h.

### 5.643.3 Constructor & Destructor Documentation

**5.643.3.1** `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine( ) [inline]`

Constructs a default `shuffle_order_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1250 of file random.h.

```
5.643.3.2 template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine ( const _RandomNumberEngine & __rne )
[inline, explicit]
```

Copy constructs a `shuffle_order_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 1261 of file `random.h`.

```
5.643.3.3 template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine ( _RandomNumberEngine && __rne )
[inline, explicit]
```

Move constructs a `shuffle_order_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 1272 of file `random.h`.

```
5.643.3.4 template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine<_RandomNumberEngine, __k
>::shuffle_order_engine ( result_type __s ) [inline,
explicit]
```

Seed constructs a `shuffle_order_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

#### Parameters

`__s` A seed value for the base class engine.

Definition at line 1283 of file `random.h`.

```

5.643.3.5 template<typename _RandomNumberEngine, size_t
           __k> template<typename _Sseq, typename = typename
           std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value
           && !std::is_same<_Sseq, _RandomNumberEngine>::value>
           ::type> std::shuffle_order_engine< _RandomNumberEngine, __k
           >::shuffle_order_engine( _Sseq & __q ) [inline, explicit]

```

Generator construct a shuffle\_order\_engine engine.

#### Parameters

*\_\_q* A seed sequence.

Definition at line 1297 of file random.h.

### 5.643.4 Member Function Documentation

```

5.643.4.1 template<typename _RandomNumberEngine, size_t __k>
           const _RandomNumberEngine& std::shuffle_order_engine<
           _RandomNumberEngine, __k >::base( ) const [inline]

```

Gets a const reference to the underlying generator engine object.

Definition at line 1340 of file random.h.

```

5.643.4.2 template<typename _RandomNumberEngine, size_t __k> void
           std::shuffle_order_engine< _RandomNumberEngine, __k >::discard
           ( unsigned long long __z ) [inline]

```

Discard a sequence of random numbers.

#### Todo

Look for a faster way to do discard.

Definition at line 1367 of file random.h.

```

5.643.4.3 template<typename _RandomNumberEngine, size_t __k>
           result_type std::shuffle_order_engine< _RandomNumberEngine, __k
           >::max( ) const [inline]

```

Gets the maximum value in the generated random number range.

#### Todo

This should be constexpr.

Definition at line 1358 of file `random.h`.

**5.643.4.4** `template<typename _RandomNumberEngine, size_t __k>`  
`result_type std::shuffle_order_engine<_RandomNumberEngine, __k`  
`>::min( ) const [inline]`

Gets the minimum value in the generated random number range.

#### Todo

This should be `constexpr`.

Definition at line 1349 of file `random.h`.

**5.643.4.5** `template<typename _RandomNumberEngine, size_t __k>`  
`shuffle_order_engine<_RandomNumberEngine, __k >::result_type`  
`std::shuffle_order_engine<_RandomNumberEngine, __k`  
`>::operator()( )`

Gets the next value in the generated random number sequence.

Definition at line 767 of file `random.tcc`.

**5.643.4.6** `template<typename _RandomNumberEngine, size_t __k>`  
`template<typename _Sseq > void std::shuffle_order_engine<`  
`_RandomNumberEngine, __k >::seed( _Sseq & __q ) [inline]`

Reseeds the `shuffle_order_engine` object with the given seed sequence.

#### Parameters

`__q` A seed generator function.

Definition at line 1330 of file `random.h`.

**5.643.4.7** `template<typename _RandomNumberEngine, size_t __k> void`  
`std::shuffle_order_engine<_RandomNumberEngine, __k >::seed(`  
`result_type __s ) [inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1317 of file `random.h`.

**5.643.4.8** `template<typename _RandomNumberEngine, size_t __k> void  
std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ( )  
[inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1306 of file `random.h`.

## 5.643.5 Friends And Related Function Documentation

**5.643.5.1** `template<typename _RandomNumberEngine, size_t __k>  
template<typename _RandomNumberEngine1 , size_t __k1,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & ,  
const std::shuffle_order_engine< _RandomNumberEngine1, __k1 >  
& ) [friend]`

Inserts the current state of a `shuffle_order_engine` random number generator engine `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `shuffle_order_engine` random number generator engine.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.643.5.2** `template<typename _RandomNumberEngine, size_t __k> bool  
operator==( const shuffle_order_engine< _RandomNumberEngine,  
__k > & __lhs, const shuffle_order_engine<  
_RandomNumberEngine, __k > & __rhs ) [friend]`

Compares two `shuffle_order_engine` random number generator objects of the same type for equality.

### Parameters

`__lhs` A `shuffle_order_engine` random number generator object.

`__rhs` Another `shuffle_order_engine` random number generator object.

### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1391 of file random.h.

```
5.643.5.3 template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1,
typename _CharT, typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > &,
std::shuffle_order_engine<_RandomNumberEngine1, __k1 > & )
[friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A shuffle\_order\_engine random number generator engine.

#### Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.644 std::slice Class Reference

Class defining one-dimensional subset of an array.

### Public Member Functions

- [slice](#) ()
- [slice](#) (size\_t, size\_t, size\_t)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

#### 5.644.1 Detailed Description

Class defining one-dimensional subset of an array. The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and

stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 58 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.645 `std::slice_array< _Tp >` Class Template Reference

Reference to one-dimensional subset of an array.

### Public Types

- typedef `_Tp value_type`

### Public Member Functions

- [slice\\_array](#) (const [slice\\_array](#) &)
- template<class `_Dom` >  
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator%=** (const [valarray](#)< `_Tp` > &) const
- void **operator&=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >  
void **operator&=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- template<class `_Dom` >  
void **operator\*=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator\*=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >  
void **operator+=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator+=** (const [valarray](#)< `_Tp` > &) const
- void **operator-=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >  
void **operator-=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- template<class `_Dom` >  
void **operator/=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void **operator/=** (const [valarray](#)< `_Tp` > &) const

- void `operator<<=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator<<=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator=` (const `_Tp` &) const
- `slice_array` & `operator=` (const `slice_array` &)
- template<class `_Dom`>  
void `operator=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator>>=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator>>=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator^=` (const `_Expr<_Dom, _Tp>` &) const
- void `operator^=` (const `valarray<_Tp>` &) const
- void `operator|=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `operator|=` (const `_Expr<_Dom, _Tp>` &) const

## Friends

- class `valarray<_Tp>`

### 5.645.1 Detailed Description

`template<typename _Tp> class std::slice_array<_Tp>`

Reference to one-dimensional subset of an array. A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[]`(slice) on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

#### Parameters

*`Tp`* Element type.

Definition at line 122 of file `slice_array.h`.

## 5.645.2 Member Function Documentation

**5.645.2.1** `template<typename _Tp> void std::slice_array< _Tp >::operator%=( const valarray< _Tp > & ) const`

Modulo slice elements by corresponding elements of  $v$ .

**5.645.2.2** `template<typename _Tp> void std::slice_array< _Tp >::operator&=( const valarray< _Tp > & ) const`

Logical and slice elements with corresponding elements of  $v$ .

**5.645.2.3** `template<typename _Tp> void std::slice_array< _Tp >::operator*=( const valarray< _Tp > & ) const`

Multiply slice elements by corresponding elements of  $v$ .

**5.645.2.4** `template<typename _Tp> void std::slice_array< _Tp >::operator+=( const valarray< _Tp > & ) const`

Add corresponding elements of  $v$  to slice elements.

**5.645.2.5** `template<typename _Tp> void std::slice_array< _Tp >::operator-=( const valarray< _Tp > & ) const`

Subtract corresponding elements of  $v$  from slice elements.

**5.645.2.6** `template<typename _Tp> void std::slice_array< _Tp >::operator/=( const valarray< _Tp > & ) const`

Divide slice elements by corresponding elements of  $v$ .

**5.645.2.7** `template<typename _Tp> void std::slice_array< _Tp >::operator<<=( const valarray< _Tp > & ) const`

Left shift slice elements by corresponding elements of  $v$ .

**5.645.2.8** `template<typename _Tp> void std::slice_array< _Tp >::operator>>=( const valarray< _Tp > & ) const`

Right shift slice elements by corresponding elements of  $v$ .

**5.645.2.9** `template<typename _Tp> void std::slice_array< _Tp >::operator^= ( const valarray< _Tp > & ) const`

Logical xor slice elements with corresponding elements of *v*.

**5.645.2.10** `template<typename _Tp> void std::slice_array< _Tp >::operator|= ( const valarray< _Tp > & ) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.646 `std::stack< _Tp, _Sequence >` Class Template Reference

A standard container giving FILO behavior.

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

### Public Member Functions

- `stack` (const `_Sequence` &\_\_c)
- `stack` (`_Sequence` &&\_\_c=`_Sequence`())
- `template<typename... _Args> void emplace` (`_Args` &&...\_\_args)
- `bool empty` () const
- `void pop` ()
- `void push` (`value_type` &&\_\_x)
- `void push` (const `value_type` &\_\_x)
- `size_type size` () const
- `void swap` (`stack` &\_\_s)
- `const_reference top` () const
- `reference top` ()

## Protected Attributes

- `_Sequence c`

## Friends

- `template<typename _Tp1, typename _Seq1 >`  
`bool operator< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 >`  
`&)`
- `template<typename _Tp1, typename _Seq1 >`  
`bool operator== (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 >`  
`&)`

### 5.646.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>> class
std::stack<_Tp, _Sequence >
```

A standard container giving FILO behavior. Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_front`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 92 of file `stl_stack.h`.

### 5.646.2 Constructor & Destructor Documentation

```
5.646.2.1 template<typename _Tp, typename _Sequence = deque<_Tp>>
std::stack<_Tp, _Sequence >::stack ( const _Sequence & __c )
[inline, explicit]
```

Default constructor creates no elements.

Definition at line 130 of file `stl_stack.h`.

### 5.646.3 Member Function Documentation

**5.646.3.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
bool std::stack< _Tp, _Sequence >::empty ( ) const [inline]`

Returns true if the stack is empty.

Definition at line 142 of file stl\_stack.h.

**5.646.3.2** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
void std::stack< _Tp, _Sequence >::pop ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 208 of file stl\_stack.h.

**5.646.3.3** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
void std::stack< _Tp, _Sequence >::push ( const value_type & __x )  
[inline]`

Add data to the top of the stack.

#### Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 182 of file stl\_stack.h.

**5.646.3.4** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
size_type std::stack< _Tp, _Sequence >::size ( ) const [inline]`

Returns the number of elements in the stack.

Definition at line 147 of file stl\_stack.h.

**5.646.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
reference std::stack<_Tp, _Sequence >::top ( ) [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 155 of file `stl_stack.h`.

**5.646.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
const_reference std::stack<_Tp, _Sequence >::top ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 166 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 5.647 `std::student_t_distribution<_RealType > Class` Template Reference

A [student\\_t\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `student_t_distribution` (`_RealType __n=_RealType(1)`)
- `student_t_distribution` (`const param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `_RealType n` () const
- `template<typename _UniformRandomNumberGenerator >  
result_type operator()` (`_UniformRandomNumberGenerator &__urng`)

## 5.647 `std::student_t_distribution<_RealType>` Class Template Reference 2921

- `template<typename _UniformRandomNumberGenerator > result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

### Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::student_t_distribution< _RealType1 > &)`
- `template<typename _RealType1 > bool operator== (const std::student_t_distribution< _RealType1 > &__d1, const std::student_t_distribution< _RealType1 > &__d2)`
- `template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::student_t_distribution< _RealType1 > &)`

### 5.647.1 Detailed Description

`template<typename _RealType = double> class std::student_t_distribution< _RealType >`

A `student_t_distribution` random number distribution. The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3061 of file `random.h`.

### 5.647.2 Member Typedef Documentation

**5.647.2.1** `template<typename _RealType = double> typedef _RealType std::student_t_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 3068 of file `random.h`.

### 5.647.3 Member Function Documentation

**5.647.3.1** `template<typename _RealType = double> result_type  
std::student_t_distribution< _RealType >::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3144 of file random.h.

**5.647.3.2** `template<typename _RealType = double> result_type  
std::student_t_distribution< _RealType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3137 of file random.h.

**5.647.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::student_t_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3152 of file random.h.

References `std::sqrt()`.

**5.647.3.4** `template<typename _RealType = double> param_type  
std::student_t_distribution< _RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3122 of file random.h.

**5.647.3.5** `template<typename _RealType = double> void  
std::student_t_distribution< _RealType >::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

*\_\_param* The new parameter set of the distribution.

## 5.647 `std::student_t_distribution<_RealType>` Class Template Reference 2923

Definition at line 3130 of file `random.h`.

**5.647.3.6** `template<typename _RealType = double> void  
std::student_t_distribution<_RealType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 3105 of file `random.h`.

References `std::gamma_distribution<_RealType>::reset()`, and `std::normal_distribution<_RealType>::reset()`.

### 5.647.4 Friends And Related Function Documentation

**5.647.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::student_t_distribution<_RealType1> & ) [friend]`

Inserts a `student_t_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `student_t_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.647.4.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::student_t_distribution<  
_RealType1 > & __d1, const std::student_t_distribution<  
_RealType1 > & __d2 ) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3174 of file `random.h`.

```
5.647.4.3 template<typename _RealType = double> template<typename
    _RealType1, typename _CharT, typename _Traits >
    std::basic_istream<_CharT, _Traits>& operator>>
    ( std::basic_istream< _CharT, _Traits > & ,
      std::student_t_distribution< _RealType1 > & ) [friend]
```

Extracts a `student_t_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A `student_t_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.648 `std::student_t_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef `student_t_distribution< _RealType >` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

### Friends

- bool `operator==` (const `param_type` &`__p1`, const `param_type` &`__p2`)

### 5.648.1 Detailed Description

**template**<typename **\_RealType** = double> **struct** **std::student\_t\_distribution**< **\_RealType** >::**param\_type**

Parameter type.

Definition at line 3070 of file random.h.

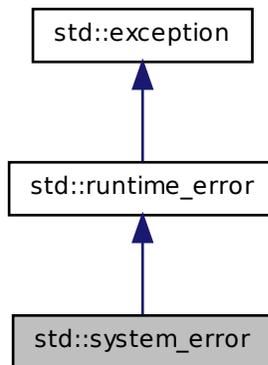
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.649 std::system\_error Class Reference

Thrown to indicate error code of underlying system.

Inheritance diagram for std::system\_error:



### Public Member Functions

- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat, const [string](#) &\_\_what)

- `system_error` (int \_\_v, const [error\\_category](#) &\_\_ecat)
- const `error_code` & `code` () const throw ()
- virtual const char \* `what` () const throw ()

### 5.649.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 307 of file `system_error`.

### 5.649.2 Member Function Documentation

#### 5.649.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.650 `std::thread` Class Reference

`thread`

### Classes

- class `id`  
*`thread::id`*

### Public Types

- typedef `shared_ptr< _Impl_base > __shared_base_type`
- typedef `__gthread_t native_handle_type`

## Public Member Functions

- **thread** (const [thread](#) &)
- template<typename \_Callable >  
**thread** (\_Callable \_\_f)
- template<typename \_Callable , typename... \_Args>  
**thread** (\_Callable &&\_\_f, \_Args &&...\_\_args)
- **thread** ([thread](#) &&\_\_t)
- void **detach** ()
- [thread::id](#) **get\_id** () const
- void **join** ()
- bool **joinable** () const
- native\_handle\_type **native\_handle** ()
- [thread](#) & **operator=** (const [thread](#) &)
- [thread](#) & **operator=** ([thread](#) &&\_\_t)
- void **swap** ([thread](#) &\_\_t)

## Static Public Member Functions

- static unsigned int **hardware\_concurrency** ()

### 5.650.1 Detailed Description

[thread](#)

Definition at line 64 of file [thread](#).

### 5.650.2 Member Function Documentation

#### 5.650.2.1 native\_handle\_type std::thread::native\_handle ( ) [**inline**]

##### Precondition

[thread](#) is joinable

Definition at line 180 of file [thread](#).

The documentation for this class was generated from the following file:

- [thread](#)

## 5.651 std::thread::id Class Reference

[thread::id](#)

## Public Member Functions

- **id** (native\_handle\_type \_\_id)

## Friends

- class **hash**< **thread::id** >
- bool **operator**< (**thread::id** \_\_x, **thread::id** \_\_y)
- template<class \_CharT, class \_Traits >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_out, **thread::id** \_\_id)
- bool **operator**== (**thread::id** \_\_x, **thread::id** \_\_y)
- class **thread**

### 5.651.1 Detailed Description

[thread::id](#)

Definition at line 72 of file thread.

The documentation for this class was generated from the following file:

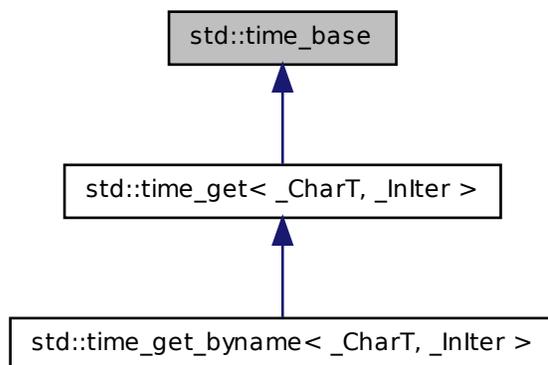
- [thread](#)

## 5.652 std::time\_base Class Reference

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Inheritance diagram for std::time\_base:



## Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ydm**,  
    **ydm** }

### 5.652.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Definition at line 50 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

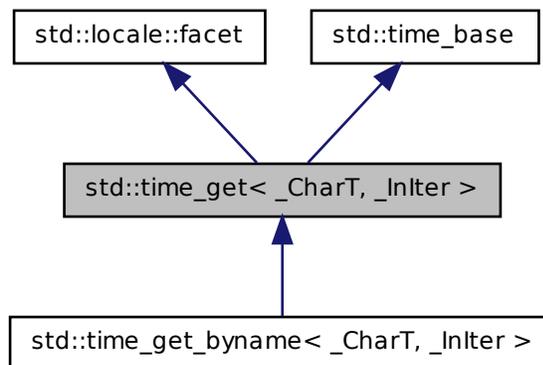
- [locale\\_facets\\_nonio.h](#)

## 5.653 `std::time_get< _CharT, _InIter >` Class Template Reference

Primary class template [time\\_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for `std::time_get< _CharT, _InIter >`:



### Public Types

- typedef [basic\\_string< \\_CharT >](#) `__string_type`
- enum `dateorder` {  
   `no_order`, `dmy`, `mdy`, `ymd`,  
   `ydm` }
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

### Public Member Functions

- [time\\_get](#) (`size_t __refs=0`)

- dateorder `date_order` () const
- `iter_type get_date` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_time` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_year` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const

## Static Public Attributes

- static `locale::id`

## Protected Member Functions

- virtual `~time_get` ()
- `iter_type _M_extract_name` (`iter_type __beg`, `iter_type __end`, `int &__member`, `const _CharT **__names`, `size_t __indexlen`, `ios_base &__io`, `ios_base::iostate &__err`) const
- `iter_type _M_extract_num` (`iter_type __beg`, `iter_type __end`, `int &__member`, `int __min`, `int __max`, `size_t __len`, `ios_base &__io`, `ios_base::iostate &__err`) const
- `iter_type _M_extract_via_format` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`, `const _CharT *__format`) const
- `iter_type _M_extract_wday_or_month` (`iter_type __beg`, `iter_type __end`, `int &__member`, `const _CharT **__names`, `size_t __indexlen`, `ios_base &__io`, `ios_base::iostate &__err`) const
- virtual dateorder `do_date_order` () const
- virtual `iter_type do_get_date` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- virtual `iter_type do_get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- virtual `iter_type do_get_time` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- virtual `iter_type do_get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- virtual `iter_type do_get_year` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Friends

- class `locale::_Impl`

### 5.653.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get< _CharT, _InIter >`

Primary class template [time\\_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators. The [time\\_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time\\_get](#) facet.

Definition at line 363 of file `locale_facets_nonio.h`.

### 5.653.2 Member Typedef Documentation

**5.653.2.1** `template<typename _CharT , typename _InIter > typedef _CharT std::time_get< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented in [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#).

Definition at line 369 of file `locale_facets_nonio.h`.

**5.653.2.2** `template<typename _CharT , typename _InIter > typedef _InIter std::time_get< _CharT, _InIter >::iter_type`

Public typedefs.

Reimplemented in `std::time_get_byname<_CharT, _InIter>`.

Definition at line 370 of file `locale_facets_nonio.h`.

### 5.653.3 Constructor & Destructor Documentation

**5.653.3.1** `template<typename _CharT, typename _InIter> std::time_get<_CharT, _InIter>::time_get( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 385 of file `locale_facets_nonio.h`.

**5.653.3.2** `template<typename _CharT, typename _InIter> virtual std::time_get<_CharT, _InIter>::~~time_get( ) [inline, protected, virtual]`

Destructor.

Definition at line 541 of file `locale_facets_nonio.h`.

### 5.653.4 Member Function Documentation

**5.653.4.1** `template<typename _CharT, typename _InIter> dateorder std::time_get<_CharT, _InIter>::date_order( ) const [inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to `time_put::put()` only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

#### Returns

A member of `timebase::dateorder`.

Definition at line 402 of file `locale_facets_nonio.h`.

**5.653.4.2** `template<typename _CharT , typename _InIter >  
 _GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder  
 std::time_get< _CharT, _InIter >::do_date_order ( ) const  
 [protected, virtual]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of `timebase::dateorder`.

Definition at line 618 of file `locale_facets_nonio.tcc`.

**5.653.4.3** `template<typename _CharT , typename _InIter > _InIter  
 std::time_get< _CharT, _InIter >::do_get_date ( iter_type __beg,  
 iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
 __tm ) const [protected, virtual]`

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_date\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct `tm` to fill in.

#### Returns

Iterator to first char beyond date string.

Definition at line 1044 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

```
5.653.4.4 template<typename _CharT , typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_monthname ( iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm ) const [protected, virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_monthname\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond month name.

Definition at line 1089 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

```
5.653.4.5 template<typename _CharT , typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_time ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [protected, virtual]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_time\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond time string.

Definition at line 1027 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

**5.653.4.6** `template<typename _CharT, typename _InIter > _InIter  
std::time_get< _CharT, _InIter >::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

### See also

[get\\_weekday\(\)](#) for details.

### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond weekday name.

Definition at line 1061 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

```
5.653.4.7 template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_year ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [protected, virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_year\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond year.

Definition at line 1117 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

```
5.653.4.8 template<typename _CharT, typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_date ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [inline]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond date string.

Definition at line 451 of file locale\_facets\_nonio.h.

```
5.653.4.9 template<typename _CharT , typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_monthname ( iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm ) const [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond month name.

Definition at line 508 of file locale\_facets\_nonio.h.

```
5.653.4.10 template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct *tm* to fill in.

#### Returns

Iterator to first char beyond time string.

Definition at line 426 of file `locale_facets_nonio.h`.

```
5.653.4.11 template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_weekday ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Definition at line 479 of file locale\_facets\_nonio.h.

```
5.653.4.12 template<typename _CharT , typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_year\(\)](#).

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond year.

Definition at line 534 of file locale\_facets\_nonio.h.

---

## 5.654 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 2941

### 5.653.5 Member Data Documentation

#### 5.653.5.1 `template<typename _CharT, typename _InIter> locale::id` `std::time_get<_CharT, _InIter>::id` [static]

Numpunct facet id.

Definition at line 375 of file `locale_facets_nonio.h`.

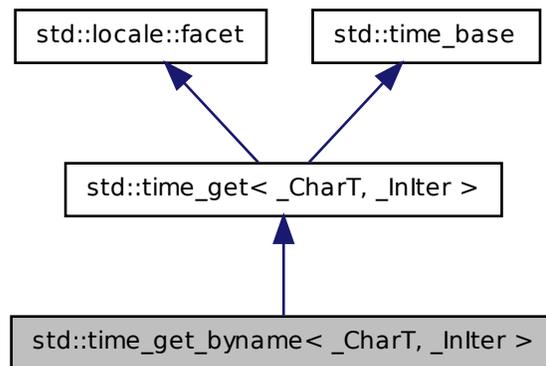
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.654 `std::time_get_byname<_CharT, _InIter>` Class Template Reference

class `time_get_byname` [22.2.5.2].

Inheritance diagram for `std::time_get_byname<_CharT, _InIter>`:



### Public Types

- typedef `basic_string<_CharT>` `__string_type`

- typedef `_CharT` `char_type`
- enum `dateorder` {  
     `no_order`, `dmy`, `mdy`, `ymd`,  
     `ydm` }
- typedef `_InIter` `iter_type`

## Public Member Functions

- `time_get_byname` (const char \*, size\_t \_\_refs=0)
- `dateorder` `date_order` () const
- `iter_type` `get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_monthname` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_time` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_weekday` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_year` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- `iter_type` `_M_extract_name` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, const `_CharT` \*\*\_\_names, size\_t \_\_indexlen, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- `iter_type` `_M_extract_num` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- `iter_type` `_M_extract_via_format` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm, const `_CharT` \*\_\_format) const
- `iter_type` `_M_extract_wday_or_month` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, const `_CharT` \*\*\_\_names, size\_t \_\_indexlen, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- virtual `dateorder` `do_date_order` () const
- virtual `iter_type` `do_get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const

## 5.654 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 2943

- virtual `iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

### 5.654.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get_byname<_CharT, _InIter>`

class `time_get_byname` [22.2.5.2].

Definition at line 681 of file `locale_facets_nonio.h`.

### 5.654.2 Member Typedef Documentation

**5.654.2.1** `template<typename _CharT, typename _InIter> typedef _CharT std::time_get_byname<_CharT, _InIter>::char_type`

Public typedefs.

Reimplemented from `std::time_get<_CharT, _InIter>`.

Definition at line 685 of file `locale_facets_nonio.h`.

### 5.654.2.2 `template<typename _CharT, typename _InIter > typedef _InIter std::time_get_byname< _CharT, _InIter >::iter_type`

Public typedefs.

Reimplemented from `std::time_get< _CharT, _InIter >`.

Definition at line 686 of file `locale_facets_nonio.h`.

## 5.654.3 Member Function Documentation

### 5.654.3.1 `template<typename _CharT, typename _InIter > dateorder std::time_get< _CharT, _InIter >::date_order ( ) const [inline, inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

#### Returns

A member of `timebase::dateorder`.

Definition at line 402 of file `locale_facets_nonio.h`.

### 5.654.3.2 `template<typename _CharT, typename _InIter > _GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order ( ) const [protected, virtual, inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of `timebase::dateorder`.

Definition at line 618 of file `locale_facets_nonio.tcc`.

## 5.654 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 2945

**5.654.3.3** `template<typename _CharT, typename _InIter > _InIter  
std::time_get<_CharT, _InIter >::do_get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual, inherited]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

### See also

[get\\_date\(\)](#) for details.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct *tm* to fill in.

### Returns

Iterator to first char beyond date string.

Definition at line 1044 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

**5.654.3.4** `template<typename _CharT, typename _InIter > _InIter  
std::time_get<_CharT, _InIter >::do_get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual, inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

### See also

[get\\_monthname\(\)](#) for details.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond month name.

Definition at line 1089 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

**5.654.3.5** `template<typename _CharT, typename _InIter > _InIter  
 std::time_get<_CharT, _InIter >::do_get_time ( iter_type __beg,  
 iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
 __tm ) const [protected, virtual, inherited]`

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

### See also

[get\\_time\(\)](#) for details.

### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond time string.

Definition at line 1027 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

## 5.654 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 2947

**5.654.3.6** `template<typename _CharT, typename _InIter > _InIter  
std::time_get<_CharT, _InIter >::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm )const [protected, virtual, inherited]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

### See also

[get\\_weekday\(\)](#) for details.

### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond weekday name.

Definition at line 1061 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

**5.654.3.7** `template<typename _CharT, typename _InIter > _InIter  
std::time_get<_CharT, _InIter >::do_get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm )const [protected, virtual, inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

### See also

[get\\_year\(\)](#) for details.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond year.

Definition at line 1117 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

**5.654.3.8** `template<typename _CharT, typename _InIter > iter_type  
 std::time_get< _CharT, _InIter >::get_date ( iter_type __beg,  
 iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
 __tm ) const [inline, inherited]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond date string.

Definition at line 451 of file locale\_facets\_nonio.h.

## 5.654 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 2949

**5.654.3.9** `template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [inline, inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond month name.

Definition at line 508 of file locale\_facets\_nonio.h.

**5.654.3.10** `template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Definition at line 426 of file locale\_facets\_nonio.h.

```
5.654.3.11 template<typename _CharT , typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_weekday ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_weekday\(\)](#).

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Definition at line 479 of file locale\_facets\_nonio.h.

## 5.654 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 2951

**5.654.3.12** `template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= `ios_base::failbit`. If parsing reads all the characters, err |= `ios_base::eofbit`.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond year.

Definition at line 534 of file locale\_facets\_nonio.h.

## 5.654.4 Member Data Documentation

**5.654.4.1** `template<typename _CharT, typename _InIter > locale::id  
std::time_get<_CharT, _InIter >::id [static, inherited]`

Numpunct facet id.

Definition at line 375 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

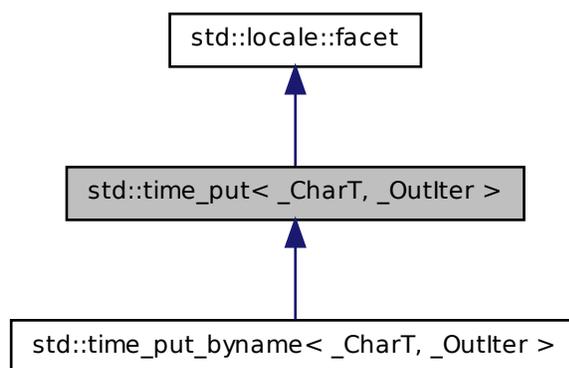
- [locale\\_facets\\_nonio.h](#)

## 5.655 `std::time_put< _CharT, _OutIter >` Class Template Reference

Primary class template [time\\_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

Inheritance diagram for `std::time_put< _CharT, _OutIter >`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

### Public Member Functions

- [time\\_put](#) (`size_t __refs=0`)
- [iter\\_type put](#) (`iter_type __s, ios_base &__io, char_type __fill, const tm * __tm, char __format, char __mod=0`) const
- [iter\\_type put](#) (`iter_type __s, ios_base &__io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end`) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~time\\_put](#) ()
- virtual [iter\\_type do\\_put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static [\\_GLIBCXX\\_CONST](#) const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

## Friends

- class [locale::\\_Impl](#)

### 5.655.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::time_put<_CharT, _OutIter >`

Primary class template [time\\_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by `strptime()`. The [time\\_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time\\_put](#) facet.

Definition at line 710 of file `locale_facets_nonio.h`.

## 5.655.2 Member Typedef Documentation

### 5.655.2.1 `template<typename _CharT, typename _OutIter > typedef _CharT std::time_put< _CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented in [std::time\\_put\\_byname< \\_CharT, \\_OutIter >](#).

Definition at line 716 of file locale\_facets\_nonio.h.

### 5.655.2.2 `template<typename _CharT, typename _OutIter > typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Reimplemented in [std::time\\_put\\_byname< \\_CharT, \\_OutIter >](#).

Definition at line 717 of file locale\_facets\_nonio.h.

## 5.655.3 Constructor & Destructor Documentation

### 5.655.3.1 `template<typename _CharT, typename _OutIter > std::time_put< _CharT, _OutIter >::time_put ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 731 of file locale\_facets\_nonio.h.

### 5.655.3.2 `template<typename _CharT, typename _OutIter > virtual std::time_put< _CharT, _OutIter >::~~time_put ( ) [inline, protected, virtual]`

Destructor.

Definition at line 777 of file locale\_facets\_nonio.h.

## 5.655.4 Member Function Documentation

**5.655.4.1** `template<typename _CharT , typename _OutIter > _OutIter  
std::time_put< _CharT, _OutIter >::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, const tm * __tm, char __format,  
char __mod ) const [protected, virtual]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

### See also

[put\(\)](#) for more details.

### Parameters

*s* The stream to write to.

*io* Source of locale.

*fill* char\_type to use for padding.

*tm* Struct tm with date and time info to format.

*format* Format char.

*mod* Optional modifier char.

### Returns

Iterator after writing.

Definition at line 1175 of file locale\_facets\_nonio.tcc.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::widen\(\)](#).

Referenced by [std::time\\_put< \\_CharT, \\_OutIter >::put\(\)](#).

**5.655.4.2** `template<typename _CharT , typename _OutIter > _OutIter  
std::time_put< _CharT, _OutIter >::put ( iter_type __s, ios_base &  
__io, char_type __fill, const tm * __tm, const _CharT * __beg,  
const _CharT * __end ) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by [strftime\(\)](#).

**Parameters**

- s* The stream to write to.
- io* Source of locale.
- fill* char\_type to use for padding.
- tm* Struct tm with date and time info to format.
- beg* Start of format string.
- end* End of format string.

**Returns**

Iterator after writing.

Definition at line 1140 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::time_put<_CharT, _OutIter >::do_put()`, and `std::__ctype_abstract_base<_CharT >::narrow()`.

**5.655.4.3** `template<typename _CharT, typename _OutIter > iter_type  
std::time_put<_CharT, _OutIter >::put ( iter_type __s, ios_base  
& __io, char_type __fill, const tm * __tm, char __format, char  
__mod = 0 ) const [inline]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

**Parameters**

- s* The stream to write to.
- io* Source of locale.
- fill* char\_type to use for padding.
- tm* Struct tm with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

**Returns**

Iterator after writing.

Definition at line 770 of file locale\_facets\_nonio.h.

References `std::time_put<_CharT, _OutIter >::do_put()`.

## 5.656 std::time\_put\_byname< \_CharT, \_OutIter > Class Template Reference 2957

### 5.655.5 Member Data Documentation

#### 5.655.5.1 `template<typename _CharT, typename _OutIter > locale::id` `std::time_put< _CharT, _OutIter >::id` [static]

Numpunct facet id.

Definition at line 721 of file locale\_facets\_nonio.h.

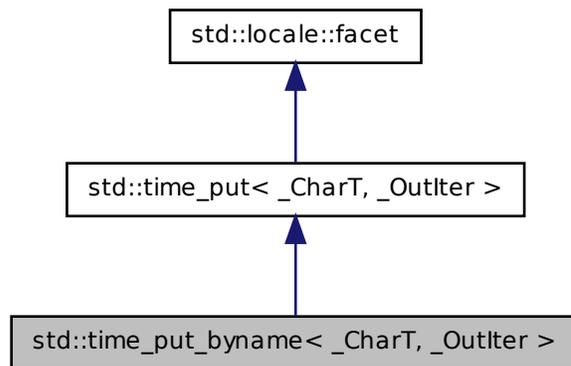
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.656 `std::time_put_byname< _CharT, _OutIter >` Class Template Reference

class `time_put_byname` [22.2.5.4].

Inheritance diagram for `std::time_put_byname< _CharT, _OutIter >`:



### Public Types

- `typedef _CharT char_type`

- typedef `_OutIter` [iter\\_type](#)

## Public Member Functions

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \* \_\_tm, char \_\_format, char \_\_mod=0) const
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \* \_\_tm, const `_CharT` \* \_\_beg, const `_CharT` \* \_\_end) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [iter\\_type do\\_put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \* \_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static `_c_locale` [\\_S\\_clone\\_c\\_locale](#) (`_c_locale` &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (`_c_locale` &\_\_cloc, const char \* \_\_s, `_c_locale` \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (`_c_locale` &\_\_cloc)
- static `_c_locale` [\\_S\\_get\\_c\\_locale](#) ()
- static `_GLIBCXX_CONST` const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static `_c_locale` [\\_S\\_lc\\_ctype\\_c\\_locale](#) (`_c_locale` \_\_cloc, const char \* \_\_s)

## Friends

- class [locale::\\_Impl](#)

### 5.656.1 Detailed Description

```
template<typename _CharT, typename _OutIter> class std::time_put_byname<
_CharT, _OutIter >
```

class [time\\_put\\_byname](#) [22.2.5.4].

Definition at line 806 of file `locale_facets_nonio.h`.

## 5.656.2 Member Typedef Documentation

### 5.656.2.1 `template<typename _CharT , typename _OutIter > typedef _CharT std::time_put_byname< _CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented from [std::time\\_put< \\_CharT, \\_OutIter >](#).

Definition at line 810 of file locale\_facets\_nonio.h.

### 5.656.2.2 `template<typename _CharT , typename _OutIter > typedef _OutIter std::time_put_byname< _CharT, _OutIter >::iter_type`

Public typedefs.

Reimplemented from [std::time\\_put< \\_CharT, \\_OutIter >](#).

Definition at line 811 of file locale\_facets\_nonio.h.

## 5.656.3 Member Function Documentation

### 5.656.3.1 `template<typename _CharT , typename _OutIter > _OutIter std::time_put< _CharT, _OutIter >::do_put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const [protected, virtual, inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

[put\(\)](#) for more details.

#### Parameters

*s* The stream to write to.

*io* Source of locale.

*fill* char\_type to use for padding.

*tm* Struct tm with date and time info to format.

*format* Format char.

*mod* Optional modifier char.

**Returns**

Iterator after writing.

Definition at line 1175 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::_ctype_abstract_base<_CharT>::widen()`.

Referenced by `std::time_put<_CharT, _OutIter>::put()`.

```
5.656.3.2 template<typename _CharT, typename _OutIter > _OutIter
std::time_put<_CharT, _OutIter >::put ( iter_type __s, ios_base &
__io, char_type __fill, const tm * __tm, const _CharT * __beg,
const _CharT * __end ) const [inherited]
```

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format string. The format string is interpreted as by `strftime()`.

**Parameters**

*s* The stream to write to.

*io* Source of locale.

*fill* `char_type` to use for padding.

*tm* Struct `tm` with date and time info to format.

*beg* Start of format string.

*end* End of format string.

**Returns**

Iterator after writing.

Definition at line 1140 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::_ctype_abstract_base<_CharT>::narrow()`.

```
5.656.3.3 template<typename _CharT, typename _OutIter > iter_type
std::time_put<_CharT, _OutIter >::put ( iter_type __s, ios_base
& __io, char_type __fill, const tm * __tm, char __format, char
__mod = 0 ) const [inline, inherited]
```

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

### Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* `char_type` to use for padding.
- tm* Struct `tm` with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

### Returns

Iterator after writing.

Definition at line 770 of file `locale_facets_nonio.h`.

References `std::time_put<_CharT, _OutIter >::do_put()`.

## 5.656.4 Member Data Documentation

### 5.656.4.1 `template<typename _CharT, typename _OutIter > locale::id` `std::time_put<_CharT, _OutIter >::id` [`static`, `inherited`]

Numpunct facet id.

Definition at line 721 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.657 `std::timed_mutex` Class Reference

[timed\\_mutex](#)

### Public Types

- `typedef __native_type * native_handle_type`

## Public Member Functions

- **timed\_mutex** (const [timed\\_mutex](#) &)
- void **lock** ()
- native\_handle\_type **native\_handle** ()
- [timed\\_mutex](#) & **operator=** (const [timed\\_mutex](#) &)
- bool **try\_lock** ()
- template<class \_Rep, class \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<class \_Clock, class \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### 5.657.1 Detailed Description

[timed\\_mutex](#)

Definition at line 167 of file mutex.

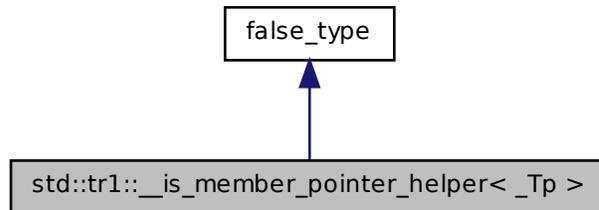
The documentation for this class was generated from the following file:

- [mutex](#)

### 5.658 `std::tr1::__is_member_pointer_helper< _Tp >` Struct Template Reference

`is_member_pointer`

Inheritance diagram for `std::tr1::__is_member_pointer_helper<_Tp>`:



### Public Types

- typedef `integral_constant<_Tp, __v>` **type**
- typedef `_Tp` **value\_type**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.658.1 Detailed Description

`template<typename _Tp> struct std::tr1::__is_member_pointer_helper<_Tp>`

`is_member_pointer`

Definition at line 295 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.659 `std::tr1::add_const<_Tp>` Struct Template Reference

[add\\_const](#)

## Public Types

- typedef `_Tp` const **type**

### 5.659.1 Detailed Description

`template<typename _Tp> struct std::tr1::add_const< _Tp >`

[add\\_const](#)

Definition at line 426 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.660 `std::tr1::add_cv< _Tp >` Struct Template Reference

[add\\_cv](#)

## Public Types

- typedef `add_const< typename add_volatile< _Tp >::type >::type` **type**

### 5.660.1 Detailed Description

`template<typename _Tp> struct std::tr1::add_cv< _Tp >`

[add\\_cv](#)

Definition at line 436 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.661 `std::tr1::add_pointer< _Tp >` Struct Template Reference

[add\\_pointer](#)

## Public Types

- typedef [remove\\_reference< \\_Tp >::type](#) \* **type**

### 5.661.1 Detailed Description

`template<typename _Tp> struct std::tr1::add_pointer< _Tp >`

[add\\_pointer](#)

Definition at line 491 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.662 `std::tr1::add_volatile< _Tp >` Struct Template Reference

[add\\_volatile](#)

## Public Types

- typedef `_Tp volatile` **type**

### 5.662.1 Detailed Description

`template<typename _Tp> struct std::tr1::add_volatile< _Tp >`

[add\\_volatile](#)

Definition at line 431 of file `tr1_impl/type_traits`.

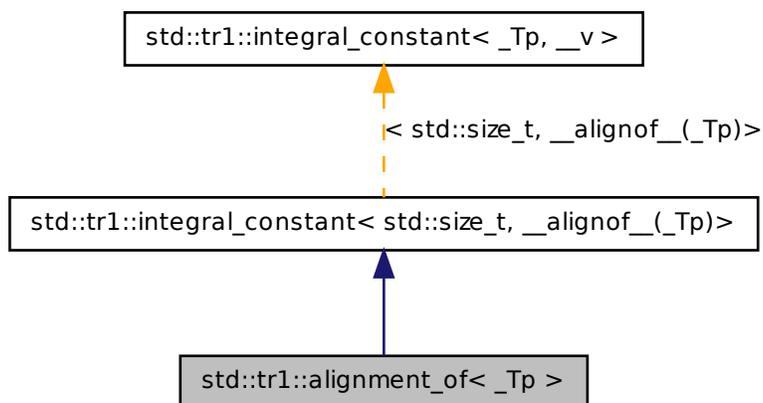
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.663 `std::tr1::alignment_of< _Tp >` Struct Template Reference

[alignment\\_of](#)

Inheritance diagram for `std::tr1::alignment_of< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

## Static Public Attributes

- static const std::size\_t **value**

### 5.663.1 Detailed Description

`template<typename _Tp> struct std::tr1::alignment_of< _Tp >`

[alignment\\_of](#)

Definition at line 350 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.664 `std::tr1::array< _Tp, _Nm >` Struct Template Reference

A standard container for storing a fixed size sequence of elements.

### Public Types

- typedef `const value_type * const_iterator`
- typedef `const _Tp * const_pointer`
- typedef `const value_type & const_reference`
- typedef `std::reverse_iterator< const_iterator > const_reverse_iterator`
- typedef `std::ptrdiff_t difference_type`
- typedef `value_type * iterator`
- typedef `_Tp * pointer`
- typedef `value_type & reference`
- typedef `std::reverse_iterator< iterator > reverse_iterator`
- typedef `std::size_t size_type`
- typedef `_Tp value_type`

### Public Member Functions

- reference `at (size_type __n)`
- const\_reference `at (size_type __n) const`
- reference `back ()`
- const\_reference `back () const`
- iterator `begin ()`
- const\_iterator `begin () const`
- const\_iterator `cbegin () const`
- const\_iterator `end () const`
- `const_reverse_iterator` `crbegin () const`
- `const_reverse_iterator` `crend () const`
- `const _Tp *` `data () const`
- `_Tp *` `data ()`
- bool `empty () const`
- iterator `end ()`
- const\_iterator `end () const`
- void `fill (const value_type &__u)`
- reference `front ()`
- const\_reference `front () const`
- size\_type `max_size () const`
- reference `operator[] (size_type __n)`

- `const_reference` **operator**[ ] (size\_type \_\_n) const
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () const
- size\_type **size** () const
- void **swap** (array &\_\_other)

### Public Attributes

- value\_type **\_M\_instance** [\_Nm?\_Nm:1]

#### 5.664.1 Detailed Description

`template<typename _Tp, std::size_t _Nm> struct std::tr1::array< _Tp, _Nm >`

A standard container for storing a fixed size sequence of elements. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

#### Parameters

***Tp*** Type of element. Required to be a complete type.

***N*** Number of elements.

Definition at line 49 of file `tr1_impl/array`.

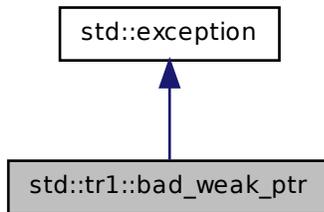
The documentation for this struct was generated from the following file:

- [tr1\\_impl/array](#)

#### 5.665 `std::tr1::bad_weak_ptr` Class Reference

Exception possibly thrown by [shared\\_ptr](#).

Inheritance diagram for `std::tr1::bad_weak_ptr`:



## Public Member Functions

- virtual `char const * what () const throw ()`

### 5.665.1 Detailed Description

Exception possibly thrown by [shared\\_ptr](#).

Definition at line 58 of file `boost_sp_counted_base.h`.

### 5.665.2 Member Function Documentation

#### 5.665.2.1 `virtual char const* std::tr1::bad_weak_ptr::what ( ) const throw ()` [inline, virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Definition at line 62 of file `boost_sp_counted_base.h`.

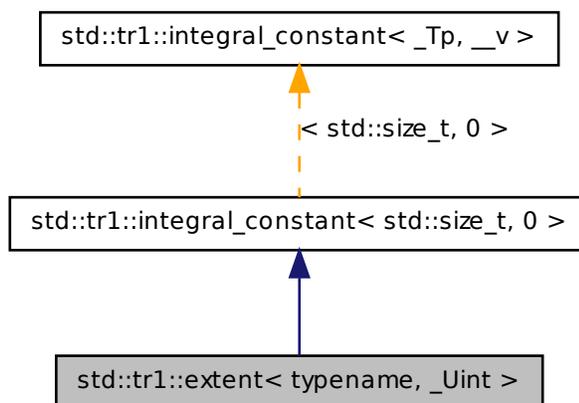
The documentation for this class was generated from the following file:

- [boost\\_sp\\_counted\\_base.h](#)

## 5.666 `std::tr1::extent< typename, _Uint >` Struct Template Reference

extent

Inheritance diagram for `std::tr1::extent< typename, _Uint >`:



### Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value\_type**

### Static Public Attributes

- static const `std::size_t` **value**

### 5.666.1 Detailed Description

```
template<typename, unsigned _Uint = 0> struct std::tr1::extent< typename, _Uint >
```

extent

## 5.667 `std::tr1::has_virtual_destructor<_Tp>` Struct Template Reference 2971

Definition at line 368 of file `tr1_impl/type_traits`.

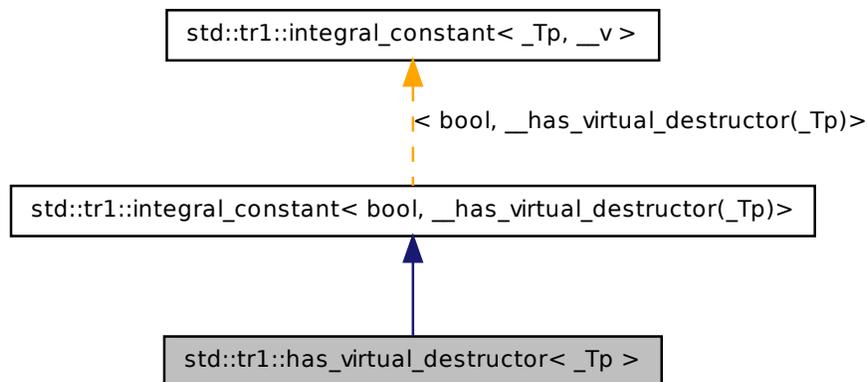
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.667 `std::tr1::has_virtual_destructor<_Tp>` Struct Template Reference

[has\\_virtual\\_destructor](#)

Inheritance diagram for `std::tr1::has_virtual_destructor<_Tp>`:



### Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

### 5.667.1 Detailed Description

```
template<typename _Tp> struct std::tr1::has_virtual_destructor< _Tp >
```

[has\\_virtual\\_destructor](#)

Definition at line 344 of file tr1\_impl/type\_traits.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.668 `std::tr1::integral_constant< _Tp, __v >` Struct Template Reference

[integral\\_constant](#)



### 5.668.1 Detailed Description

```
template<typename _Tp, _Tp __v> struct std::tr1::integral_constant< _Tp, __v >
```

[integral\\_constant](#)

Definition at line 67 of file tr1\_impl/type\_traits.

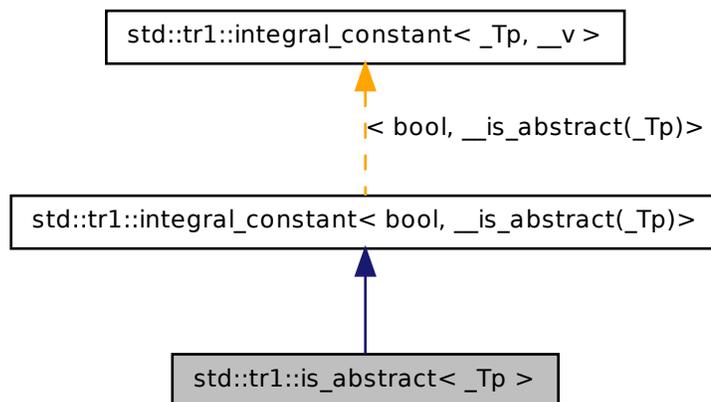
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.669 std::tr1::is\_abstract< \_Tp > Struct Template Reference

[is\\_abstract](#)

Inheritance diagram for std::tr1::is\_abstract< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**

- typedef bool `value_type`

## Static Public Attributes

- static const bool `value`

### 5.669.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_abstract< _Tp >`

[is\\_abstract](#)

Definition at line 338 of file `tr1_impl/type_traits`.

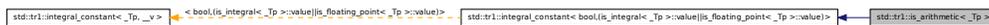
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.670 `std::tr1::is_arithmetic< _Tp >` Struct Template Reference

[is\\_arithmetic](#)

Inheritance diagram for `std::tr1::is_arithmetic< _Tp >`:



## Public Types

- typedef `integral_constant< bool, __v > type`
- typedef bool `value_type`

## Static Public Attributes

- static const bool `value`

### 5.670.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_arithmetic< _Tp >`

[is\\_arithmetic](#)

Definition at line 255 of file `tr1_impl/type_traits`.

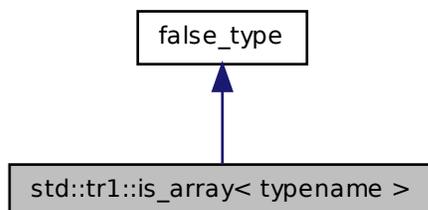
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.671 `std::tr1::is_array< typename >` Struct Template Reference

[is\\_array](#)

Inheritance diagram for `std::tr1::is_array< typename >`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Static Public Attributes

- static const \_Tp **value**

### 5.671.1 Detailed Description

```
template<typename> struct std::tr1::is_array< typename >
```

[is\\_array](#)

Definition at line 147 of file tr1\_impl/type\_traits.

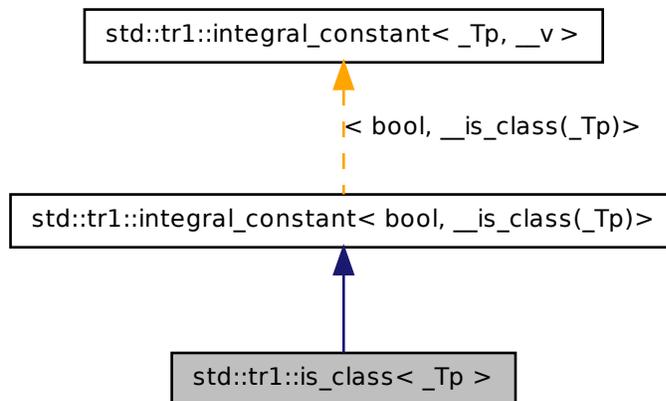
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.672 std::tr1::is\_class< \_Tp > Struct Template Reference

[is\\_class](#)

Inheritance diagram for std::tr1::is\_class< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Static Public Attributes

- static const bool **value**

### 5.672.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_class< _Tp >`

[is\\_class](#)

Definition at line 218 of file `tr1_impl/type_traits`.

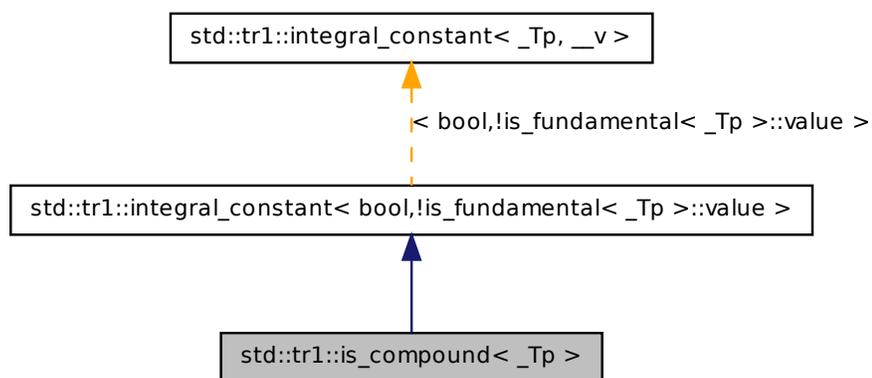
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.673 `std::tr1::is_compound< _Tp >` Struct Template Reference

[is\\_compound](#)

Inheritance diagram for `std::tr1::is_compound< _Tp >`:



## Public Types

- typedef `integral_constant< bool, __v >` `type`
- typedef `bool` `value_type`

## Static Public Attributes

- static const `bool` `value`

### 5.673.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_compound< _Tp >`

[is\\_compound](#)

Definition at line 290 of file `tr1_impl/type_traits`.

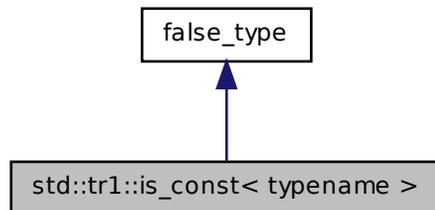
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.674 `std::tr1::is_const< typename >` Struct Template Reference

[is\\_const](#)

Inheritance diagram for `std::tr1::is_const< typename >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Static Public Attributes

- static const \_Tp **value**

### 5.674.1 Detailed Description

`template<typename> struct std::tr1::is_const< typename >`

[is\\_const](#)

Definition at line 308 of file `tr1_impl/type_traits`.

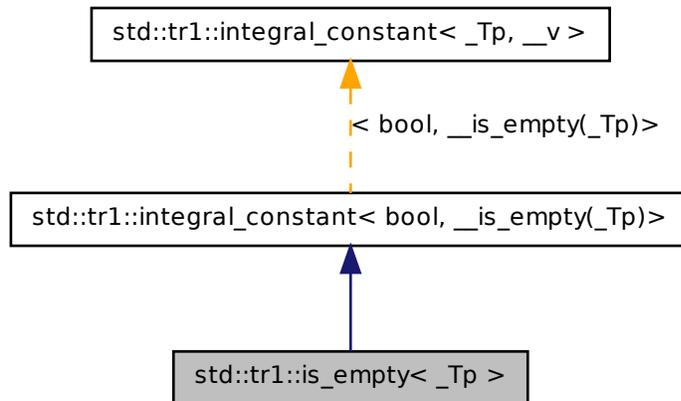
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.675 `std::tr1::is_empty< _Tp >` Struct Template Reference

[is\\_empty](#)

Inheritance diagram for std::tr1::is\_empty< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Static Public Attributes

- static const `bool` **value**

### 5.675.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_empty< _Tp >
```

[is\\_empty](#)

Definition at line 326 of file `tr1_impl/type_traits`.

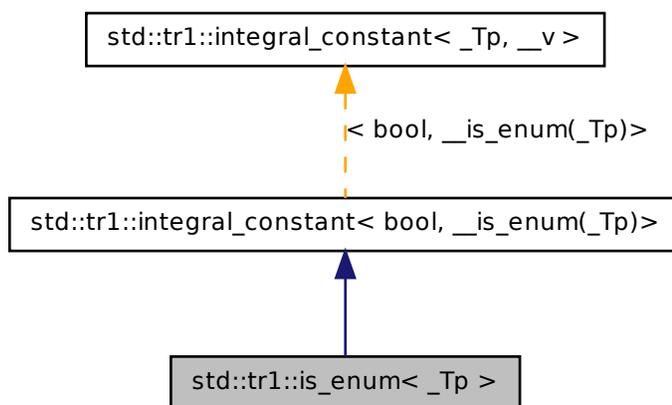
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.676 `std::tr1::is_enum< _Tp >` Struct Template Reference

[is\\_enum](#)

Inheritance diagram for `std::tr1::is_enum< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.676.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_enum< _Tp >
```

[is\\_enum](#)

Definition at line 206 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.677 `std::tr1::is_floating_point< _Tp >` Struct Template Reference

[is\\_floating\\_point](#)

Inheritance diagram for `std::tr1::is_floating_point< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.677.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_floating_point< _Tp >`

[is\\_floating\\_point](#)

Definition at line 140 of file `tr1_impl/type_traits`.

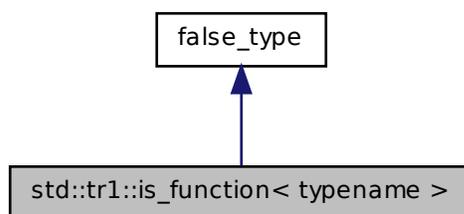
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.678 `std::tr1::is_function< typename >` Struct Template Reference

[is\\_function](#)

Inheritance diagram for `std::tr1::is_function< typename >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

### Static Public Attributes

- static const `_Tp` **value**

### 5.678.1 Detailed Description

`template<typename> struct std::tr1::is_function< typename >`

[is\\_function](#)

Definition at line 224 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.679 `std::tr1::is_fundamental< _Tp >` Struct Template Reference

### [is\\_fundamental](#)

Inheritance diagram for `std::tr1::is_fundamental< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.679.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_fundamental< _Tp >`

### [is\\_fundamental](#)

Definition at line 262 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.680 `std::tr1::is_integral< _Tp >` Struct Template Reference

### [is\\_integral](#)

Inheritance diagram for `std::tr1::is_integral< _Tp >`:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Static Public Attributes

- static const `bool` **value**

### 5.680.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_integral< _Tp >`

[is\\_integral](#)

Definition at line 126 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.681 `std::tr1::is_member_function_pointer< _Tp >` Struct Template Reference

[is\\_member\\_function\\_pointer](#)

Inheritance diagram for `std::tr1::is_member_function_pointer< _Tp >`:



## Public Types

- typedef `integral_constant< bool, __v >` **type**

## 5.682 `std::tr1::is_member_object_pointer< _Tp >` Struct Template Reference 2987

- typedef `bool value_type`

### Static Public Attributes

- static const `bool value`

### 5.681.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_member_function_pointer< _Tp >`

[is\\_member\\_function\\_pointer](#)

Definition at line 199 of file `tr1_impl/type_traits`.

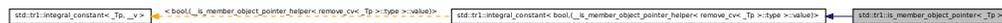
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.682 `std::tr1::is_member_object_pointer< _Tp >` Struct Template Reference

[is\\_member\\_object\\_pointer](#)

Inheritance diagram for `std::tr1::is_member_object_pointer< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v > type`
- typedef `bool value_type`

### Static Public Attributes

- static const `bool value`

### 5.682.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_member_object_pointer< _Tp >`

[is\\_member\\_object\\_pointer](#)

Definition at line 186 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

### 5.683 `std::tr1::is_object< _Tp >` Struct Template Reference

[is\\_object](#)

Inheritance diagram for `std::tr1::is_object< _Tp >`:



#### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

#### Static Public Attributes

- static const `bool` **value**

### 5.683.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_object< _Tp >`

[is\\_object](#)

Definition at line 269 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.684 std::tr1::is\_pointer< \_Tp > Struct Template Reference

[is\\_pointer](#)

Inheritance diagram for std::tr1::is\_pointer< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Static Public Attributes

- static const bool **value**

#### 5.684.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_pointer< _Tp >
```

[is\\_pointer](#)

Definition at line 165 of file tr1\_impl/type\_traits.

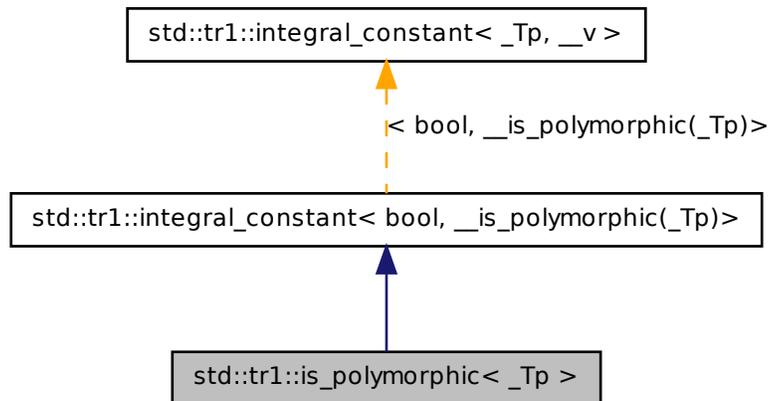
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.685 std::tr1::is\_polymorphic< \_Tp > Struct Template Reference

[is\\_polymorphic](#)

Inheritance diagram for `std::tr1::is_polymorphic<_Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Static Public Attributes

- static const bool **value**

### 5.685.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_polymorphic<_Tp >
```

[is\\_polymorphic](#)

Definition at line 332 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

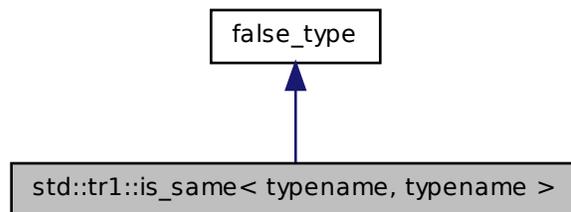
- [tr1\\_impl/type\\_traits](#)

## 5.686 std::tr1::is\_same< typename, typename > Struct Template Reference 2991

### **5.686 std::tr1::is\_same< typename, typename > Struct Template Reference**

[is\\_same](#)

Inheritance diagram for std::tr1::is\_same< typename, typename >:



#### **Public Types**

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### **Static Public Attributes**

- static const \_Tp **value**

#### **5.686.1 Detailed Description**

```
template<typename, typename> struct std::tr1::is_same< typename, typename >
```

[is\\_same](#)

Definition at line 389 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.687 `std::tr1::is_scalar< _Tp >` Struct Template Reference

[is\\_scalar](#)

Inheritance diagram for `std::tr1::is_scalar< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.687.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_scalar< _Tp >`

[is\\_scalar](#)

Definition at line 281 of file `tr1_impl/type_traits`.

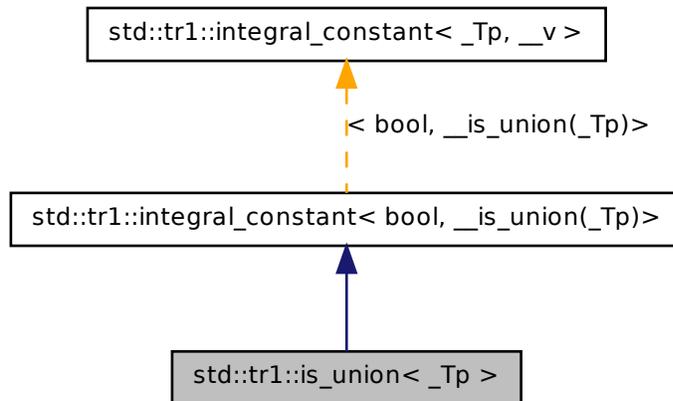
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.688 `std::tr1::is_union< _Tp >` Struct Template Reference

[is\\_union](#)

Inheritance diagram for std::tr1::is\_union< \_Tp >:



## Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Static Public Attributes

- static const `bool` **value**

### 5.688.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_union< _Tp >
```

[is\\_union](#)

Definition at line 212 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.689 `std::tr1::is_void< _Tp >` Struct Template Reference

[is\\_void](#)

Inheritance diagram for `std::tr1::is_void< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Static Public Attributes

- static const `bool` **value**

#### 5.689.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_void< _Tp >`

[is\\_void](#)

Definition at line 96 of file `tr1_impl/type_traits`.

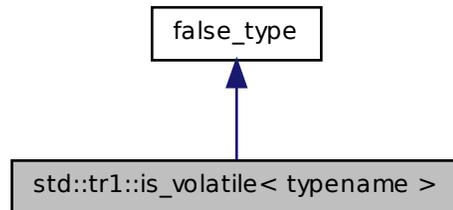
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.690 `std::tr1::is_volatile< typename >` Struct Template Reference

[is\\_volatile](#)

Inheritance diagram for `std::tr1::is_volatile< typename >`:



### Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

### Static Public Attributes

- static const `_Tp` **value**

#### 5.690.1 Detailed Description

`template<typename> struct std::tr1::is_volatile< typename >`

[is\\_volatile](#)

Definition at line 317 of file `tr1_impl/type_traits`.

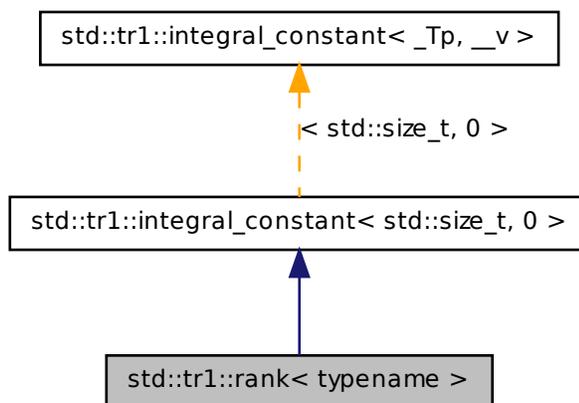
The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.691 `std::tr1::rank< typename >` Struct Template Reference

`rank`

Inheritance diagram for `std::tr1::rank< typename >`:



## Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value\_type**

## Static Public Attributes

- static const `std::size_t` **value**

### 5.691.1 Detailed Description

`template<typename> struct std::tr1::rank< typename >`

rank

Definition at line 355 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.692 `std::tr1::remove_all_extents< _Tp >` Struct Template Reference

[remove\\_all\\_extents](#)

### Public Types

- `typedef _Tp type`

#### 5.692.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_all_extents< _Tp >
```

[remove\\_all\\_extents](#)

Definition at line 459 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.693 `std::tr1::remove_const< _Tp >` Struct Template Reference

[remove\\_const](#)

### Public Types

- `typedef _Tp type`

#### 5.693.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_const< _Tp >
```

[remove\\_const](#)

Definition at line 400 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.694 `std::tr1::remove_cv< _Tp >` Struct Template Reference

[remove\\_cv](#)

### Public Types

- typedef [remove\\_const](#)< typename [remove\\_volatile](#)< \_Tp >::type >::type **type**

#### 5.694.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_cv< _Tp >`

[remove\\_cv](#)

Definition at line 418 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.695 `std::tr1::remove_extent< _Tp >` Struct Template Reference

[remove\\_extent](#)

### Public Types

- typedef `_Tp` **type**

#### 5.695.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_extent< _Tp >`

[remove\\_extent](#)

Definition at line 446 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.696 `std::tr1::remove_pointer< _Tp >` Struct Template Reference

[remove\\_pointer](#)

Inherits `std::tr1::__remove_pointer_helper< _Tp, remove_cv< _Tp >::type >`.

### Public Types

- `typedef _Tp type`

#### 5.696.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_pointer< _Tp >
```

[remove\\_pointer](#)

Definition at line 482 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.697 `std::tr1::remove_volatile< _Tp >` Struct Template Reference

[remove\\_volatile](#)

### Public Types

- `typedef _Tp type`

#### 5.697.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_volatile< _Tp >
```

[remove\\_volatile](#)

Definition at line 409 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1\\_impl/type\\_traits](#)

## 5.698 `std::try_to_lock_t` Struct Reference

Try to acquire ownership of the mutex without blocking.

### 5.698.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

Definition at line 379 of file mutex.

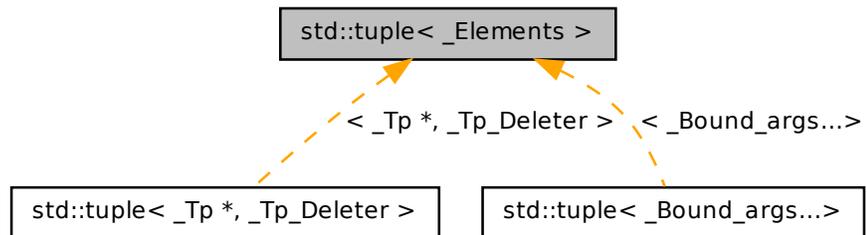
The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.699 `std::tuple< _Elements >` Class Template Reference

tuple

Inheritance diagram for `std::tuple< _Elements >`:



### Public Member Functions

- **tuple** (const `_Elements` &...\_\_elements)
- **tuple** (const [tuple](#) &\_\_in)
- `template<typename... _UElements>`  
**tuple** ([tuple](#)< `_UElements...`> &\_\_in)

- `tuple` (`tuple` &&\_\_in)
- `template<typename... _UElements>`  
`tuple` (`_UElements` &&...\_\_elements)
- `template<typename... _UElements>`  
`tuple` (`const tuple< _UElements...>` &\_\_in)
- `template<typename... _UElements>`  
`tuple` (`tuple< _UElements...>` &&\_\_in)
- `tuple` & `operator=` (`tuple` &&\_\_in)
- `template<typename... _UElements>`  
`tuple` & `operator=` (`const tuple< _UElements...>` &\_\_in)
- `template<typename... _UElements>`  
`tuple` & `operator=` (`tuple< _UElements...>` &&\_\_in)
- `tuple` & `operator=` (`const tuple` &\_\_in)
- `void swap` (`tuple` &\_\_in)

### 5.699.1 Detailed Description

`template<typename... _Elements> class std::tuple< _Elements >`

`tuple`

Definition at line 224 of file `tuple`.

The documentation for this class was generated from the following file:

- `tuple`

## 5.700 `std::tuple< _T1, _T2 >` Class Template Reference

`tuple` (2-element), with construction and assignment from a pair.

Inherits `_Tuple_impl< 0, _T1, _T2 >`.

### Public Member Functions

- `tuple` (`const _T1` &\_\_a1, `const _T2` &\_\_a2)
- `tuple` (`const tuple` &\_\_in)
- `template<typename _U1, typename _U2 >`  
`tuple` (`const pair< _U1, _U2 >` &\_\_in)
- `template<typename _U1, typename _U2 >`  
`tuple` (`pair< _U1, _U2 >` &&\_\_in)
- `tuple` (`tuple` &&\_\_in)

- `template<typename _U1, typename _U2 >`  
`tuple (_U1 &&_a1, _U2 &&_a2)`
- `template<typename _U1, typename _U2 >`  
`tuple (const tuple< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const tuple< _U1, _U2 > &__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (pair< _U1, _U2 > &&__in)`
- `void swap (tuple &__in)`

### 5.700.1 Detailed Description

`template<typename _T1, typename _T2> class std::tuple< _T1, _T2 >`

tuple (2-element), with construction and assignment from a pair.

Definition at line 307 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

### 5.701 std::tuple\_element< 0, tuple< \_Head, \_Tail...> > Struct Template Reference

#### Public Types

- `typedef _Head type`

#### 5.701.1 Detailed Description

`template<typename _Head, typename... _Tail> struct std::tuple_element< 0, tuple< _Head, _Tail...> >`

Basis case for `tuple_element`: The first element is the one we're seeking.

## 5.702 `std::tuple_element< __i, tuple< _Head, _Tail...> >` Struct Template Reference 3003

---

Definition at line 421 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.702 `std::tuple_element< __i, tuple< _Head, _Tail...> >` Struct Template Reference

Inherits `tuple_element< __i-1, tuple< _Tail...> >`.

### 5.702.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail> struct std::tuple_
element< __i, tuple< _Head, _Tail...> >
```

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 414 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.703 `std::tuple_size< tuple< _Elements...> >` Struct Template Reference

```
class tuple_size
```

### Static Public Attributes

- static const `std::size_t` value

### 5.703.1 Detailed Description

```
template<typename... _Elements> struct std::tuple_size< tuple< _Elements...>
>
```

```
class tuple_size
```

Definition at line 432 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.704 std::type\_info Class Reference

Part of RTTI.

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

### Public Member Functions

- virtual `~type_info ()`
- virtual `bool __do_catch (const type_info * __thr_type, void ** __thr_obj, unsigned __outer) const`
- virtual `bool __do_upcast (const __cxxabiv1::__class_type_info * __target, void ** __obj_ptr) const`
- virtual `bool __is_function_p () const`
- virtual `bool __is_pointer_p () const`
- `bool before (const type_info & __arg) const`
- `const char * name () const`
- `bool operator!= (const type_info & __arg) const`
- `bool operator== (const type_info & __arg) const`

### Protected Member Functions

- `type_info (const char * __n)`

### Protected Attributes

- `const char * __name`

#### 5.704.1 Detailed Description

Part of RTTI. The `type_info` class describes type information generated by an implementation.

Definition at line 87 of file typeinfo.

## 5.704.2 Constructor & Destructor Documentation

### 5.704.2.1 virtual std::type\_info::~~type\_info ( ) [virtual]

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated [type\\_info](#) structures in the new-abi.

## 5.704.3 Member Function Documentation

### 5.704.3.1 const char\* std::type\_info::name ( ) const [inline]

Returns an *implementation-defined* byte string; this is not portable between compilers!

Definition at line 98 of file typeinfo.

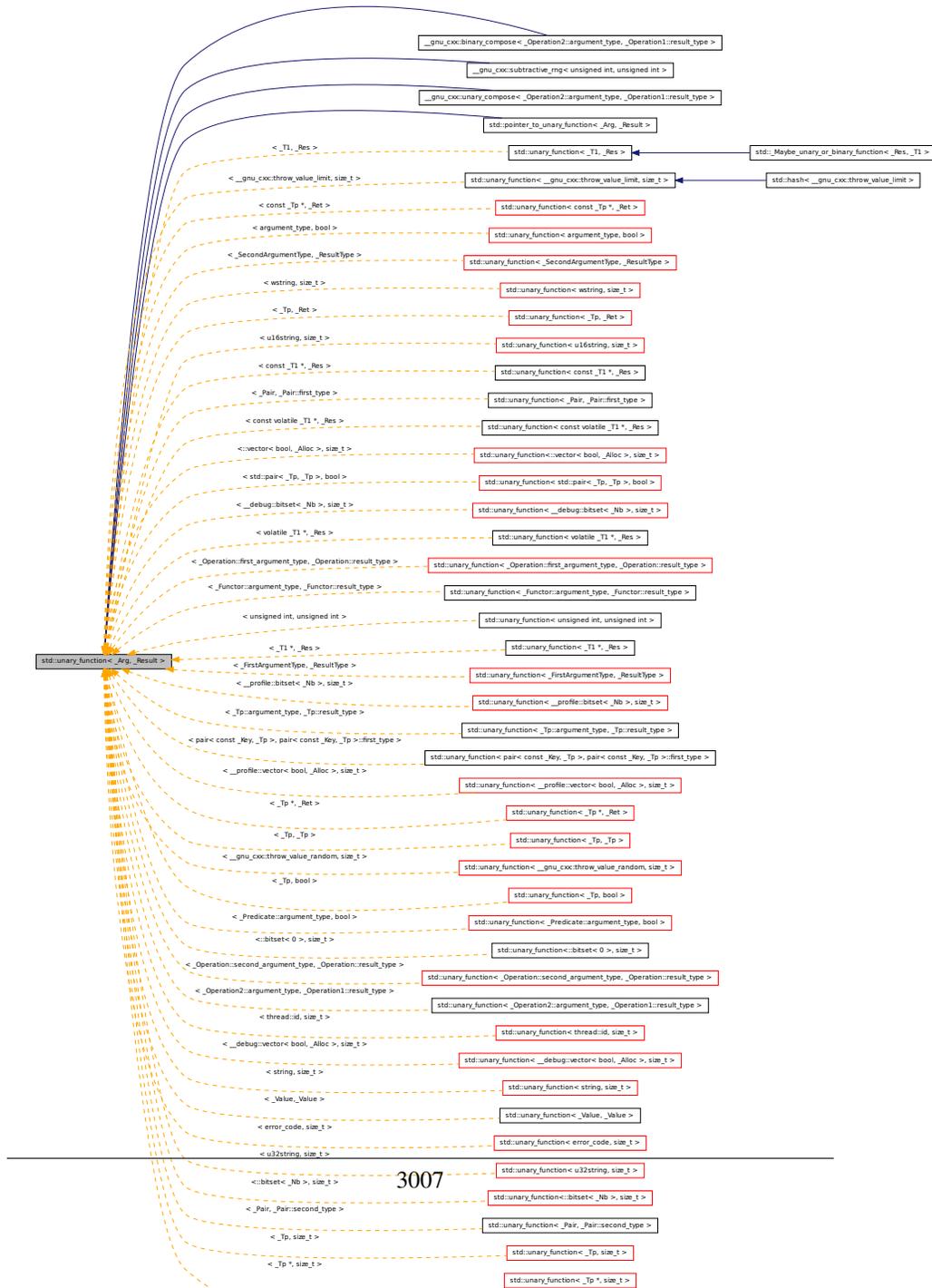
The documentation for this class was generated from the following file:

- [typeinfo](#)



## 5.705 std::unary\_function< \_Arg, \_Result > Struct Template Reference

Inheritance diagram for std::unary\_function< \_Arg, \_Result >:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### 5.705.1 Detailed Description

**template**<typename `_Arg`, typename `_Result`> struct `std::unary_function`< `_Arg`, `_Result` >

This is one of the [functor base classes](#).

Definition at line 100 of file `stl_function.h`.

### 5.705.2 Member Typedef Documentation

**5.705.2.1** **template**<typename `_Arg`, typename `_Result`> typedef `_Arg` `std::unary_function`< `_Arg`, `_Result` >::`argument_type`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

**5.705.2.2** **template**<typename `_Arg`, typename `_Result`> typedef `_Result` `std::unary_function`< `_Arg`, `_Result` >::`result_type`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

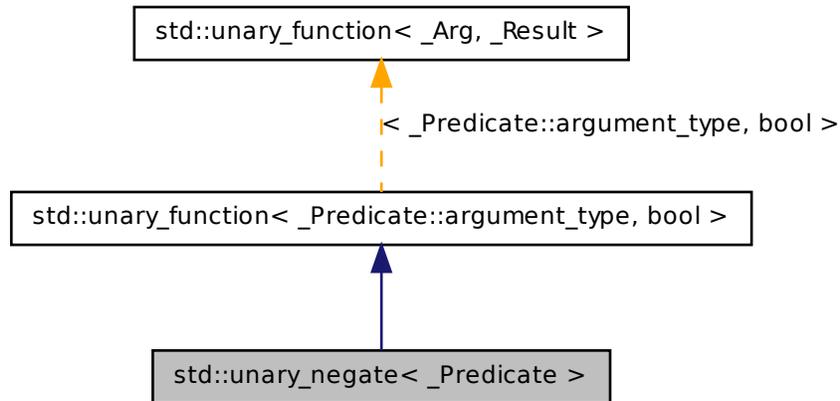
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.706 `std::unary_negate`< `_Predicate` > Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::unary\_negate<\_Predicate >:



## Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- **unary\_negate** (`const _Predicate &__x`)
- **operator()** (`const typename _Predicate::argument_type &__x`) `const`

## Protected Attributes

- `_Predicate _M_pred`

### 5.706.1 Detailed Description

`template<typename _Predicate> class std::unary_negate<_Predicate >`

One of the [negation functors](#).

Definition at line 346 of file `stl_function.h`.

## 5.706.2 Member Typedef Documentation

### 5.706.2.1 `typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the `///` argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

### 5.706.2.2 `typedef bool std::unary_function< _Predicate::argument_type , bool >::result_type [inherited]`

`result_type` is the return type

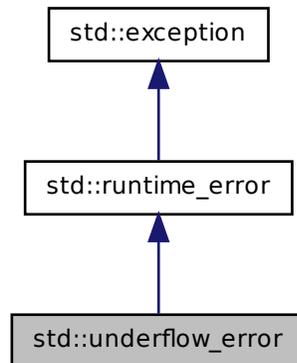
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.707 `std::underflow_error` Class Reference

Inheritance diagram for `std::underflow_error`:



### Public Member Functions

- `underflow_error` (const [string](#) &\_\_arg)
- virtual const char \* `what` () const throw ()

#### 5.707.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 140 of file `stdexcept`.

#### 5.707.2 Member Function Documentation

##### 5.707.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [[virtual](#), [inherited](#)]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.708 `std::uniform_int_distribution< _IntType >` Class Template Reference

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- [uniform\\_int\\_distribution](#) (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType >::max()`)
- [uniform\\_int\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type param](#) () const
- void [reset](#) ()

### Public Attributes

- [param\\_type \\_M\\_param](#)

## 5.708 std::uniform\_int\_distribution< \_IntType > Class Template Reference 3013

### 5.708.1 Detailed Description

```
template<typename _IntType = int> class std::uniform_int_distribution< _-  
_IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 1628 of file random.h.

### 5.708.2 Member Typedef Documentation

**5.708.2.1** `template<typename _IntType = int> typedef _IntType  
std::uniform_int_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 1635 of file random.h.

### 5.708.3 Constructor & Destructor Documentation

**5.708.3.1** `template<typename _IntType = int> std::uniform_int_distribution<  
_IntType >::uniform_int_distribution ( _IntType __a = 0, _IntType  
__b = std::numeric_limits<_IntType>::max() ) [inline,  
explicit]`

Constructs a uniform distribution object.

Definition at line 1671 of file random.h.

### 5.708.4 Member Function Documentation

**5.708.4.1** `template<typename _IntType = int> result_type  
std::uniform_int_distribution< _IntType >::max ( ) const  
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1723 of file random.h.

**5.708.4.2** `template<typename _IntType = int> result_type  
std::uniform_int_distribution< _IntType >::min ( ) const  
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1716 of file random.h.

**5.708.4.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::uniform_int_distribution< _IntType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 1731 of file random.h.

References `std::uniform_int_distribution< _IntType >::operator()`, and `std::uniform_int_distribution< _IntType >::param()`.

Referenced by `std::uniform_int_distribution< _IntType >::operator()`.

**5.708.4.4** `template<typename _IntType = int> void std::uniform_int_  
distribution< _IntType >::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

#### Parameters

*\_\_param* The new parameter set of the distribution.

Definition at line 1709 of file random.h.

**5.708.4.5** `template<typename _IntType = int> param_type  
std::uniform_int_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 1701 of file random.h.

Referenced by `std::uniform_int_distribution< _IntType >::operator()`, `std::operator==( )`, and `std::operator>>()`.

## 5.709 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference 3015

**5.708.4.6** `template<typename _IntType = int> void  
std::uniform_int_distribution<_IntType>::reset( ) [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1687 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.709 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference

### Public Types

- typedef `uniform_int_distribution<_IntType>` `distribution_type`

### Public Member Functions

- `param_type` (`_IntType __a=0`, `_IntType __b=std::numeric_limits<_IntType>::max()`)
- `result_type a` () const
- `result_type b` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.709.1 Detailed Description

`template<typename _IntType = int> struct std::uniform_int_distribution<_IntType>::param_type`

Parameter type.

Definition at line 1637 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.710 `std::uniform_real_distribution< _RealType >` Class Template Reference

Uniform continuous distribution for random numbers.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- [uniform\\_real\\_distribution](#) (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- [uniform\\_real\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type param](#) () const
- void [reset](#) ()

### 5.710.1 Detailed Description

```
template<typename _RealType = double> class std::uniform_real_distribution<
_RealType >
```

Uniform continuous distribution for random numbers. A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1800 of file random.h.

## 5.710 std::uniform\_real\_distribution<\_RealType> Class Template Reference

### 5.710.2 Member Typedef Documentation

**5.710.2.1** `template<typename _RealType = double> typedef _RealType  
std::uniform_real_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 1807 of file random.h.

### 5.710.3 Constructor & Destructor Documentation

**5.710.3.1** `template<typename _RealType = double> std::uniform_real_  
distribution<_RealType>::uniform_real_distribution ( _RealType  
__a = _RealType(0), _RealType __b = _RealType(1) )  
[inline, explicit]`

Constructs a [uniform\\_real\\_distribution](#) object.

#### Parameters

`__min` [IN] The lower bound of the distribution.

`__max` [IN] The upper bound of the distribution.

Definition at line 1846 of file random.h.

### 5.710.4 Member Function Documentation

**5.710.4.1** `template<typename _RealType = double> result_type  
std::uniform_real_distribution<_RealType>::max ( ) const  
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1898 of file random.h.

**5.710.4.2** `template<typename _RealType = double> result_type  
std::uniform_real_distribution<_RealType>::min ( ) const  
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1891 of file random.h.

**5.710.4.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::uniform_real_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & _urng ) [inline]`

Generating functions.

Definition at line 1906 of file random.h.

References `std::uniform_real_distribution< _RealType >::operator()`, and `std::uniform_real_distribution< _RealType >::param()`.

Referenced by `std::uniform_real_distribution< _RealType >::operator()`.

**5.710.4.4** `template<typename _RealType = double> void  
std::uniform_real_distribution< _RealType >::param ( const  
param_type & _param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

*\_param* The new parameter set of the distribution.

Definition at line 1884 of file random.h.

**5.710.4.5** `template<typename _RealType = double> param_type  
std::uniform_real_distribution< _RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 1876 of file random.h.

Referenced by `std::uniform_real_distribution< _RealType >::operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.710.4.6** `template<typename _RealType = double> void  
std::uniform_real_distribution< _RealType >::reset ( )  
[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1862 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.711 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `uniform_real_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `result_type a` () const
- `result_type b` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.711.1 Detailed Description

```
template<typename _RealType = double> struct std::uniform_real_
distribution<_RealType>::param_type
```

Parameter type.

Definition at line 1809 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.712 `std::unique_lock<_Mutex>` Class Template Reference

[unique\\_lock](#)

### Public Types

- typedef `_Mutex` `mutex_type`

## Public Member Functions

- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (const [unique\\_lock](#) &)
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- **unique\_lock** ([unique\\_lock](#) &&\_\_u)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#))
- template<typename \_Clock, typename \_Duration >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Rep, typename \_Period >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- void **lock** ()
- mutex\_type \* **mutex** () const
- **operator bool** () const
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u)
- bool **owns\_lock** () const
- mutex\_type \* **release** ()
- void **swap** ([unique\\_lock](#) &\_\_u)
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### 5.712.1 Detailed Description

template<typename \_Mutex> class [std::unique\\_lock](#)< \_Mutex >

[unique\\_lock](#)

Definition at line 416 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.713 `std::unique_ptr< _Tp, _Tp_Deleter >` Class Template Reference

20.7.12.2 `unique_ptr` for single objects.

### Public Types

- typedef `_Tp_Deleter` **deleter\_type**
- typedef `_Tp` **element\_type**
- typedef `_Tp *` **pointer**

### Public Member Functions

- `unique_ptr` (pointer `__p`)
- `unique_ptr` (pointer `__p`, typename `std::remove_reference< deleter_type >::type` `&&__d`)
- `unique_ptr` (`unique_ptr` `&&__u`)
- `unique_ptr` (const `unique_ptr` `&`)
- `unique_ptr` (pointer `__p`, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type` `__d`)
- template<typename `_Up`, typename `_Up_Deleter` >  
`unique_ptr` (`unique_ptr`< `_Up`, `_Up_Deleter` > `&&__u`)
- pointer `get` () const
- const `deleter_type` `& get_deleter` () const
- `deleter_type` `& get_deleter` ()
- **operator bool** () const
- `std::add_lvalue_reference< element_type >::type` **operator\*** () const
- pointer **operator->** () const
- `unique_ptr` `& operator=` (const `unique_ptr` `&`)
- `unique_ptr` `& operator=` (`unique_ptr` `&&__u`)
- `unique_ptr` `& operator=` (`__unspecified_pointer_type`)
- template<typename `_Up`, typename `_Up_Deleter` >  
`unique_ptr` `& operator=` (`unique_ptr`< `_Up`, `_Up_Deleter` > `&&__u`)
- pointer **release** ()
- void **reset** (pointer `__p=pointer()`)
- void **swap** (`unique_ptr` `&__u`)

### 5.713.1 Detailed Description

```
template<typename _Tp, typename _Tp_Deleter = default_delete<_Tp>> class
std::unique_ptr<_Tp, _Tp_Deleter >
```

20.7.12.2 [unique\\_ptr](#) for single objects.

Definition at line 81 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 5.714 `std::unique_ptr<_Tp[], _Tp_Deleter >` Class Template Reference

20.7.12.3 [unique\\_ptr](#) for array objects with a runtime length

### Public Types

- typedef `_Tp_Deleter` `deleter_type`
- typedef `_Tp` `element_type`
- typedef `_Tp *` `pointer`

### Public Member Functions

- `unique_ptr` (`pointer __p`)
- `template<typename _Up >`  
`unique_ptr` (`_Up *`, `typename std::remove_reference< deleter_type >::type`  
`&&`, `typename std::enable_if< std::is_convertible< _Up *, pointer >::value`  
`>::type *=0`)
- `unique_ptr` (`pointer __p`, `typename std::remove_reference< deleter_type`  
`>::type &&__d`)
- `unique_ptr` (`unique_ptr &&__u`)
- `unique_ptr` (`const unique_ptr &`)
- `template<typename _Up >`  
`unique_ptr` (`_Up *`, `typename std::conditional< std::is_reference< deleter_type`  
`>::value, deleter_type, const deleter_type & >::type`, `typename std::enable_if<`  
`std::is_convertible< _Up *, pointer >::value >::type *=0`)
- `unique_ptr` (`pointer __p`, `typename std::conditional< std::is_reference<`  
`deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- `template<typename _Up, typename _Up_Deleter >`  
`unique_ptr` (`unique_ptr< _Up, _Up_Deleter > &&__u`)

## 5.715 `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference 3023

---

- `template<typename _Up >`  
`unique_ptr` (`_Up *`, `typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type !=0`)
- pointer `get` () `const`
- deleter\_type & `get_deleter` ()
- `const` deleter\_type & `get_deleter` () `const`
- `operator bool` () `const`
- `unique_ptr` & `operator=` (`unique_ptr` &&`_u`)
- `template<typename _Up, typename _Up_Deleter >`  
`unique_ptr` & `operator=` (`unique_ptr`< `_Up`, `_Up_Deleter` > &&`_u`)
- `unique_ptr` & `operator=` (`__unspecified_pointer_type`)
- `unique_ptr` & `operator=` (`const unique_ptr` &)
- `std::add_lvalue_reference< element_type >::type operator[]` (`size_t __i`) `const`
  
- pointer `release` ()
- `template<typename _Up >`  
void `reset` (`_Up`)
- void `reset` (`pointer __p=pointer()`)
- void `swap` (`unique_ptr` &`_u`)

### 5.714.1 Detailed Description

`template<typename _Tp, typename _Tp_Deleter> class std::unique_ptr< _Tp[], _Tp_Deleter >`

20.7.12.3 `unique_ptr` for array objects with a runtime length

Definition at line 219 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 5.715 `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >`.

## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code >` **const\_iterator**
- typedef `__detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code >` **const\_local\_iterator**
- typedef `_Allocator::const_pointer` **const\_pointer**
- typedef `_Allocator::const_reference` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `__detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Key` **key\_type**
- typedef `__detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code >` **local\_iterator**
- typedef `_Allocator::pointer` **pointer**
- typedef `_Allocator::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `\_Base::value\_type` **value\_type**

## Public Member Functions

- **unordered\_map** (`size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_map** (`_InputIterator __f, _InputIterator __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- **unordered\_map** (`initializer\_list< value_type > __l, size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- **unordered\_map** (`unordered\_map &&__x`)
- `void` **\_\_rehash\_policy** (`const _RehashPolicy &`)
- `const _RehashPolicy &` **\_\_rehash\_policy** (`() const`)
- `_Value_allocator_type` **M\_get\_Value\_allocator** (`() const`)
- `iterator` **begin** (`()`)
- `const_iterator` **begin** (`() const`)
- `local_iterator` **begin** (`size_type __n`)
- `const_local_iterator` **begin** (`size_type __n`) `const`
- `size_type` **bucket** (`const key_type &__k`) `const`
- `size_type` **bucket\_count** (`() const`)

## 5.715 `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference 3025

---

- `size_type` **bucket\_size** (`size_type __n`) `const`
- `const_iterator` **cbegin** () `const`
- `const_local_iterator` **cbegin** (`size_type __n`) `const`
- `const_iterator` **cend** () `const`
- `const_local_iterator` **cend** (`size_type`) `const`
- `void` **clear** ()
- `size_type` **count** (`const key_type &__k`) `const`
- `bool` **empty** () `const`
- `iterator` **end** ()
- `const_iterator` **end** () `const`
- `local_iterator` **end** (`size_type`)
- `const_local_iterator` **end** (`size_type`) `const`
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type &__k`) `const`
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type &__k`)
- `iterator` **erase** (`const_iterator, const_iterator`)
- `iterator` **erase** (`const_iterator`)
- `size_type` **erase** (`const key_type &`)
- `iterator` **find** (`const key_type &__k`)
- `const_iterator` **find** (`const key_type &__k`) `const`
- `allocator_type` **get\_allocator** () `const`
- `iterator` **insert** (`const_iterator, const value_type &__v`)
- `template<typename _InputIterator >`  
`void` **insert** (`_InputIterator __first, _InputIterator __last`)
- `_Insert_Return_Type` **insert** (`const value_type &__v`)
- `void` **insert** (`initializer_list< value_type > __l`)
- `key_equal` **key\_eq** () `const`
- `float` **load\_factor** () `const`
- `size_type` **max\_bucket\_count** () `const`
- `size_type` **max\_size** () `const`
- `unordered_map` & **operator=** (`initializer_list< value_type > __l`)
- `unordered_map` & **operator=** (`unordered_map &&__x`)
- `void` **rehash** (`size_type __n`)
- `size_type` **size** () `const`
- `void` **swap** (`_Hashtable &`)

### Friends

- `struct` `__detail::Map_base`

### 5.715.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

*Key* Type of key objects.

*Tp* Type of mapped objects.

*Hash* Hashing function object type, defaults to `hash<Value>`.

*Pred* Predicate function object type, defaults to `equal_to<Value>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 220 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

### 5.716 `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

#### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `__detail::Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Key` **key\_type**
- typedef `__detail::Node_iterator< value_type, __constant_iterators, __cache_hash_code >` **local\_iterator**
- typedef `_Allocator::pointer` **pointer**
- typedef `_Allocator::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multimap** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_multimap** (`_InputIterator __f`, `_InputIterator __l`, `typename _Base::size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multimap** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multimap** (`unordered_multimap &&__x`)
- `void __rehash_policy` (`const _RehashPolicy &`)
- `const _RehashPolicy & __rehash_policy` () `const`
- `_Value_allocator_type _M_get_Value_allocator` () `const`
- `iterator begin` ()
- `const_iterator begin` () `const`
- `local_iterator begin` (`size_type __n`)
- `const_local_iterator begin` (`size_type __n`) `const`
- `size_type bucket` (`const key_type &__k`) `const`
- `size_type bucket_count` () `const`
- `size_type bucket_size` (`size_type __n`) `const`
- `const_iterator cbegin` () `const`
- `const_local_iterator cbegin` (`size_type __n`) `const`
- `const_iterator cend` () `const`
- `const_local_iterator cend` (`size_type`) `const`
- `void clear` ()
- `size_type count` (`const key_type &__k`) `const`
- `bool empty` () `const`
- `iterator end` ()

- const\_iterator **end** () const
- local\_iterator **end** (size\_type)
- const\_local\_iterator **end** (size\_type) const
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_k) const
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_k)
- iterator **erase** (const\_iterator, const\_iterator)
- iterator **erase** (const\_iterator)
- size\_type **erase** (const key\_type &)
- iterator **find** (const key\_type &\_\_k)
- const\_iterator **find** (const key\_type &\_\_k) const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [\\_Insert\\_Return\\_Type](#) **insert** (const value\_type &\_\_v)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- key\_equal **key\_eq** () const
- float **load\_factor** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- [unordered\\_multimap](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_multimap](#) & **operator=** ([unordered\\_multimap](#) &&\_\_x)
- void **rehash** (size\_type \_\_n)
- size\_type **size** () const
- void **swap** (\_Hashtable &)

## Friends

- struct [\\_\\_detail::\\_Map\\_base](#)

### 5.716.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

*Key* Type of key objects.

## 5.717 `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference 3029

---

*Tp* Type of mapped objects.

*Hash* Hashing function object type, defaults to `hash<Value>`.

*Pred* Predicate function object type, defaults to `equal_to<Value>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 301 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.717 `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_multiset< _Value, _Hash, _Pred, _Alloc >`.

### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, typename \_Base::size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_x)
- void **\_\_rehash\_policy** (const \_RehashPolicy &)
- const \_RehashPolicy & **\_\_rehash\_policy** () const
- \_Value\_allocator\_type **\_M\_get\_Value\_allocator** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- local\_iterator **begin** (size\_type \_\_n)
- const\_local\_iterator **begin** (size\_type \_\_n) const
- size\_type **bucket** (const key\_type &\_\_k) const
- size\_type **bucket\_count** () const
- size\_type **bucket\_size** (size\_type \_\_n) const
- const\_iterator **cbegin** () const
- const\_local\_iterator **cbegin** (size\_type \_\_n) const
- const\_iterator **cend** () const
- const\_local\_iterator **cend** (size\_type) const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_k) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- local\_iterator **end** (size\_type)
- const\_local\_iterator **end** (size\_type) const
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_k) const
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_k)
- iterator **erase** (const\_iterator, const\_iterator)
- iterator **erase** (const\_iterator)
- size\_type **erase** (const key\_type &)
- iterator **find** (const key\_type &\_\_k)
- const\_iterator **find** (const key\_type &\_\_k) const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const\_iterator, const value\_type &\_\_v)

## 5.717 `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >` Class Template Reference 3031

---

- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `_Insert_Return_Type insert (const value_type &__v)`
- `void insert (initializer_list< value_type > __l)`
- `key_equal key_eq () const`
- `float load_factor () const`
- `size_type max_bucket_count () const`
- `size_type max_size () const`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `unordered_multiset & operator= (unordered_multiset &&__x)`
- `void rehash (size_type __n)`
- `size_type size () const`
- `void swap (_Hashtable &)`

### Friends

- `struct __detail::_Map_base`

### 5.717.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

*Value* Type of key objects.

*Hash* Hashing function object type, defaults to `hash<Value>`.

*Pred* Predicate function object type, defaults to `equal_to<Value>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

Definition at line 291 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.718 `std::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Inherits `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`.

### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__detail::Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code >` **const\_iterator**
- typedef `__detail::Node_const_iterator< value_type, __constant_iterators, __cache_hash_code >` **const\_local\_iterator**
- typedef `_Allocator::const_pointer` **const\_pointer**
- typedef `_Allocator::const_reference` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `__detail::Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Key` **key\_type**
- typedef `__detail::Node_iterator< value_type, __constant_iterators, __cache_hash_code >` **local\_iterator**
- typedef `_Allocator::pointer` **pointer**
- typedef `_Allocator::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_set** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**unordered\_set** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_set** (`initializer_list< value_type > __l`, `size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_set** (`unordered_set &&__x`)
- `void __rehash_policy` (`const _RehashPolicy &`)

- `const _RehashPolicy & __rehash_policy () const`
- `_Value_allocator_type _M_get_Value_allocator () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `size_type bucket (const key_type &__k) const`
- `size_type bucket_count () const`
- `size_type bucket_size (size_type __n) const`
- `const_iterator cbegin () const`
- `const_local_iterator cbegin (size_type __n) const`
- `const_iterator cend () const`
- `const_local_iterator cend (size_type) const`
- `void clear ()`
- `size_type count (const key_type &__k) const`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `local_iterator end (size_type)`
- `const_local_iterator end (size_type) const`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__k)`
- `iterator erase (const_iterator, const_iterator)`
- `iterator erase (const_iterator)`
- `size_type erase (const key_type &)`
- `iterator find (const key_type &__k)`
- `const_iterator find (const key_type &__k) const`
- `allocator_type get_allocator () const`
- `iterator insert (const_iterator, const value_type &__v)`
- `template<typename _InputIterator >  
void insert (_InputIterator __first, _InputIterator __last)`
- `_Insert_Return_Type insert (const value_type &__v)`
- `void insert (initializer_list< value_type > __l)`
- `key_equal key_eq () const`
- `float load_factor () const`
- `size_type max_bucket_count () const`
- `size_type max_size () const`
- `unordered_set & operator= (initializer_list< value_type > __l)`
- `unordered_set & operator= (unordered_set &&__x)`
- `void rehash (size_type __n)`
- `size_type size () const`
- `void swap (_Hashtable &)`

## Friends

- struct `__detail::_Map_base`

### 5.718.1 Detailed Description

```
template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_ -
to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set<
_Value, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

*Value* Type of key objects.

*Hash* Hashing function object type, defaults to `hash<Value>`.

*Pred* Predicate function object type, defaults to `equal_to<Value>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

Definition at line 213 of file `unordered_set.h`.

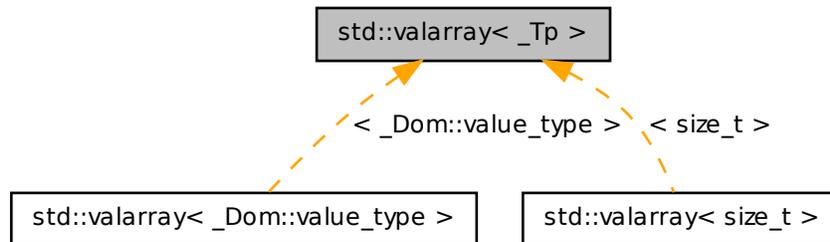
The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

### 5.719 `std::valarray<_Tp>` Class Template Reference

Smart array designed to support numeric processing.

Inheritance diagram for `std::valarray<_Tp>`:



## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- [valarray](#) ()
- [valarray](#) (size\_t)
- [valarray](#) (const `_Tp` \*\_\_restrict\_\_, size\_t)
- [valarray](#) (const [mask\\_array](#)< `_Tp` > &)
- [valarray](#) (const [indirect\\_array](#)< `_Tp` > &)
- template<typename `_Tp`>  
**valarray** (const `_Tp` \*\_\_restrict\_\_ \_\_p, size\_t \_\_n)
- [valarray](#) (const [valarray](#) &)
- [valarray](#) ([initializer\\_list](#)< `_Tp` >)
- template<class `_Dom` >  
**valarray** (const `_Expr`< `_Dom`, `_Tp` > &\_\_e)
- [valarray](#) (const `_Tp` &, size\_t)
- [valarray](#) (const [slice\\_array](#)< `_Tp` > &)
- [valarray](#) (const [gslice\\_array](#)< `_Tp` > &)
- `_Expr`< `_ValFunClos`< `_ValArray`, `_Tp` >, `_Tp` > [apply](#) (`_Tp` func(`_Tp`)) const
- `_Expr`< `_RefFunClos`< `_ValArray`, `_Tp` >, `_Tp` > [apply](#) (`_Tp` func(const `_Tp` &)) const
- [valarray](#)< `_Tp` > [cshift](#) (int) const
- `_Tp` [max](#) () const

- `_Tp min () const`
- `_UnaryOp< __logical_not >::_Rt operator! () const`
- `valarray< _Tp > & operator%= (const _Tp &)`
- `valarray< _Tp > & operator%= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator%= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator&= (const _Tp &)`
- `valarray< _Tp > & operator&= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator&= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator*%= (const _Tp &)`
- `valarray< _Tp > & operator*%= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator*%= (const _Expr< _Dom, _Tp > &)`
- `_UnaryOp< __unary_plus >::_Rt operator+ () const`
- `valarray< _Tp > & operator+= (const _Tp &)`
- `valarray< _Tp > & operator+= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator+= (const _Expr< _Dom, _Tp > &)`
- `_UnaryOp< __negate >::_Rt operator- () const`
- `valarray< _Tp > & operator-= (const _Tp &)`
- `valarray< _Tp > & operator-= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator-= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator/= (const _Tp &)`
- `valarray< _Tp > & operator/= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator/= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator<<= (const _Tp &)`
- `valarray< _Tp > & operator<<= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & operator<<= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator= (const _Tp &)`
- `valarray< _Tp > & operator= (const gslice_array< _Tp > &)`
- `valarray< _Tp > & operator= (const slice_array< _Tp > &)`
- `valarray< _Tp > & operator= (const indirect_array< _Tp > &)`
- `valarray & operator= (initializer_list< _Tp >)`
- `template<class _Dom >`  
`valarray< _Tp > & operator= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator= (const mask_array< _Tp > &)`
- `valarray< _Tp > & operator= (const valarray< _Tp > &)`
- `valarray< _Tp > & operator>>= (const valarray< _Tp > &)`

- template<class \_Dom >  
valarray< \_Tp > & operator>>= (const \_Expr< \_Dom, \_Tp > &)
- valarray< \_Tp > & operator>>= (const \_Tp &)
- gslice\_array< \_Tp > operator[] (const gslice &)
- \_Tp & operator[] (size\_t)
- \_Expr< \_SClos< \_ValArray, \_Tp >, \_Tp > operator[] (slice) const
- valarray< \_Tp > operator[] (const valarray< bool > &) const
- \_Expr< \_IClos< \_ValArray, \_Tp >, \_Tp > operator[] (const valarray< size\_t > &) const
- slice\_array< \_Tp > operator[] (slice)
- \_Expr< \_GClos< \_ValArray, \_Tp >, \_Tp > operator[] (const gslice &) const
- indirect\_array< \_Tp > operator[] (const valarray< size\_t > &)
- const \_Tp & operator[] (size\_t) const
- mask\_array< \_Tp > operator[] (const valarray< bool > &)
- valarray< \_Tp > & operator^= (const valarray< \_Tp > &)
- template<class \_Dom >  
valarray< \_Tp > & operator^= (const \_Expr< \_Dom, \_Tp > &)
- valarray< \_Tp > & operator^= (const \_Tp &)
- template<class \_Dom >  
valarray< \_Tp > & operator|= (const \_Expr< \_Dom, \_Tp > &)
- valarray< \_Tp > & operator|= (const \_Tp &)
- valarray< \_Tp > & operator|= (const valarray< \_Tp > &)
- \_UnaryOp< \_\_bitwise\_not >::\_Rt operator~ () const
- void resize (size\_t \_\_size, \_Tp \_\_c=\_Tp())
- valarray< \_Tp > shift (int) const
- size\_t size () const
- \_Tp sum () const

## Friends

- class \_Array< \_Tp >

### 5.719.1 Detailed Description

template<class \_Tp> class std::valarray< \_Tp >

Smart array designed to support numeric processing. A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

**Parameters**

*Tp* Type of object in the array.

Definition at line 112 of file valarray.

**5.719.2 Constructor & Destructor Documentation****5.719.2.1 `template<class _Tp> std::valarray< _Tp >::valarray ( const _Tp * __restrict __, size_t )`**

Construct an array initialized to the first *n* elements of *t*.

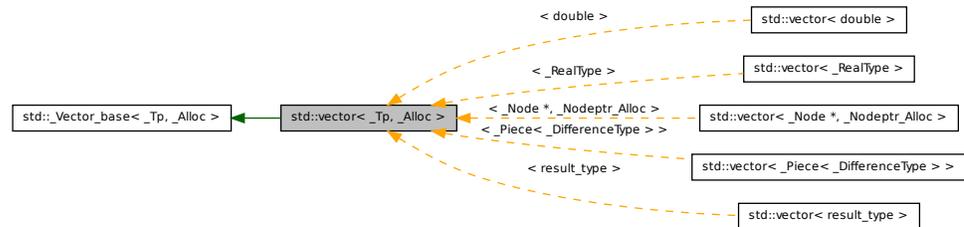
The documentation for this class was generated from the following file:

- [valarray](#)

**5.720 `std::vector< _Tp, _Alloc >` Class Template Reference**

A standard container which offers fixed time access to individual elements in any order.

Inheritance diagram for `std::vector< _Tp, _Alloc >`:

**Public Types**

- `typedef _Alloc allocator_type`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, vector > const_iterator`
- `typedef _Tp_alloc_type::const_pointer const_pointer`
- `typedef _Tp_alloc_type::const_reference const_reference`

- typedef `std::reverse_iterator`< const\_iterator > **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef `__gnu_cxx::__normal_iterator`< pointer, `vector` > **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator`< iterator > **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `vector` ()
- `vector` (const allocator\_type &\_\_a)
- `vector` (const `vector` &\_\_x)
- `vector` (`vector` &&\_\_x)
- `vector` (size\_type \_\_n, const value\_type &\_\_value=value\_type(), const allocator\_type &\_\_a=allocator\_type())
- `vector` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
`vector` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- `~vector` ()
- void `assign` (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (`initializer_list`< value\_type > \_\_l)
- reference `at` (size\_type \_\_n)
- const\_reference `at` (size\_type \_\_n) const
- reference `back` ()
- const\_reference `back` () const
- const\_iterator `begin` () const
- iterator `begin` ()
- size\_type `capacity` () const
- const\_iterator `cbegin` () const
- const\_iterator `cend` () const
- void `clear` ()
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- pointer `data` ()
- const\_pointer `data` () const
- template<typename... \_Args >  
iterator `emplace` (iterator \_\_position, \_Args &&...\_\_args)

- `template<typename... _Args>`  
void **emplace\_back** (\_Args &&... \_\_args)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- reference **front** ()
- const\_reference **front** () const
- void **insert** (iterator \_\_position, initializer\_list< value\_type > \_\_l)
- iterator **insert** (iterator \_\_position, value\_type &&\_\_x)
- void **insert** (iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- `template<typename _InputIterator >`  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (iterator \_\_position, const value\_type &\_\_x)
- size\_type **max\_size** () const
- **vector** & **operator=** (const **vector** &\_\_x)
- **vector** & **operator=** (**vector** &&\_\_x)
- **vector** & **operator=** (initializer\_list< value\_type > \_\_l)
- reference **operator[ ]** (size\_type \_\_n)
- const\_reference **operator[ ]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (value\_type &&\_\_x)
- void **push\_back** (const value\_type &\_\_x)
- **reverse\_iterator** **rbegin** ()
- **const\_reverse\_iterator** **rbegin** () const
- **const\_reverse\_iterator** **rend** () const
- **reverse\_iterator** **rend** ()
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, value\_type \_\_x=value\_type())
- void **shrink\_to\_fit** ()
- size\_type **size** () const
- void **swap** (**vector** &\_\_x)

## Protected Member Functions

- `_Tp_alloc_type::pointer` **\_M\_allocate** (size\_t \_\_n)
- `template<typename _ForwardIterator >`  
pointer **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- `template<typename _InputIterator >`  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, `std::input_iterator_tag`)

- `template<typename _ForwardIterator >`  
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __-`  
`false_type)`
- `template<typename _Integer >`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `void _M_deallocate (typename _Tp_alloc_type::pointer __p, size_t __n)`
- `void _M_erase_at_end (pointer __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __-`  
`false_type)`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __value, __true_type)`
- `template<typename... _Args>`  
`void _M_insert_aux (iterator __position, _Args &&...__args)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_`  
`type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator`  
`__last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last,`  
`std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_insert (iterator __pos, _ForwardIterator __first, _-`  
`ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __-`  
`last, std::input_iterator_tag)`
- `allocator_type get_allocator () const`

## Protected Attributes

- `_Vector_impl_M_impl`

### 5.720.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::vector<_Tp, _Alloc >
```

A standard container which offers fixed time access to individual elements in any order. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 170 of file `stl_vector.h`.

### 5.720.2 Constructor & Destructor Documentation

```
5.720.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector ( ) [inline]
```

Default constructor creates no elements.

Definition at line 207 of file `stl_vector.h`.

```
5.720.2.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector ( const allocator_type & __a )
[inline, explicit]
```

Creates a vector with no elements.

#### Parameters

- a* An allocator object.

Definition at line 215 of file `stl_vector.h`.

```
5.720.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector ( size_type __n, const value_type
& __value = value_type(), const allocator_type & __a =
allocator_type() ) [inline, explicit]
```

Creates a vector with copies of an exemplar element.

#### Parameters

- n* The number of elements to initially create.
- value* An element to copy.
- a* An allocator.

This constructor fills the vector with *n* copies of *value*.

Definition at line 227 of file stl\_vector.h.

```
5.720.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector ( const vector<_Tp, _Alloc> &
__x ) [inline]
```

Vector copy constructor.

#### Parameters

- x* A vector of identical element and allocator types.

The newly-created vector uses a copy of the allocation object used by *x*. All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied.

Definition at line 241 of file stl\_vector.h.

```
5.720.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector ( vector<_Tp, _Alloc> && __x
) [inline]
```

Vector move constructor.

#### Parameters

- x* A vector of identical element and allocator types.

The newly-created vector contains the exact contents of *x*. The contents of *x* are a valid, but unspecified vector.

Definition at line 257 of file stl\_vector.h.

**5.720.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc >::vector ( initializer_list< value_type  
> __l, const allocator_type & __a = allocator_type() )  
[inline]`

Builds a vector from an initializer list.

#### Parameters

*l* An [initializer\\_list](#).  
*a* An allocator.

Create a vector consisting of copies of the elements in the [initializer\\_list](#) *l*.

This will call the element type's copy constructor N times (where N is *l.size()*) and do no memory reallocation.

Definition at line 271 of file `stl_vector.h`.

**5.720.2.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > std::vector<_Tp, _Alloc  
>::vector ( _InputIterator __first, _InputIterator __last, const  
allocator_type & __a = allocator_type() ) [inline]`

Builds a vector from a range.

#### Parameters

*first* An input iterator.  
*last* An input iterator.  
*a* An allocator.

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 297 of file `stl_vector.h`.

**5.720.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc >::~~vector ( ) [inline]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 312 of file `stl_vector.h`.

### 5.720.3 Member Function Documentation

**5.720.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`template<typename _ForwardIterator > pointer std::vector<_Tp,`  
`_Alloc >::M_allocate_and_copy ( size_type __n, _ForwardIterator`  
`__first, _ForwardIterator __last ) [inline, protected]`

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Definition at line 964 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc >::operator=()`.

**5.720.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`void std::vector<_Tp, _Alloc >::M_range_check ( size_type __n )`  
`const [inline, protected]`

Safety check used only from `at()`.

Definition at line 639 of file `stl_vector.h`.

**5.720.3.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`template<typename _InputIterator > void std::vector<_Tp,`  
`_Alloc >::assign ( _InputIterator __first, _InputIterator __last )`  
`[inline]`

Assigns a range to a vector.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a vector with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 392 of file `stl_vector.h`.

**5.720.3.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc >::assign ( size_type __n, const  
value_type & __val ) [inline]`

Assigns a given value to a vector.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a vector with *n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 375 of file `stl_vector.h`.

**5.720.3.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc >::assign ( initializer_list< value_type  
> __l ) [inline]`

Assigns an initializer list to a vector.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 412 of file `stl_vector.h`.

Referenced by `std::vector< result_type >::assign()`.

**5.720.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::vector<_Tp, _Alloc >::at ( size_type __n )  
[inline]`

Provides access to the data contained in the vector.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read/write reference to data.

**Exceptions**

*[std::out\\_of\\_range](#)* If  $n$  is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [out\\_of\\_range](#) if the check fails.

Definition at line 658 of file `stl_vector.h`.

```
5.720.3.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::at ( size_type __n )
const [inline]
```

Provides access to the data contained in the vector.

**Parameters**

$n$  The index of the element for which data should be accessed.

**Returns**

Read-only (constant) reference to data.

**Exceptions**

*[std::out\\_of\\_range](#)* If  $n$  is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [out\\_of\\_range](#) if the check fails.

Definition at line 676 of file `stl_vector.h`.

```
5.720.3.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::back ( ) const
[inline]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 711 of file `stl_vector.h`.

```
5.720.3.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the vector.

Definition at line 703 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::max()`, and `std::piecewise_constant_distribution<_RealType>::max()`.

**5.720.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector< _Tp, _Alloc >::begin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 435 of file `stl_vector.h`.

**5.720.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::vector< _Tp, _Alloc >::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 426 of file `stl_vector.h`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _Alloc >::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::vector< result_type >::vector()`.

**5.720.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::vector< _Tp, _Alloc >::capacity ( ) const  
[inline]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 573 of file `stl_vector.h`.

Referenced by `std::vector< _Tp, _Alloc >::operator=()`.

**5.720.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector< _Tp, _Alloc >::cbegin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 499 of file `stl_vector.h`.

**5.720.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector<_Tp, _Alloc>::cend ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 508 of file `stl_vector.h`.

**5.720.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 954 of file `stl_vector.h`.

**5.720.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::crbegin ( )  
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 517 of file `stl_vector.h`.

**5.720.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 526 of file `stl_vector.h`.

**5.720.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
pointer std::vector<_Tp, _Alloc>::data ( ) [inline]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 722 of file `stl_vector.h`.

**5.720.3.19** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc  
>::emplace ( iterator __position, _Args &&... _args )`

Inserts an object in vector before specified iterator.

#### Parameters

*position* An iterator into the vector.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 272 of file vector.tcc.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

**5.720.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::vector< _Tp, _Alloc >::empty ( ) const [inline]`

Returns true if the vector is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Definition at line 582 of file stl\_vector.h.

**5.720.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector< _Tp, _Alloc >::end ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 453 of file stl\_vector.h.

**5.720.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::vector< _Tp, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 444 of file `stl_vector.h`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _Alloc >::erase()`, `std::vector< _Tp, _Alloc >::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::operator==(,)`, and `std::vector< result_type >::vector()`.

### 5.720.3.23 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase ( iterator __first, iterator __last )`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 146 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::end()`.

### 5.720.3.24 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase ( iterator __position )`

Remove element at given position.

#### Parameters

*position* Iterator pointing to element to be erased.

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using [std::list](#). The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 134 of file vector.tcc.

References `std::vector<_Tp, _Alloc >::end()`.

**5.720.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::vector<_Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 687 of file stl\_vector.h.

Referenced by `std::piecewise_linear_distribution<_RealType >::min()`, and `std::piecewise_constant_distribution<_RealType >::min()`.

**5.720.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::vector<_Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 695 of file stl\_vector.h.

**5.720.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc >::insert ( iterator __position,  
size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the vector.

#### Parameters

*position* An iterator into the vector.

*n* Number of elements to be inserted.

*x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 858 of file `stl_vector.h`.

**5.720.3.28** `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::insert ( iterator __position, const value_type & __x )`

Inserts given value into vector before specified iterator.

#### Parameters

*position* An iterator into the vector.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 107 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

**5.720.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::insert ( iterator __position, value_type && __x ) [inline]`

Inserts given rvalue into vector before specified iterator.

#### Parameters

*position* An iterator into the vector.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 823 of file `stl_vector.h`.

```
5.720.3.30 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc >::insert ( iterator __position,  
initializer_list<value_type > __l ) [inline]
```

Inserts an [initializer\\_list](#) into the vector.

#### Parameters

*position* An iterator into the vector.

*l* An [initializer\\_list](#).

This function will insert copies of the data in the [initializer\\_list](#) *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 840 of file `stl_vector.h`.

Referenced by `std::vector<result_type >::insert()`.

```
5.720.3.31 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > void std::vector<_Tp,  
_Alloc >::insert ( iterator __position, _InputIterator __first,  
_InputIterator __last ) [inline]
```

Inserts a range into the vector.

#### Parameters

*position* An iterator into the vector.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range [*first*,*last*) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 877 of file `stl_vector.h`.

```
5.720.3.32 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::vector<_Tp, _Alloc >::max_size ( ) const  
[inline]
```

Returns the [size\(\)](#) of the largest possible vector.

Definition at line 538 of file `stl_vector.h`.

**5.720.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
vector& std::vector<_Tp, _Alloc>::operator= ( vector<_Tp,  
_Alloc> && __x ) [inline]`

Vector move assignment operator.

#### Parameters

*x* A vector of identical element and allocator types.

The contents of *x* are moved into this vector (without copying). *x* is a valid, but unspecified vector.

Definition at line 336 of file stl\_vector.h.

**5.720.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
vector& std::vector<_Tp, _Alloc>::operator= ( initializer_list<  
value_type > __l ) [inline]`

Vector list assignment operator.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 357 of file stl\_vector.h.

**5.720.3.35** `template<typename _Tp, typename _Alloc > vector<_Tp, _Alloc  
> & std::vector<_Tp, _Alloc>::operator= ( const vector<_Tp,  
_Alloc> & __x )`

Vector assignment operator.

#### Parameters

*x* A vector of identical element and allocator types.

All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 157 of file vector.tcc.

References `std::_Destroy()`, `std::vector<_Tp, _Alloc >::_M_allocate_and_copy()`, `std::vector<_Tp, _Alloc >::begin()`, `std::vector<_Tp, _Alloc >::capacity()`, `std::vector<_Tp, _Alloc >::end()`, and `std::vector<_Tp, _Alloc >::size()`.

**5.720.3.36** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::vector<_Tp, _Alloc >::operator[] ( size_type  
__n ) const [inline]`

Subscript access to the data contained in the vector.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 633 of file `stl_vector.h`.

**5.720.3.37** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::vector<_Tp, _Alloc >::operator[] ( size_type __n )  
[inline]`

Subscript access to the data contained in the vector.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 618 of file `stl_vector.h`.

**5.720.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc >::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 772 of file `stl_vector.h`.

```
5.720.3.39 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::push_back ( const value_type &  
    _x ) [inline]
```

Add data to the end of the vector.

#### Parameters

**x** Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 741 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

```
5.720.3.40 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( )  
    [inline]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 462 of file `stl_vector.h`.

```
5.720.3.41 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) const  
    [inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 471 of file `stl_vector.h`.

**5.720.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 489 of file `stl_vector.h`.

**5.720.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 480 of file `stl_vector.h`.

**5.720.3.44** `template<typename _Tp, typename _Alloc > void std::vector<  
_Tp, _Alloc>::reserve ( size_type __n )`

Attempt to preallocate enough memory for specified number of elements.

#### Parameters

*n* Number of elements required.

#### Exceptions

*std::length\_error* If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 65 of file `vector.tcc`.

References `std::_Destroy()`.

**5.720.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::resize ( size_type __new_size,  
value_type __x = value_type() ) [inline]`

Resizes the vector to the specified number of elements.

**Parameters**

*new\_size* Number of elements the vector should contain.

*x* Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 553 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::multiway_merge_exact_splitting()`, and `__gnu_parallel::parallel_sort_mwms()`.

**5.720.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`void std::vector<_Tp, _Alloc>::shrink_to_fit ( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 564 of file `stl_vector.h`.

**5.720.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`size_type std::vector<_Tp, _Alloc>::size ( ) const [inline]`

Returns the number of elements in the vector.

Definition at line 533 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution<_IntType>::max()`, `std::vector<_Tp, _Alloc>::operator=()`, and `std::operator==( )`.

**5.720.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`void std::vector<_Tp, _Alloc>::swap ( vector<_Tp, _Alloc> &`  
`__x ) [inline]`

Swaps data with another vector.

**Parameters**

*x* A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 934 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 5.721 `std::vector< bool, _Alloc >` Class Template Reference

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.

Inherits `std::_Bvector_base< _Alloc >`.

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

### Public Member Functions

- **vector** (`size_type __n`, `const bool &__value=bool()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**vector** (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a=allocator_type()`)
- **vector** (`const vector &__x`)
- **vector** (`const allocator_type &__a`)
- **vector** (`vector &&__x`)
- **vector** (`initializer_list< bool > __l`, `const allocator_type &__a=allocator_type()`)

- void **assign** (size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator >  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (initializer\_list< bool > \_\_l)
- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- const\_iterator **begin** () const
- iterator **begin** ()
- size\_type **capacity** () const
- const\_iterator **cbegin** () const
- const\_iterator **cend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- void **data** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- void **flip** ()
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- void **insert** (iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- iterator **insert** (iterator \_\_position, const bool &\_\_x=bool())
- template<typename \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_p, initializer\_list< bool > \_\_l)
- size\_type **max\_size** () const
- vector & **operator=** (vector &&\_\_x)
- vector & **operator=** (const vector &\_\_x)
- vector & **operator=** (initializer\_list< bool > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (bool \_\_x)
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rbegin** ()
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const

- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- void **shrink\_to\_fit** ()
- size\_type **size** () const
- void **swap** (vector &\_\_x)

## Static Public Member Functions

- static void **swap** (reference \_\_x, reference \_\_y)

## Protected Types

- typedef \_Alloc::template rebind< \_Bit\_type >::other **\_Bit\_alloc\_type**

## Protected Member Functions

- \_Bit\_type \* **\_M\_allocate** (size\_t \_\_n)
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<class \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- void **\_M\_deallocate** ()
- void **\_M\_erase\_at\_end** (iterator \_\_pos)
- void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** ()
- const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const
- void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_InputIterator >  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)

- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_range (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_initialize_range (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `void _M_insert_aux (iterator __position, bool __x)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _ForwardIterator >`  
`void _M_insert_range (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`void _M_insert_range (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `void _M_range_check (size_type __n) const`

## Protected Attributes

- `_Bvector_impl_M_impl`

## Friends

- class `hash`

### 5.721.1 Detailed Description

`template<typename _Alloc> class std::vector< bool, _Alloc >`

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order. Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

#### See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 474 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 5.722 `std::weak_ptr<_Tp>` Class Template Reference

A smart pointer with weak semantics.

Inherits `__weak_ptr<_Tp>`.

### Public Member Functions

- `template<typename _Tp1 >`  
`weak_ptr` (const `weak_ptr<_Tp1 >` &\_\_r)
- `template<typename _Tp1 >`  
`weak_ptr` (const `shared_ptr<_Tp1 >` &\_\_r)
- `shared_ptr<_Tp >` `lock` () const
- `template<typename _Tp1 >`  
`weak_ptr` & `operator=` (const `shared_ptr<_Tp1 >` &\_\_r)
- `template<typename _Tp1 >`  
`weak_ptr` & `operator=` (const `weak_ptr<_Tp1 >` &\_\_r)

### 5.722.1 Detailed Description

`template<typename _Tp> class std::weak_ptr<_Tp >`

A smart pointer with weak semantics. With forwarding constructors and assignment operators.

Definition at line 321 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.723 `std::weibull_distribution<_RealType>` Class Template Reference

A `weibull_distribution` random number distribution.

### Classes

- struct `param_type`

### Public Types

- typedef `_RealType` `result_type`

### Public Member Functions

- `weibull_distribution` (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- `weibull_distribution` (const `param_type` &\_\_p)
- `_RealType a` () const
- `_RealType b` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator`> `result_type operator()` (`_UniformRandomNumberGenerator` &\_\_urng, const `param_type` &\_\_p)
- template<typename `_UniformRandomNumberGenerator`> `result_type operator()` (`_UniformRandomNumberGenerator` &\_\_urng)
- void `param` (const `param_type` &\_\_param)
- `param_type param` () const
- void `reset` ()

### 5.723.1 Detailed Description

```
template<typename _RealType = double> class std::weibull_distribution<_RealType>
```

A `weibull_distribution` random number distribution. The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4335 of file `random.h`.

## 5.723.2 Member Typedef Documentation

### 5.723.2.1 `template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4342 of file random.h.

## 5.723.3 Member Function Documentation

### 5.723.3.1 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::a ( ) const [inline]`

Return the  $a$  parameter of the distribution.

Definition at line 4393 of file random.h.

### 5.723.3.2 `template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::b ( ) const [inline]`

Return the  $b$  parameter of the distribution.

Definition at line 4400 of file random.h.

### 5.723.3.3 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4429 of file random.h.

### 5.723.3.4 `template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4422 of file random.h.

### 5.723.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::weibull_distribution< _RealType >::operator() ( _UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4437 of file `random.h`.

References `std::weibull_distribution<_RealType>::operator()`, and `std::weibull_distribution<_RealType>::param()`.

Referenced by `std::weibull_distribution<_RealType>::operator()`.

**5.723.3.6** `template<typename _RealType = double> void  
std::weibull_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 4415 of file `random.h`.

**5.723.3.7** `template<typename _RealType = double> param_type  
std::weibull_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4407 of file `random.h`.

Referenced by `std::weibull_distribution<_RealType>::operator()`, `std::operator==( )`, and `std::operator>>()`.

**5.723.3.8** `template<typename _RealType = double> void  
std::weibull_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4386 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.724 `std::weibull_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `weibull_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 5.724.1 Detailed Description

`template<typename _RealType = double> struct std::weibull_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4344 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

# Chapter 6

## File Documentation

### 6.1 algo.h File Reference

Parallel STL function calls corresponding to the [stl\\_algo.h](#) header.

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber< \\_MustBeInt >](#)  
*Functor wrapper for std::rand().*

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

#### Functions

- `template<typename _RAIter >`  
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter`  
`__end, random_access_iterator_tag)`
- `template<typename _FIterator, typename _IteratorTag >`  
`_FIterator std::__parallel::__adjacent_find_switch (_FIterator __begin, _-`  
`FIterator __end, _IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::__parallel::__adjacent_find_switch (_FIterator __begin, _-`  
`FIterator __end, _BinaryPredicate __pred, _IteratorTag)`

- `template<typename _RAIter, typename _BinaryPredicate >`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter`  
`__end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Predicate >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_`  
`switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_`  
`access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_`  
`parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if_switch`  
`(_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_`  
`switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_`  
`access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_`  
`parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_`  
`_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`  
`IteratorTag >`  
`_RAIter std::parallel::find_first_of_switch (_RAIter __begin1, _RAIter`  
`__end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp,`  
`random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __`  
`end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _`  
`IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`  
`>`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1,`  
`_FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`_Iter std::parallel::find_if_switch (_Iter __begin, _Iter __end, _`  
`Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end,`  
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter std::parallel::find_switch (_Iter __begin, _Iter __end, const _Tp`  
`&__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end,`  
`const _Tp &__val, random_access_iterator_tag)`

- `template<typename _Iter , typename _Function , typename _IteratorTag >`  
`_Function std::__parallel::for_each_switch (_Iter __begin, _Iter __end, _`  
`Function __f, _IteratorTag)`
- `template<typename _RAIter , typename _Function >`  
`_Function std::__parallel::for_each_switch (_RAIter __begin, _RAIter _`  
`__end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`__parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _OutputIterator , typename _Size , typename _Generator , typename _`  
`IteratorTag >`  
`_OutputIterator std::__parallel::generate_n_switch (_OutputIterator __`  
`begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Size , typename _Generator >`  
`_RAIter std::__parallel::generate_n_switch (_RAIter __begin, _Size __n,`  
`_Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`__parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator , typename _Generator , typename _IteratorTag >`  
`void std::__parallel::generate_switch (_FIterator __begin, _FIterator __end,`  
`_Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Generator >`  
`void std::__parallel::generate_switch (_RAIter __begin, _RAIter __end, _`  
`Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`__parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`  
`_FIterator std::__parallel::max_element_switch (_FIterator __begin, _`  
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`  
`_RAIter std::__parallel::max_element_switch (_RAIter __begin, _RAIter`  
`__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare ,`  
`typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OutputIterator std::__parallel::merge_switch (_Iter1 __begin1, _Iter1 _`  
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`  
`__comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare >`  
`_OutputIterator std::__parallel::merge_switch (_Iter1 __begin1, _Iter1 _`  
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`  
`__comp, random_access_iterator_tag, random_access_iterator_tag, random_`  
`access_iterator_tag)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`  
`_FIterator std::__parallel::min_element_switch (_FIterator __begin, _`  
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`  
`_RAIter std::__parallel::min_element_switch (_RAIter __begin, _RAIter`

- 
- ```

__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

```
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator std::__parallel::partition_switch (_FIterator __begin, _FIterator`  
`__end, _Predicate __pred, _IteratorTag)`
  - `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::__parallel::partition_switch (_RAIter __begin, _RAIter __`  
`end, _Predicate __pred, random_access_iterator_tag)`
  - `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void std::__parallel::replace_if_switch (_FIterator __begin, _FIterator __`  
`end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
  - `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if_switch (_RAIter __begin, _RAIter __`  
`end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_`  
`tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_`  
`balanced)`
  - `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void std::__parallel::replace_switch (_FIterator __begin, _FIterator __end,`  
`const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
  - `template<typename _RAIter, typename _Tp >`  
`void std::__parallel::replace_switch (_RAIter __begin, _RAIter __end,`  
`const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_`  
`tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_`  
`balanced)`
  - `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate,`  
`typename _IteratorTag >`  
`_FIterator std::__parallel::search_n_switch (_FIterator __begin, _FIterator`  
`__end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`  
`_IteratorTag)`
  - `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter std::__parallel::search_n_switch (_RAIter __begin, _RAIter __`  
`end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`  
`random_access_iterator_tag)`
  - `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::__parallel::search_switch (_RAIter1 __begin1, _RAIter1 __`  
`end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag,`  
`random_access_iterator_tag)`
  - `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename`  
`_IteratorTag2 >`  
`_FIterator1 std::__parallel::search_switch (_FIterator1 __begin1, _`  
`FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1,`  
`_IteratorTag2)`
  - `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 std::__parallel::search_switch (_RAIter1 __begin1, _RAIter1`
-

- \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, \_BinaryPredicate \_\_pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
- template<typename \_FIterator1 , typename \_FIterator2 , typename \_BinaryPredicate , typename \_IteratorTag1 , typename \_IteratorTag2 >  
\_FIterator1 **std::parallel::search\_switch** (\_FIterator1 \_\_begin1, \_FIterator1 \_\_end1, \_FIterator2 \_\_begin2, \_FIterator2 \_\_end2, \_BinaryPredicate \_\_pred, \_IteratorTag1, \_IteratorTag2)
  - template<typename \_IIter1 , typename \_IIter2 , typename \_Predicate , typename \_OutputIterator , typename \_IteratorTag1 , typename \_IteratorTag2 , typename \_IteratorTag3 >  
\_OutputIterator **std::parallel::set\_difference\_switch** (\_IIter1 \_\_begin1, \_IIter1 \_\_end1, \_IIter2 \_\_begin2, \_IIter2 \_\_end2, \_OutputIterator \_\_result, \_Predicate \_\_pred, \_IteratorTag1, \_IteratorTag2, \_IteratorTag3)
  - template<typename \_RAIter1 , typename \_RAIter2 , typename \_Output\_RAIter , typename \_Predicate >  
\_Output\_RAIter **std::parallel::set\_difference\_switch** (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, \_Output\_RAIter \_\_result, \_Predicate \_\_pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_IIter1 , typename \_IIter2 , typename \_Predicate , typename \_OutputIterator , typename \_IteratorTag1 , typename \_IteratorTag2 , typename \_IteratorTag3 >  
\_OutputIterator **std::parallel::set\_intersection\_switch** (\_IIter1 \_\_begin1, \_IIter1 \_\_end1, \_IIter2 \_\_begin2, \_IIter2 \_\_end2, \_OutputIterator \_\_result, \_Predicate \_\_pred, \_IteratorTag1, \_IteratorTag2, \_IteratorTag3)
  - template<typename \_RAIter1 , typename \_RAIter2 , typename \_Output\_RAIter , typename \_Predicate >  
\_Output\_RAIter **std::parallel::set\_intersection\_switch** (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, \_Output\_RAIter \_\_result, \_Predicate \_\_pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_IIter1 , typename \_IIter2 , typename \_Predicate , typename \_OutputIterator , typename \_IteratorTag1 , typename \_IteratorTag2 , typename \_IteratorTag3 >  
\_OutputIterator **std::parallel::set\_symmetric\_difference\_switch** (\_IIter1 \_\_begin1, \_IIter1 \_\_end1, \_IIter2 \_\_begin2, \_IIter2 \_\_end2, \_OutputIterator \_\_result, \_Predicate \_\_pred, \_IteratorTag1, \_IteratorTag2, \_IteratorTag3)
  - template<typename \_RAIter1 , typename \_RAIter2 , typename \_Output\_RAIter , typename \_Predicate >  
\_Output\_RAIter **std::parallel::set\_symmetric\_difference\_switch** (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, \_Output\_RAIter \_\_result, \_Predicate \_\_pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_IIter1 , typename \_IIter2 , typename \_Predicate , typename \_OutputIterator , typename \_IteratorTag1 , typename \_IteratorTag2 , typename \_IteratorTag3 >  
\_OutputIterator **std::parallel::set\_union\_switch** (\_IIter1 \_\_begin1, \_IIter1 \_\_end1, \_IIter2 \_\_begin2, \_IIter2 \_\_end2, \_OutputIterator \_\_result, \_Predicate \_\_pred, \_IteratorTag1, \_IteratorTag2, \_IteratorTag3)

- `template<typename _RAIter1 , typename _RAIter2 , typename _OutputRAIter , typename _Predicate >`  
`_OutputRAIter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _OutputRAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BinaryOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter , typename _OutputIterator , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter , typename RandomAccessOutputIterator , typename _Predicate >`  
`RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator , typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end)`

- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __-`  
`end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __-`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __-`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __-`  
`parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __-`  
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __-`  
`begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __-`  
`begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __-`  
`parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __-`  
`begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, -`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`  
`__begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`  
`__begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`  
`__begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, -`  
`FIterator __begin2, FIterator __end2, \_BinaryPredicate __comp, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`  
`Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function`  
`__f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`  
`Function __f)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`  
`Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`  
`Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`  
`Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`  
`_n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`  
`_n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`  
`_n, _Generator __gen)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __-`  
`end)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _-`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)`
  
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
  
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end)`

- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __-`  
`parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_difference (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`  
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _-`  
`Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`  
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`  
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _-`  
`Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`  
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_`  
`parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu-`  
`parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_`  
`parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_`  
`parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_`  
`parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_`  
`parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`  
`__comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __`  
`end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`  
`BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`  
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end, _`  
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`  
`BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`  
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end,`  
`_OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`  
`BinaryOperation >`

- `_OutputIterator` **std::\_\_parallel::transform** (`_Iter1 __begin1`, `_Iter1 __end1`, `_Iter2 __begin2`, `_OutputIterator __result`, `_BinaryOperation __binary_op`, `__gnu_parallel::_Parallelism __parallelism_tag`)
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator` **std::\_\_parallel::unique\_copy** (`_Iter __begin1`, `_Iter __end1`, `_OutputIterator __out`)
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator` **std::\_\_parallel::unique\_copy** (`_Iter __begin1`, `_Iter __end1`, `_OutputIterator __out`, `_Predicate __pred`)
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator` **std::\_\_parallel::unique\_copy** (`_Iter __begin1`, `_Iter __end1`, `_OutputIterator __out`, `__gnu_parallel::sequential_tag`)
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator` **std::\_\_parallel::unique\_copy** (`_Iter __begin1`, `_Iter __end1`, `_OutputIterator __out`, `_Predicate __pred`, `__gnu_parallel::sequential_tag`)

### 6.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file `algo.h`.

## 6.2 `algbase.h` File Reference

Parallel STL function calls corresponding to the `stl_algbase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace `std`
- namespace `std::__parallel`

### Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool` **std::\_\_parallel::\_\_lexicographical\_compare\_switch** (`_Iter1 __begin1`,

```

    _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _-
    IteratorTag1, _IteratorTag2)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  bool std::parallel::lexicographical_compare_switch (_RAIter1 __-
  begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate
  __pred, random_access_iterator_tag, random_access_iterator_tag)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1
  __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_-
  access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,
  typename _IteratorTag2 >
  pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1 __-
  begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _-
  IteratorTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __-
  begin2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-
  end1, _Iter2 __begin2, _Iter2 __end2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_-
  parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred)

```

- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`

### 6.2.1 Detailed Description

Parallel STL function calls corresponding to the [stl\\_algobase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algobase.h](#).

## 6.3 algorithm File Reference

### 6.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [algorithm](#).

## 6.4 algorithm File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > __gnu_cxx::__copy_n (_InputIterator`  
`__first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`  
`pair< _RAIterator, _OutputIterator > __gnu_cxx::__copy_n (_RAIterator __-`  
`first, _Size __count, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

- `int __gnu_cxx::__lexicographical_compare_3way` (const unsigned char \*\_\_first1, const unsigned char \*\_\_last1, const unsigned char \*\_\_first2, const unsigned char \*\_\_last2)
- `int __gnu_cxx::__lexicographical_compare_3way` (const char \*\_\_first1, const char \*\_\_last1, const char \*\_\_first2, const char \*\_\_last2)
- `template<typename _Tp, typename _Compare >`  
`const _Tp & __gnu_cxx::__median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c, \_Compare \_\_comp)
- `template<typename _Tp >`  
`const _Tp & __gnu_cxx::__median` (const \_Tp &\_\_a, const \_Tp &\_\_b, const \_Tp &\_\_c)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator __gnu_cxx::__random_sample` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_\_out, const \_Distance \_\_n)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator __gnu_cxx::__random_sample` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_\_out, \_RandomNumberGenerator &\_\_rand, const \_Distance \_\_n)
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair<_InputIterator, _OutputIterator > __gnu_cxx::__copy_n` (\_InputIterator \_\_first, \_Size \_\_count, \_OutputIterator \_\_result)
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void __gnu_cxx::__count` (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Tp &\_\_value, \_Size &\_\_n)
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void __gnu_cxx::__count_if` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Predicate \_\_pred, \_Size &\_\_n)
- `template<typename _RandomAccessIterator >`  
`bool __gnu_cxx::__is_heap` (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool __gnu_cxx::__is_heap` (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_StrictWeakOrdering \_\_comp)
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`  
`bool __gnu_cxx::__is_sorted` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_StrictWeakOrdering \_\_comp)
- `template<typename _ForwardIterator >`  
`bool __gnu_cxx::__is_sorted` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::__lexicographical_compare_3way` (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_InputIterator2 \_\_last2)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`

- ```

    _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-
    _first, _InputIterator __last, _RandomAccessIterator __out_first, _-
    RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)

```
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-`  
`__first, _InputIterator __last, _RandomAccessIterator __out_first, _-`  
`RandomAccessIterator __out_last)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename`  
`_RandomNumberGenerator >`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`  
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _-`  
`RandomNumberGenerator &__rand)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`  
`ForwardIterator __last, _OutputIterator __out, const _Distance __n)`

### 6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/algorithm](#).

## 6.5 algorithm File Reference

### 6.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [parallel/algorithm](#).

## 6.6 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Filter >`  
`_Filter std::adjacent_find (_Filter, _Filter)`

- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`  
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
  
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _-`  
`Predicate)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _-`  
`Compare)`
- `template<typename _Filter, typename _Tp >`  
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`

- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
  
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`  
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`  
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter >`  
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`  
`void std::iter_swap (_Filter1, _Filter2)`

- `template<typename _Iter1, typename _Iter2 >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`  
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`

- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _FIter >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2, _BinaryPredicate)`
- `template<typename _BIter >`  
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _IIter, typename _Predicate >`  
`bool std::none_of (_IIter, _IIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter, typename _Compare >`  
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter >`  
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > std::partition_copy (_IIter, _IIter, _OIter1, _OIter2, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::pop_heap (_RAIter, _RAIter, _Compare)`

- `template<typename _BIter >`  
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter, typename _Generator >`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _FIter, typename _Tp >`  
`_FIter std::remove (_FIter, _FIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter std::remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`  
`_OIter std::remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Tp >`  
`void std::replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter std::replace_copy (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter std::replace_copy_if (_IIter, _IIter, _OIter, _Predicate, const _Tp &)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void std::replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _BIter >`  
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`  
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _FIter >`  
`void std::rotate (_FIter, _FIter, _FIter)`
- `template<typename _FIter, typename _OIter >`  
`_OIter std::rotate_copy (_FIter, _FIter, _FIter, _OIter)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`  
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp >`  
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &)`

- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
  
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _RAIter, typename _UGenerator >`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter >`  
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter, typename _Compare >`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _Tp, size_t _Nm >`  
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation >`  
`_OIter std::transform (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)`
- `template<typename _IIter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::transform (_IIter, _IIter, _OIter, _UnaryOperation)`

- `template<typename _Filter >`  
`_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::unique_copy (_IIter, _IIter, _OIter)`
- `template<typename _IIter, typename _OIter, typename _BinaryPredicate >`  
`_OIter std::unique_copy (_IIter, _IIter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

### 6.6.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/algorithmfwd.h](#).

## 6.7 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Filter, typename _IterTag >`  
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter >`  
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _RAIter, typename _BiPredicate >`  
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _IIter, typename _Predicate, typename _IterTag >`  
`iterator_traits< _IIter >::difference_type std::__parallel::__count_if_switch (_IIter, _IIter, _Predicate, _IterTag)`

- `template<typename _RAIter, typename _Predicate >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _FIter, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter std::parallel::find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter std::parallel::find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function std::parallel::for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n,`

- ```

_Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _-
_parallelism_tag=__gnu_parallel::parallel_balanced)

```
- ```

template<typename _FIter, typename _Generator, typename _IterTag >
void std::parallel::generate_switch (_FIter, _FIter, _Generator, _IterTag)

```
- ```

template<typename _RAIter, typename _Generator >
void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _-
Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _-
parallelism_tag=__gnu_parallel::parallel_balanced)

```
- ```

template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, type-
name _IterTag2 >
bool std::parallel::lexicographical_compare_switch (_IIter1, _IIter1, _-
IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)

```
- ```

template<typename _RAIter1, typename _RAIter2, typename _Predicate >
bool std::parallel::lexicographical_compare_switch (_RAIter1 __-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate
__pred, random_access_iterator_tag, random_access_iterator_tag)

```
- ```

template<typename _FIter, typename _Compare, typename _IterTag >
_FIter std::parallel::max_element_switch (_FIter, _FIter, _Compare, _-
IterTag)

```
- ```

template<typename _RAIter, typename _Compare >
_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter
__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::-
Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

```
- ```

template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::parallel::merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _-
OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)

```
- ```

template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >
_OIter std::parallel::merge_switch (_IIter1, _IIter1, _IIter2, _IIter2,
_OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag,
random_access_iterator_tag)

```
- ```

template<typename _FIter, typename _Compare, typename _IterTag >
_FIter std::parallel::min_element_switch (_FIter, _FIter, _Compare, _-
IterTag)

```
- ```

template<typename _RAIter, typename _Compare >
_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter
__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::-
Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

```
- ```

template<typename _RAIter1, typename _RAIter2, typename _Predicate >
pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1
__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_-
access_iterator_tag, random_access_iterator_tag)

```
- ```

template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, type-
name _IterTag2 >

```

- pair<\_Iter1, \_Iter2 > **std::parallel::mismatch\_switch** (\_Iter1, \_Iter1, \_Iter2, \_Predicate, \_IterTag1, \_IterTag2)
- template<typename \_Filter, typename \_Predicate, typename \_IterTag >  
\_Filter **std::parallel::partition\_switch** (\_Filter, \_Filter, \_Predicate, \_IterTag)
  - template<typename \_RAIter, typename \_Predicate >  
\_RAIter **std::parallel::partition\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag)
  - template<typename \_Filter, typename \_Predicate, typename \_Tp, typename \_IterTag >  
void **std::parallel::replace\_if\_switch** (\_Filter, \_Filter, \_Predicate, const \_Tp &, \_IterTag)
  - template<typename \_RAIter, typename \_Predicate, typename \_Tp >  
void **std::parallel::replace\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, const \_Tp &\_\_new\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_balanced)
  - template<typename \_Filter, typename \_Tp, typename \_IterTag >  
void **std::parallel::replace\_switch** (\_Filter, \_Filter, const \_Tp &, const \_Tp &, \_IterTag)
  - template<typename \_RAIter, typename \_Tp >  
void **std::parallel::replace\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_balanced)
  - template<typename \_RAIter, typename \_Integer, typename \_Tp, typename \_BiPredicate >  
\_RAIter **std::parallel::search\_n\_switch** (\_RAIter, \_RAIter, \_Integer, const \_Tp &, \_BiPredicate, random\_access\_iterator\_tag)
  - template<typename \_Filter, typename \_Integer, typename \_Tp, typename \_BiPredicate, typename \_IterTag >  
\_Filter **std::parallel::search\_n\_switch** (\_Filter, \_Filter, \_Integer, const \_Tp &, \_BiPredicate, \_IterTag)
  - template<typename \_RAIter1, typename \_RAIter2 >  
\_RAIter1 **std::parallel::search\_switch** (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_Filter1, typename \_Filter2, typename \_IterTag1, typename \_IterTag2 >  
\_Filter1 **std::parallel::search\_switch** (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_IterTag1, \_IterTag2)
  - template<typename \_RAIter1, typename \_RAIter2, typename \_BiPredicate >  
\_RAIter1 **std::parallel::search\_switch** (\_RAIter1, \_RAIter1, \_RAIter2, \_RAIter2, \_BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_Filter1, typename \_Filter2, typename \_BiPredicate, typename \_IterTag1, typename \_IterTag2 >

- `_FIter1 std::__parallel::__search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::__set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::__parallel::__set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::__set_intersection_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::__parallel::__set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::__set_symmetric_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::__parallel::__set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::__set_union_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter std::__parallel::__set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _IIter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::__parallel::__transform1_switch (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter,`  
`_RAOIter, _UnaryOperation, random_access_iterator_tag, random_`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_`  
`parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation`  
`>`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _`  
`RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_`  
`access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism`  
`__parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, type-`  
`name _Tag1, typename _Tag2, typename _Tag3 >`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter,`  
`_BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _`  
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter std::parallel::unique_copy_switch (_RAIter,`  
`_RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, \_\_gnu\_`  
`parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_`  
`parallel::sequential_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __`  
`parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`  
`Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_-  
parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::-  
Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_-  
parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_-  
parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_-  
parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::-  
Parallelism)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_-  
parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`

- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::max_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, \_\_gnu\_parallel::Parallelism \_\_parallelism\_tag)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > std::__parallel::mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`pair< _IIter1, _IIter2 > std::__parallel::mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > std::__parallel::mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate >`  
`pair< _IIter1, _IIter2 > std::__parallel::mismatch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)`

- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::__parallel::partition (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::__parallel::partition (_FIter, _FIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Tp >`  
`void std::__parallel::replace (_FIter, _FIter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIter, typename _Tp >`  
`void std::__parallel::replace (_FIter, _FIter, const _Tp &, const _Tp &)`

- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`Filter1 std::__parallel::search (_Filter1, Filter1, _Filter2, Filter2, \_BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`Filter1 std::__parallel::search (_Filter1, Filter1, _Filter2, Filter2, \_BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`Filter1 std::__parallel::search (_Filter1, Filter1, _Filter2, Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`Filter1 std::__parallel::search (_Filter1, Filter1, _Filter2, Filter2)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`Filter std::__parallel::search_n (_Filter, Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`Filter std::__parallel::search_n (_Filter, Filter, _Integer, const _Tp &, \_BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`Filter std::__parallel::search_n (_Filter, Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`Filter std::__parallel::search_n (_Filter, Filter, _Integer, const _Tp &, \_BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`OIter std::__parallel::set_difference (_IIter1, IIter1, _IIter2, IIter2, OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`OIter std::__parallel::set_difference (_IIter1, IIter1, _IIter2, IIter2, OIter, \_Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`OIter std::__parallel::set_difference (_IIter1, IIter1, _IIter2, IIter2, OIter, \_Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`OIter std::__parallel::set_difference (_IIter1, IIter1, _IIter2, IIter2, OIter)`

- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _-`  
`OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _-`  
`OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _-`  
`OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _-`  
`OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _-`  
`IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _-`  
`IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _-`  
`IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _-`  
`IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _-`  
`Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _-`  
`Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`

- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_\_parallel::\_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_\_parallel::\_Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_\_parallel::sequential\_tag)`

### 6.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/algorithmfwd.h](#).

## 6.8 allocator.h File Reference

### Classes

- class [std::allocator<\\_Tp>](#)  
*The standard allocator, as per [20.4].*  
*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*
- class [std::allocator<void>](#)  
*allocator<void> specialization.*

### Namespaces

- namespace [std](#)

### Functions

- [template<typename \\_T1 , typename \\_T2 >](#)  
[bool std::operator!=](#) (const allocator<\_T1 > &, const allocator<\_T2 > &)
- [template<typename \\_Tp >](#)  
[bool std::operator!=](#) (const allocator<\_Tp > &, const allocator<\_Tp > &)
- [template<typename \\_T1 , typename \\_T2 >](#)  
[bool std::operator==](#) (const allocator<\_T1 > &, const allocator<\_T2 > &)
- [template<typename \\_Tp >](#)  
[bool std::operator==](#) (const allocator<\_Tp > &, const allocator<\_Tp > &)

#### 6.8.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [allocator.h](#).

## 6.9 array File Reference

### 6.9.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

## 6.10 array File Reference

### Classes

- struct [std::tr1::array](#)< [\\_Tp](#), [\\_Nm](#) >  
*A standard container for storing a fixed size sequence of elements.*

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
_Tp & std::tr1::get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
const _Tp & std::tr1::get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator!= (const array< _Tp, _Nm > &__one, const array<  
_Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator< (const array< _Tp, _Nm > &__a, const array< _Tp,  
_Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator<= (const array< _Tp, _Nm > &__one, const array<  
_Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator== (const array< _Tp, _Nm > &__one, const array<  
_Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator> (const array< _Tp, _Nm > &__one, const array<  
_Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator>= (const array< _Tp, _Nm > &__one, const array<  
_Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
void std::tr1::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__-  
two)`

### 6.10.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/array](#).

## 6.11 array\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator< \\_Tp, \\_Array >](#)  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base< \\_Tp >](#)  
*Base class.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp, typename _Array >`  
`bool \_\_gnu\_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp, typename _Array >`  
`bool \_\_gnu\_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

### 6.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array\\_allocator.h](#).

## 6.12 `assoc_container.hpp` File Reference

### Classes

- class `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator >  
*An abstract basic hash-based associative container.*
- class `__gnu_pbds::basic_tree`< Key, Mapped, Tag, Node\_Update, Policy\_TL, Allocator >  
*An abstract basic tree-like (tree, trie) associative container.*
- class `__gnu_pbds::cc_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, Allocator >  
*A concrete collision-chaining hash-based associative container.*
- class `__gnu_pbds::container_base`< Key, Mapped, Tag, Policy\_TL, Allocator >  
*An abstract basic associative container.*
- class `__gnu_pbds::gp_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, Allocator >  
*A concrete general-probing hash-based associative container.*
- class `__gnu_pbds::list_update`< Key, Mapped, Eq\_Fn, Update\_Policy, Allocator >  
*A list-update based associative container.*
- class `__gnu_pbds::tree`< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, Allocator >  
*A concrete basic tree-based associative container.*
- class `__gnu_pbds::trie`< Key, Mapped, E\_Access\_Traits, Tag, Node\_Update, Allocator >  
*A concrete basic trie-based associative container.*

### Namespaces

- namespace `__gnu_pbds`

## Defines

- #define `PB_DS_BASE_C_DEC`
- #define `PB_DS_CLASS_NAME`
- #define `PB_DS_CLASS_NAME`
- #define `PB_DS_CLASS_NAME`
- #define `PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- #define `PB_DS_TRIE_NODE_AND_ITS_TRAITS`

### 6.12.1 Detailed Description

Contains associative containers.

Definition in file [assoc\\_container.hpp](#).

## 6.13 atomic File Reference

### Classes

- struct `std::atomic< _Tp >`  
*atomic /// 29.4.3, Generic atomic type, primary class template.*
- struct `std::atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `std::atomic< bool >`  
*Explicit specialization for bool.*
- struct `std::atomic< char >`  
*Explicit specialization for char.*
- struct `std::atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `std::atomic< char32_t >`

*Explicit specialization for char32\_t.*

- struct `std::atomic< int >`  
*Explicit specialization for int.*
- struct `std::atomic< long >`  
*Explicit specialization for long.*
- struct `std::atomic< long long >`  
*Explicit specialization for long long.*
- struct `std::atomic< short >`  
*Explicit specialization for short.*
- struct `std::atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `std::atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `std::atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `std::atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::atomic< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `std::atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::atomic< void * >`  
*Explicit specialization for void\*.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Namespaces

- namespace `std`

## Functions

- memory\_order **std::\_\_calculate\_memory\_order** (memory\_order \_\_m)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_strong** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool **std::atomic\_compare\_exchange\_strong** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool **std::atomic\_compare\_exchange\_strong** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_strong\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **std::atomic\_compare\_exchange\_strong\_explicit** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **std::atomic\_compare\_exchange\_strong\_explicit** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_weak** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool **std::atomic\_compare\_exchange\_weak** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool **std::atomic\_compare\_exchange\_weak** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool **std::atomic\_compare\_exchange\_weak\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **std::atomic\_compare\_exchange\_weak\_explicit** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool **std::atomic\_compare\_exchange\_weak\_explicit** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- void \* **std::atomic\_exchange** (atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
\_ITp **std::atomic\_exchange** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- bool **std::atomic\_exchange** (atomic\_bool \*\_\_a, bool \_\_i)
- template<typename \_ITp >  
\_ITp **std::atomic\_exchange\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void \* **std::atomic\_exchange\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- bool **std::atomic\_exchange\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- void \* **std::atomic\_fetch\_add** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a, memory_order __m)`

- void \* **std::atomic\_load\_explicit** (const atomic\_address \*\_\_a, memory\_order \_\_m)
- bool **std::atomic\_load\_explicit** (const atomic\_bool \*\_\_a, memory\_order \_\_m)
- void **std::atomic\_store** (atomic\_bool \*\_\_a, bool \_\_i)
- template<typename \_ITp >  
void **std::atomic\_store** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void **std::atomic\_store** (atomic\_address \*\_\_a, void \*\_\_v)
- void **std::atomic\_store\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- template<typename \_Tp >  
\_Tp **std::kill\_dependency** (\_Tp \_\_y)

### 6.13.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

## 6.14 atomic\_0.h File Reference

### Classes

- struct [\\_\\_atomic0::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic0::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic0::atomic\\_flag](#)  
*atomic\_flag*

### Defines

- #define [\\_ATOMIC\\_CMPEXCHNG](#)(\_\_a, \_\_e, \_\_m, \_\_x)
- #define [\\_ATOMIC\\_LOAD](#)(\_\_a, \_\_x)
- #define [\\_ATOMIC\\_MODIFY](#)(\_\_a, \_\_o, \_\_m, \_\_x)
- #define [\\_ATOMIC\\_STORE](#)(\_\_a, \_\_m, \_\_x)

### 6.14.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_0.h](#).

## 6.15 atomic\_2.h File Reference

### Classes

- struct [\\_\\_atomic2::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic2::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic2::atomic\\_flag](#)  
*atomic\_flag*

### 6.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_2.h](#).

## 6.16 atomic\_base.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define [\\_ATOMIC\\_CMPEXCHNG\\_](#)(*\_\_a*, *\_\_e*, *\_\_m*, *\_\_x*)
- #define [\\_ATOMIC\\_LOAD\\_](#)(*\_\_a*, *\_\_x*)
- #define [\\_ATOMIC\\_MODIFY\\_](#)(*\_\_a*, *\_\_o*, *\_\_m*, *\_\_x*)
- #define [\\_ATOMIC\\_STORE\\_](#)(*\_\_a*, *\_\_m*, *\_\_x*)
- #define [\\_GLIBCXX\\_ATOMIC\\_BASE\\_H](#)

- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base **std::\_\_atomic\_flag\_base**
- typedef `atomic_short` **std::atomic\_int\_fast16\_t**
- typedef `atomic_int` **std::atomic\_int\_fast32\_t**
- typedef `atomic_llong` **std::atomic\_int\_fast64\_t**
- typedef `atomic_schar` **std::atomic\_int\_fast8\_t**
- typedef `atomic_short` **std::atomic\_int\_least16\_t**
- typedef `atomic_int` **std::atomic\_int\_least32\_t**
- typedef `atomic_llong` **std::atomic\_int\_least64\_t**
- typedef `atomic_schar` **std::atomic\_int\_least8\_t**
- typedef `atomic_llong` **std::atomic\_intmax\_t**
- typedef `atomic_long` **std::atomic\_intptr\_t**
- typedef `atomic_long` **std::atomic\_ptrdiff\_t**
- typedef `atomic_ulong` **std::atomic\_size\_t**
- typedef `atomic_long` **std::atomic\_ssize\_t**
- typedef `atomic_ushort` **std::atomic\_uint\_fast16\_t**
- typedef `atomic_uint` **std::atomic\_uint\_fast32\_t**
- typedef `atomic_ullong` **std::atomic\_uint\_fast64\_t**
- typedef `atomic_uchar` **std::atomic\_uint\_fast8\_t**
- typedef `atomic_ushort` **std::atomic\_uint\_least16\_t**
- typedef `atomic_uint` **std::atomic\_uint\_least32\_t**
- typedef `atomic_ullong` **std::atomic\_uint\_least64\_t**
- typedef `atomic_uchar` **std::atomic\_uint\_least8\_t**
- typedef `atomic_ullong` **std::atomic\_uintmax\_t**
- typedef `atomic_ulong` **std::atomic\_uintptr\_t**
- typedef enum `std::memory_order` **std::memory\_order**

## Enumerations

- enum `std::memory_order` {  
    **memory\_order\_relaxed**,   **memory\_order\_consume**,   **memory\_order\_-**  
    **acquire**, **memory\_order\_release**,  
    **memory\_order\_acq\_rel**, **memory\_order\_seq\_cst** }

## Functions

- void **std::\_\_atomic\_flag\_wait\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) \_GLIBCXX\_NOTHROW
- **std::\_\_attribute\_\_** ((\_\_const\_\_)) \_\_atomic\_flag\_base \*\_\_atomic\_flag\_for\_address(const void \*\_\_z) \_GLIBCXX\_NOTHROW
- void **std::atomic\_flag\_clear** (\_\_atomic\_flag\_base \*\_\_a)
- void **std::atomic\_flag\_clear\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) \_GLIBCXX\_NOTHROW
- bool **std::atomic\_flag\_test\_and\_set** (\_\_atomic\_flag\_base \*\_\_a)
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (\_\_atomic\_flag\_base \*, memory\_order) \_GLIBCXX\_NOTHROW

### 6.16.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic\\_base.h](#).

## 6.17 atomic\_word.h File Reference

### Typedefs

- typedef int **\_Atomic\_word**

### 6.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic\\_word.h](#).

## 6.18 atomicfwd\_c.h File Reference

### Defines

- #define **\_ATOMIC\_MEMBER\_**
- #define **atomic\_compare\_exchange**(\_\_a, \_\_e, \_\_m)
- #define **atomic\_compare\_exchange\_explicit**(\_\_a, \_\_e, \_\_m, \_\_x, \_\_y)
- #define **atomic\_exchange**(\_\_a, \_\_m)
- #define **atomic\_exchange\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_add**(\_\_a, \_\_m)

- #define **atomic\_fetch\_add\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_and**(\_\_a, \_\_m)
- #define **atomic\_fetch\_and\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_or**(\_\_a, \_\_m)
- #define **atomic\_fetch\_or\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_sub**(\_\_a, \_\_m)
- #define **atomic\_fetch\_sub\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_xor**(\_\_a, \_\_m)
- #define **atomic\_fetch\_xor\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_is\_lock\_free**(\_\_a)
- #define **atomic\_load**(\_\_a)
- #define **atomic\_load\_explicit**(\_\_a, \_\_x)
- #define **atomic\_store**(\_\_a, \_\_m)
- #define **atomic\_store\_explicit**(\_\_a, \_\_m, \_\_x)

## Typedefs

- typedef struct \_\_atomic\_address\_base **atomic\_address**
- typedef struct \_\_atomic\_bool\_base **atomic\_bool**
- typedef struct \_\_atomic\_char\_base **atomic\_char**
- typedef struct \_\_atomic\_short\_base **atomic\_char16\_t**
- typedef struct \_\_atomic\_int\_base **atomic\_char32\_t**
- typedef struct \_\_atomic\_flag\_base **atomic\_flag**
- typedef struct \_\_atomic\_int\_base **atomic\_int**
- typedef struct \_\_atomic\_llong\_base **atomic\_llong**
- typedef struct \_\_atomic\_long\_base **atomic\_long**
- typedef struct \_\_atomic\_schar\_base **atomic\_schar**
- typedef struct \_\_atomic\_short\_base **atomic\_short**
- typedef struct \_\_atomic\_uchar\_base **atomic\_uchar**
- typedef struct \_\_atomic\_uint\_base **atomic\_uint**
- typedef struct \_\_atomic\_ullong\_base **atomic\_ullong**
- typedef struct \_\_atomic\_ulong\_base **atomic\_ulong**
- typedef struct \_\_atomic\_ushort\_base **atomic\_ushort**
- typedef struct \_\_atomic\_wchar\_t\_base **atomic\_wchar\_t**

### 6.18.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_c.h](#).

## 6.19 atomicfwd\_cxx.h File Reference

### Defines

- #define `_ATOMIC_MEMBER_`

### Typedefs

- typedef `__atomic_base< char >` [atomic\\_char](#)
- typedef `__atomic_base< char16_t >` [atomic\\_char16\\_t](#)
- typedef `__atomic_base< char32_t >` [atomic\\_char32\\_t](#)
- typedef `__atomic_base< int >` [atomic\\_int](#)
- typedef `__atomic_base< long long >` [atomic\\_llong](#)
- typedef `__atomic_base< long >` [atomic\\_long](#)
- typedef `__atomic_base< signed char >` [atomic\\_schar](#)
- typedef `__atomic_base< short >` [atomic\\_short](#)
- typedef `__atomic_base< unsigned char >` [atomic\\_uchar](#)
- typedef `__atomic_base< unsigned int >` [atomic\\_uint](#)
- typedef `__atomic_base< unsigned long long >` [atomic\\_ullong](#)
- typedef `__atomic_base< unsigned long >` [atomic\\_ulong](#)
- typedef `__atomic_base< unsigned short >` [atomic\\_ushort](#)
- typedef `__atomic_base< wchar_t >` [atomic\\_wchar\\_t](#)

### 6.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_cxx.h](#).

## 6.20 atomicity.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_GLIBCXX_READ_MEM_BARRIER`
- #define `_GLIBCXX_WRITE_MEM_BARRIER`

## Functions

- static void `__gnu_cxx::__atomic_add` (volatile `_Atomic_word` \*`__mem`, int `__val`)
- static void `__gnu_cxx::__atomic_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- static `_Atomic_word` `__gnu_cxx::__attribute__((__unused__)) __exchange_and_add_dispatch` (`_Atomic_word` \*`__mem`)
- static `_Atomic_word` `__gnu_cxx::__exchange_and_add` (volatile `_Atomic_word` \*`__mem`, int `__val`)
- else return `__gnu_cxx::__exchange_and_add_single` (`__mem`, `__val`)
- static `_Atomic_word` `__gnu_cxx::__exchange_and_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- static `_Atomic_word` int `__val` `__gnu_cxx::if` (`__gthread_active_p`()) return `__exchange_and_add`(`__mem`

## Variables

- static `_Atomic_word` int `__val` `__gnu_cxx::__val`

### 6.20.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicity.h](#).

## 6.21 auto\_ptr.h File Reference

### Classes

- class `std::auto_ptr<_Tp>`  
*A simple smart pointer providing strict ownership semantics.*
- struct `std::auto_ptr_ref<_Tp1>`

### Namespaces

- namespace `std`

### 6.21.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [auto\\_ptr.h](#).

## 6.22 `balanced_quicksort.h` File Reference

Implementation of a dynamically load-balanced parallel quicksort.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter >](#)  
*Information local to one thread in the parallel quicksort run.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_qsb\_conquer (_QSBThreadLocal<_RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator\_traits<\_RAIter >::difference\_type \\_\\_gnu\\_parallel::\\_\\_qsb\\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_qsb\_local\_sort\_with\_helping (_QSBThreadLocal<_RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)`

### 6.22.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [balanced\\_quicksort.h](#).

## 6.23 base.h File Reference

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`  
*Similar to `std::binder1st`, but giving the argument types explicitly.*
- class `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`  
*Similar to `std::binder2nd`, but giving the argument types explicitly.*
- class `__gnu_parallel::__unary_negate< _Predicate, argument_type >`  
*Similar to `std::unary_negate`, but giving the argument types explicitly.*
- class `__gnu_parallel::__EqualFromLess< _T1, _T2, _Compare >`  
*Constructs predicate for equality from strict weak ordering predicate.*
- struct `__gnu_parallel::__EqualTo< _T1, _T2 >`  
*Similar to `std::equal_to`, but allows two different types.*
- struct `__gnu_parallel::__Less< _T1, _T2 >`  
*Similar to `std::less`, but allows two different types.*
- struct `__gnu_parallel::__Multiplies< _Tp1, _Tp2, _Result >`  
*Similar to `std::multiplies`, but allows two different types.*

- struct `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`  
*Similar to `std::plus`, but allows two different types.*
- class `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp >`  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class `__gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp >`  
*Iterator associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.*

## Namespaces

- namespace `__gnu_parallel`
- namespace `__gnu_sequential`
- namespace `std`
- namespace `std::__parallel`

## Defines

- `#define _GLIBCXX_PARALLEL_ASSERT(_Condition)`

## Functions

- void `__gnu_parallel::_decode2 (_CASable __x, int &__a, int &__b)`
- `_CASable __gnu_parallel::_encode2 (int __a, int __b)`
- `_ThreadIndex __gnu_parallel::_get_max_threads ()`
- bool `__gnu_parallel::_is_parallel (const _Parallelism __p)`
- template<typename `_RAIter`, typename `_Compare` >  
`_RAIter __gnu_parallel::_median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- template<typename `_Size` >  
`_Size __gnu_parallel::_rd_log2 (_Size __n)`
- template<typename `_Tp` >  
`const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)`
- template<typename `_Tp` >  
`const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)`

### 6.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

## 6.24 base.h File Reference

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)
- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### 6.24.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

## 6.25 basic\_file.h File Reference

### Namespaces

- namespace [std](#)

### 6.25.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_file.h](#).

## 6.26 `basic_ios.h` File Reference

### Classes

- class `std::basic_ios<_CharT, _Traits >`

*Virtual base class for all stream classes.*

*Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*

### Namespaces

- namespace `std`

### Functions

- template<typename \_Facet >  
const \_Facet & `std::__check_facet` (const \_Facet \*\_\_f)

#### 6.26.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_ios.h](#).

## 6.27 `basic_ios.tcc` File Reference

### Namespaces

- namespace `std`

#### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_ios.tcc](#).

## 6.28 `basic_iterator.h` File Reference

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

### 6.28.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [basic\\_iterator.h](#).

## 6.29 `basic_string.h` File Reference

### Classes

- class [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>](#)  
*Managing sequences of characters and character-like objects.*
- struct [std::hash<string>](#)  
*std::hash specialization for string.*
- struct [std::hash<u16string>](#)  
*std::hash specialization for u16string.*
- struct [std::hash<u32string>](#)  
*std::hash specialization for u32string.*
- struct [std::hash<wstring>](#)  
*std::hash specialization for wstring.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_istream<_CharT, _Traits> & std::getline (basic_istream<_CharT, _Traits> &__is, basic_string<_CharT, _Traits, _Alloc> &__str, _CharT __delim)`

- `template<>`  
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in,`  
`basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _-`  
`Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::getline (basic_istream< char > &__in, basic_-`  
`string< char > &__str, char __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`  
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`  
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _-`  
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-  
_CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _-  
_Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,  
_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type  
std::operator== (const basic_string< _CharT > &__lhs, const basic_string<  
_CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits,  
_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _-  
_Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is,`  
`basic_string< char > &__str)`
- `double std::stod (const string &__str, size_t * __idx=0)`
- `float std::stof (const string &__str, size_t * __idx=0)`
- `int std::stoi (const string &__str, size_t * __idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t * __idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t * __idx=0)`
- `long long std::stoll (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string<`  
`_CharT, _Traits, _Alloc > &__rhs)`
- `string std::to_string (long double __val)`
- `string std::to_string (int __val)`
- `string std::to_string (long __val)`
- `string std::to_string (unsigned __val)`
- `string std::to_string (float __val)`
- `string std::to_string (unsigned long long __val)`
- `string std::to_string (unsigned long __val)`
- `string std::to_string (double __val)`
- `string std::to_string (long long __val)`
- `wstring std::to_wstring (unsigned long __val)`
- `wstring std::to_wstring (long __val)`
- `wstring std::to_wstring (int __val)`
- `wstring std::to_wstring (unsigned __val)`
- `wstring std::to_wstring (unsigned long long __val)`
- `wstring std::to_wstring (long long __val)`
- `wstring std::to_wstring (double __val)`
- `wstring std::to_wstring (long double __val)`
- `wstring std::to_wstring (float __val)`

### 6.29.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.h](#).

## 6.30 `basic_string.tcc` File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc > basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc > basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

### 6.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.tcc](#).

## 6.31 `basic_types.hpp` File Reference

### Classes

- `struct \_\_gnu\_pbds::detail::value\_type\_base< Key, Mapped, Allocator, false >`
- `struct \_\_gnu\_pbds::detail::value\_type\_base< Key, Mapped, Allocator, true >`
- `struct \_\_gnu\_pbds::detail::value\_type\_base< Key, null_mapped_type, Allocator, false >`
- `struct \_\_gnu\_pbds::detail::value\_type\_base< Key, null_mapped_type, Allocator, true >`

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Defines

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

### 6.31.1 Detailed Description

Contains basic types used by containers.

Definition in file [basic\\_types.hpp](#).

## 6.32 binders.h File Reference

### Classes

- class [std::binder1st< \\_Operation >](#)  
*One of the binder functors.*
- class [std::binder2nd< \\_Operation >](#)  
*One of the binder functors.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Operation , typename _Tp >  
binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &_  
_x)`
- `template<typename _Operation , typename _Tp >  
binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &_  
_x)`

## Variables

- [std::binder1st std::\\_GLIBCXX\\_DEPRECATED\\_ATTR](#)

### 6.32.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [binders.h](#).

## 6.33 bitmap\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector< \\_Tp >](#)  
*\_\_mini\_vector<> is a stripped down version of the full-fledged std::vector<>.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Bitmap\\_counter< \\_Tp >](#)  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Ffit\\_finder< \\_Tp >](#)  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)  
*Bitmap Allocator, primary template.*
- class [\\_\\_gnu\\_cxx::free\\_list](#)  
*The free list class for managing chunks of memory to be given to and returned by the bitmap\_allocator.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)

### Defines

- [#define \\_BALLOC\\_ALIGN\\_BYTES](#)

## Enumerations

- enum { **bits\_per\_byte**, **bits\_per\_block** }

## Functions

- void **\_\_gnu\_cxx::\_\_detail::\_\_bit\_allocate** (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- void **\_\_gnu\_cxx::\_\_detail::\_\_bit\_free** (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator **\_\_gnu\_cxx::\_\_detail::\_\_lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t **\_\_gnu\_cxx::\_\_detail::\_\_num\_bitmaps** (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t **\_\_gnu\_cxx::\_\_detail::\_\_num\_blocks** (\_AddrPair \_\_ap)
- size\_t **\_\_gnu\_cxx::\_\_Bit\_scan\_forward** (size\_t \_\_num)
- template<typename \_Tp1, typename \_Tp2 >  
bool **\_\_gnu\_cxx::operator!=** (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()
- template<typename \_Tp1, typename \_Tp2 >  
bool **\_\_gnu\_cxx::operator==** (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()

### 6.33.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [bitmap\\_allocator.h](#).

### 6.33.2 Define Documentation

#### 6.33.2.1 #define \_BALLOC\_ALIGN\_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 45 of file [bitmap\\_allocator.h](#).

## 6.34 bitset File Reference

### Classes

- struct [std::\\_Base\\_bitset< \\_Nw >](#)

- struct `std::_Base_bitset< 0 >`
- struct `std::_Base_bitset< 1 >`
- class `std::bitset< _Nb >`

*The bitset class represents a fixed-size sequence of bits.*

- class `std::bitset< _Nb >::reference`
- struct `std::hash<::bitset< _Nb > >`

*std::hash specialization for bitset.*

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS(__n)`

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`

### 6.34.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

## 6.35 bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset<\\_Nb>](#)  
*Class [std::bitset](#) with additional safety/checking/debug instrumentation.*
- struct [std::hash<\\_\\_debug::bitset<\\_Nb>>](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<size_t _Nb>`  
`bitset<_Nb> std::__debug::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_ostream<_CharT, _Traits> & std::__debug::operator<< (std::basic\_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_istream<_CharT, _Traits> & std::__debug::operator>> (std::basic\_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::__debug::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::__debug::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`

#### 6.35.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

## 6.36 bitset File Reference

### Classes

- class [std::\\_\\_profile::bitset<\\_Nb>](#)  
*Class [std::bitset](#) wrapper with performance instrumentation.*
- struct [std::hash<\\_\\_profile::bitset<\\_Nb>>](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<size_t _Nb>  
bitset<_Nb> std::\_\_profile::operator&(const bitset<_Nb> &__x, const  
bitset<_Nb> &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_ostream<_CharT, _Traits> & std::\_\_profile::operator<<  
(std::basic\_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_istream<_CharT, _Traits> & std::\_\_profile::operator>>  
(std::basic\_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>  
bitset<_Nb> std::\_\_profile::operator^(const bitset<_Nb> &__x, const  
bitset<_Nb> &__y)`
- `template<size_t _Nb>  
bitset<_Nb> std::\_\_profile::operator|(const bitset<_Nb> &__x, const  
bitset<_Nb> &__y)`

#### 6.36.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

## 6.37 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define `_GLIBCXX_CLASS_REQUIRES`(\_type\_var, \_ns, \_concept)
- #define `_GLIBCXX_CLASS_REQUIRES2`(\_type\_var1, \_type\_var2, \_ns, \_concept)
- #define `_GLIBCXX_CLASS_REQUIRES3`(\_type\_var1, \_type\_var2, \_type\_var3, \_ns, \_concept)
- #define `_GLIBCXX_CLASS_REQUIRES4`(\_type\_var1, \_type\_var2, \_type\_var3, \_type\_var4, \_ns, \_concept)
- #define `_GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT`(\_OP, \_NAME)
- #define `_GLIBCXX_DEFINE_BINARY_PREDICATE_OPERATOR_CONSTRAINT`(\_OP, \_NAME)
- #define `_IsUnused`

### Functions

- template<class \_Tp >  
void `__gnu_cxx::__aux_require_boolean_expr` (const \_Tp &\_\_t)
- void `__gnu_cxx::__error_type_must_be_a_signed_integer_type` ()
- void `__gnu_cxx::__error_type_must_be_an_integer_type` ()
- void `__gnu_cxx::__error_type_must_be_an_unsigned_integer_type` ()
- template<class \_Concept >  
void `__gnu_cxx::__function_requires` ()

#### 6.37.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_concept\\_check.h](#).

## 6.38 boost\_sp\_counted\_base.h File Reference

### Classes

- class [std::tr1::bad\\_weak\\_ptr](#)  
*Exception possibly thrown by [shared\\_ptr](#).*

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- void [std::tr1::\\_\\_throw\\_bad\\_weak\\_ptr\(\)](#)

#### 6.38.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_sp\\_counted\\_base.h](#).

## 6.39 c++0x\_warning.h File Reference

#### 6.39.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [c++0x\\_warning.h](#).

## 6.40 c++allocator.h File Reference

### Defines

- #define [\\_\\_glibcxx\\_base\\_allocator](#)

### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++allocator.h](#).

## 6.41 c++config.h File Reference

### Defines

- #define **\_\_GLIBCXX\_\_**
- #define **\_\_glibcxx\_assert**(\_Condition)
- #define **\_\_N**(msgid)
- #define **\_GLIBCXX\_ATOMIC\_BUILTINS\_1**
- #define **\_GLIBCXX\_ATOMIC\_BUILTINS\_2**
- #define **\_GLIBCXX\_ATOMIC\_BUILTINS\_4**
- #define **\_GLIBCXX\_ATOMIC\_BUILTINS\_8**
- #define **\_GLIBCXX\_BEGIN\_EXTERN\_C**
- #define **\_GLIBCXX\_BEGIN\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_BEGIN\_NAMESPACE(X)**
- #define **\_GLIBCXX\_BEGIN\_NESTED\_NAMESPACE(X, Y)**
- #define **\_GLIBCXX\_CONST**
- #define **\_GLIBCXX\_DEPRECATED\_ATTR**
- #define **\_GLIBCXX\_END\_EXTERN\_C**
- #define **\_GLIBCXX\_END\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_END\_NAMESPACE**
- #define **\_GLIBCXX\_END\_NESTED\_NAMESPACE**
- #define **\_GLIBCXX\_EXTERN\_TEMPLATE**
- #define **\_GLIBCXX\_FAST\_MATH**
- #define **\_GLIBCXX\_HAS\_GTHREADS**
- #define **\_GLIBCXX\_HAVE\_ACOSF**
- #define **\_GLIBCXX\_HAVE\_ACOSL**
- #define **\_GLIBCXX\_HAVE\_AS\_SYMVER\_DIRECTIVE**
- #define **\_GLIBCXX\_HAVE\_ASINF**
- #define **\_GLIBCXX\_HAVE\_ASINL**
- #define **\_GLIBCXX\_HAVE\_ATAN2F**
- #define **\_GLIBCXX\_HAVE\_ATAN2L**
- #define **\_GLIBCXX\_HAVE\_ATANF**
- #define **\_GLIBCXX\_HAVE\_ATANL**
- #define **\_GLIBCXX\_HAVE\_ATTRIBUTE\_VISIBILITY**
- #define **\_GLIBCXX\_HAVE\_CEILF**

- #define **\_GLIBCXX\_HAVE\_CEILL**
- #define **\_GLIBCXX\_HAVE\_COMPLEX\_H**
- #define **\_GLIBCXX\_HAVE\_COSF**
- #define **\_GLIBCXX\_HAVE\_COSHF**
- #define **\_GLIBCXX\_HAVE\_COSHL**
- #define **\_GLIBCXX\_HAVE\_COSL**
- #define **\_GLIBCXX\_HAVE\_DLFCN\_H**
- #define **\_GLIBCXX\_HAVE\_EBADMSG**
- #define **\_GLIBCXX\_HAVE\_ECANCELED**
- #define **\_GLIBCXX\_HAVE\_EIDRM**
- #define **\_GLIBCXX\_HAVE\_ENDIAN\_H**
- #define **\_GLIBCXX\_HAVE\_ENODATA**
- #define **\_GLIBCXX\_HAVE\_ENOLINK**
- #define **\_GLIBCXX\_HAVE\_ENOSR**
- #define **\_GLIBCXX\_HAVE\_ENOSTR**
- #define **\_GLIBCXX\_HAVE\_ENOTRECOVERABLE**
- #define **\_GLIBCXX\_HAVE\_ENOTSUP**
- #define **\_GLIBCXX\_HAVE\_EOVERFLOW**
- #define **\_GLIBCXX\_HAVE\_EOWNERDEAD**
- #define **\_GLIBCXX\_HAVE\_EPROTO**
- #define **\_GLIBCXX\_HAVE\_ETIME**
- #define **\_GLIBCXX\_HAVE\_ETXTBSY**
- #define **\_GLIBCXX\_HAVE\_EXECINFO\_H**
- #define **\_GLIBCXX\_HAVE\_EXPF**
- #define **\_GLIBCXX\_HAVE\_EXPL**
- #define **\_GLIBCXX\_HAVE\_FABSF**
- #define **\_GLIBCXX\_HAVE\_FABSL**
- #define **\_GLIBCXX\_HAVE\_FENV\_H**
- #define **\_GLIBCXX\_HAVE\_FINITE**
- #define **\_GLIBCXX\_HAVE\_FINITEF**
- #define **\_GLIBCXX\_HAVE\_FINITEL**
- #define **\_GLIBCXX\_HAVE\_FLOAT\_H**
- #define **\_GLIBCXX\_HAVE\_FLOORF**
- #define **\_GLIBCXX\_HAVE\_FLOORL**
- #define **\_GLIBCXX\_HAVE\_FMODF**
- #define **\_GLIBCXX\_HAVE\_FMODL**
- #define **\_GLIBCXX\_HAVE\_FREXPF**
- #define **\_GLIBCXX\_HAVE\_FREXPL**
- #define **\_GLIBCXX\_HAVE\_GETIPINFO**
- #define **\_GLIBCXX\_HAVE\_GTHR\_DEFAULT**
- #define **\_GLIBCXX\_HAVE\_HYPOT**
- #define **\_GLIBCXX\_HAVE\_HYPOTF**
- #define **\_GLIBCXX\_HAVE\_HYPOTL**

---

- #define `_GLIBCXX_HAVE_ICONV`
- #define `_GLIBCXX_HAVE_INT64_T`
- #define `_GLIBCXX_HAVE_INT64_T_LONG`
- #define `_GLIBCXX_HAVE_INTTYPES_H`
- #define `_GLIBCXX_HAVE_ISINF`
- #define `_GLIBCXX_HAVE_ISINFF`
- #define `_GLIBCXX_HAVE_ISINFL`
- #define `_GLIBCXX_HAVE_ISNAN`
- #define `_GLIBCXX_HAVE_ISNANF`
- #define `_GLIBCXX_HAVE_ISNANL`
- #define `_GLIBCXX_HAVE_ISWBLANK`
- #define `_GLIBCXX_HAVE_LC_MESSAGES`
- #define `_GLIBCXX_HAVE_LDEXPF`
- #define `_GLIBCXX_HAVE_LDEXPL`
- #define `_GLIBCXX_HAVE_LIBINTL_H`
- #define `_GLIBCXX_HAVE_LIMIT_AS`
- #define `_GLIBCXX_HAVE_LIMIT_DATA`
- #define `_GLIBCXX_HAVE_LIMIT_FSIZE`
- #define `_GLIBCXX_HAVE_LIMIT_RSS`
- #define `_GLIBCXX_HAVE_LIMIT_VMEM`
- #define `_GLIBCXX_HAVE_LINUX_FUTEX`
- #define `_GLIBCXX_HAVE_LOCALE_H`
- #define `_GLIBCXX_HAVE_LOG10F`
- #define `_GLIBCXX_HAVE_LOG10L`
- #define `_GLIBCXX_HAVE_LOGF`
- #define `_GLIBCXX_HAVE_LOGL`
- #define `_GLIBCXX_HAVE_MBSTATE_T`
- #define `_GLIBCXX_HAVE_MEMORY_H`
- #define `_GLIBCXX_HAVE_MODF`
- #define `_GLIBCXX_HAVE_MODFF`
- #define `_GLIBCXX_HAVE_MODFL`
- #define `_GLIBCXX_HAVE_POLL`
- #define `_GLIBCXX_HAVE_POWF`
- #define `_GLIBCXX_HAVE_POWL`
- #define `_GLIBCXX_HAVE_S_ISREG`
- #define `_GLIBCXX_HAVE_SETENV`
- #define `_GLIBCXX_HAVE_SINCOS`
- #define `_GLIBCXX_HAVE_SINCOSF`
- #define `_GLIBCXX_HAVE_SINCOSL`
- #define `_GLIBCXX_HAVE_SINF`
- #define `_GLIBCXX_HAVE_SINHF`
- #define `_GLIBCXX_HAVE_SINHL`
- #define `_GLIBCXX_HAVE_SINL`

- #define `_GLIBCXX_HAVE_SQRTF`
- #define `_GLIBCXX_HAVE_SQRTL`
- #define `_GLIBCXX_HAVE_STDBOOL_H`
- #define `_GLIBCXX_HAVE_STDINT_H`
- #define `_GLIBCXX_HAVE_STDLIB_H`
- #define `_GLIBCXX_HAVE_STRERROR_L`
- #define `_GLIBCXX_HAVE_STRERROR_R`
- #define `_GLIBCXX_HAVE_STRING_H`
- #define `_GLIBCXX_HAVE_STRINGS_H`
- #define `_GLIBCXX_HAVE_STRTOF`
- #define `_GLIBCXX_HAVE_STRTOLD`
- #define `_GLIBCXX_HAVE_STRXFRM_L`
- #define `_GLIBCXX_HAVE_SYS_IOCTL_H`
- #define `_GLIBCXX_HAVE_SYS_IPC_H`
- #define `_GLIBCXX_HAVE_SYS_PARAM_H`
- #define `_GLIBCXX_HAVE_SYS_RESOURCE_H`
- #define `_GLIBCXX_HAVE_SYS_SEM_H`
- #define `_GLIBCXX_HAVE_SYS_STAT_H`
- #define `_GLIBCXX_HAVE_SYS_TIME_H`
- #define `_GLIBCXX_HAVE_SYS_TYPES_H`
- #define `_GLIBCXX_HAVE_SYS_UIO_H`
- #define `_GLIBCXX_HAVE_TANF`
- #define `_GLIBCXX_HAVE_TANHF`
- #define `_GLIBCXX_HAVE_TANHL`
- #define `_GLIBCXX_HAVE_TANL`
- #define `_GLIBCXX_HAVE_TGMATH_H`
- #define `_GLIBCXX_HAVE_TLS`
- #define `_GLIBCXX_HAVE_UNISTD_H`
- #define `_GLIBCXX_HAVE_VFWSCANF`
- #define `_GLIBCXX_HAVE_VSWSCANF`
- #define `_GLIBCXX_HAVE_VWSCANF`
- #define `_GLIBCXX_HAVE_WCHAR_H`
- #define `_GLIBCXX_HAVE_WCSTOF`
- #define `_GLIBCXX_HAVE_WCTYPE_H`
- #define `_GLIBCXX_HAVE_WRITEV`
- #define `_GLIBCXX_HOSTED`
- #define `_GLIBCXX_ICONV_CONST`
- #define `_GLIBCXX_LDBL_NAMESPACE`
- #define `_GLIBCXX_NAMESPACE_ASSOCIATION_VERSION`
- #define `_GLIBCXX_NORETURN`
- #define `_GLIBCXX_NOTHROW`
- #define `_GLIBCXX_PACKAGE_GLIBCXX_VERSION`
- #define `_GLIBCXX_PACKAGE_BUGREPORT`

- #define `_GLIBCXX_PACKAGE_NAME`
- #define `_GLIBCXX_PACKAGE_STRING`
- #define `_GLIBCXX_PACKAGE_TARNAME`
- #define `_GLIBCXX_PACKAGE_URL`
- #define `_GLIBCXX_PSEUDO_VISIBILITY(V)`
- #define `_GLIBCXX_PURE`
- #define `_GLIBCXX_RES_LIMITS`
- #define `_GLIBCXX_STD`
- #define `_GLIBCXX_STD_D`
- #define `_GLIBCXX_STD_P`
- #define `_GLIBCXX_STD_PR`
- #define `_GLIBCXX_STDIO_MACROS`
- #define `_GLIBCXX_SYMVER`
- #define `_GLIBCXX_SYMVER_GNU`
- #define `_GLIBCXX_USE_C99`
- #define `_GLIBCXX_USE_C99_COMPLEX`
- #define `_GLIBCXX_USE_C99_COMPLEX_TR1`
- #define `_GLIBCXX_USE_C99_CTYPE_TR1`
- #define `_GLIBCXX_USE_C99_FENV_TR1`
- #define `_GLIBCXX_USE_C99_INTTYPES_TR1`
- #define `_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- #define `_GLIBCXX_USE_C99_MATH`
- #define `_GLIBCXX_USE_C99_MATH_TR1`
- #define `_GLIBCXX_USE_C99_STDINT_TR1`
- #define `_GLIBCXX_USE_DECIMAL_FLOAT`
- #define `_GLIBCXX_USE_GETTIMEOFDAY`
- #define `_GLIBCXX_USE_LFS`
- #define `_GLIBCXX_USE_LONG_LONG`
- #define `_GLIBCXX_USE_NLS`
- #define `_GLIBCXX_USE_RANDOM_TR1`
- #define `_GLIBCXX_USE_WCHAR_T`
- #define `_GLIBCXX_VISIBILITY_ATTR(V)`
- #define `_GLIBCXX_WEAK_DEFINITION`
- #define `LT_OBJDIR`
- #define `STDC_HEADERS`

### 6.41.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++config.h](#).

## 6.42 c++io.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef FILE [std::\\_\\_c\\_file](#)
- typedef [\\_\\_gthread\\_mutex\\_t std::\\_\\_c\\_lock](#)

#### 6.42.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++io.h](#).

## 6.43 c++locale.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- [#define \\_GLIBCXX\\_C\\_LOCALE\\_GNU](#)
- [#define \\_GLIBCXX\\_NUM\\_CATEGORIES](#)

### Typedefs

- typedef [\\_\\_locale\\_t std::\\_\\_c\\_locale](#)

### Functions

- int [std::\\_\\_convert\\_from\\_v](#) (const [\\_\\_c\\_locale](#) &[\\_\\_cloc](#) [\\_\\_attribute\\_\\_\(\(\\_\\_unused\\_\\_\)\)](#)), char \*[\\_\\_out](#), const int [\\_\\_size](#) [\\_\\_attribute\\_\\_\(\(\\_\\_unused\\_\\_\)\)](#)), const char \*[\\_\\_fmt](#),...)

### 6.43.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale.h](#).

## 6.44 c++locale\_internal.h File Reference

### 6.44.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale\\_internal.h](#).

## 6.45 cassert File Reference

### 6.45.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cassert](#).

## 6.46 ccomplex File Reference

### 6.46.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

## 6.47 `ccomplex` File Reference

### 6.47.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

## 6.48 `cctype` File Reference

### Namespaces

- namespace [std](#)

### 6.48.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `cctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cctype](#).

## 6.49 `cctype` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.49.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cctype](#).

## 6.50 `cctype` File Reference

### 6.50.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cctype](#).

## 6.51 `cerrno` File Reference

### Defines

- #define `errno`

### 6.51.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cerrno](#).

## 6.52 `cfenv` File Reference

### 6.52.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfenv](#).

## 6.53 `cfenv` File Reference

### Defines

- #define `_GLIBCXX_BEGIN_NAMESPACE_TR1`
- #define `_GLIBCXX_END_NAMESPACE_TR1`
- #define `_GLIBCXX_TR1`

### 6.53.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfenv](#).

## 6.54 `cfenv` File Reference

### 6.54.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cfenv](#).

## 6.55 `cfloat` File Reference

### Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.55.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cfloat](#).

## 6.56 `cfloat` File Reference

### Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

## 6.57 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)  
*Mapping from character type to associated types.*
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)  
*Base class used to implement `std::char_traits`.*
- struct [std::char\\_traits<\\_CharT>](#)  
*Basis for explicit traits specializations.*
- struct [std::char\\_traits<char>](#)  
*21.1.3.1 `char_traits` specializations*
- struct [std::char\\_traits<wchar\\_t>](#)  
*21.1.3.2 `char_traits` specializations*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Defines

- `#define` [\\_CHAR\\_TRAITS\\_EOF](#)

### 6.57.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [char\\_traits.h](#).

## 6.58 checkers.h File Reference

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Compare >  
bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __-  
comp)`

#### 6.58.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [checkers.h](#).

## 6.59 chrono File Reference

### Classes

- struct [std::chrono::duration< \\_Rep, \\_Period >](#)  
*duration*
- struct [std::chrono::duration\\_values< \\_Rep >](#)  
*duration\_values*
- struct [std::chrono::system\\_clock](#)  
*system\_clock*
- struct [std::chrono::time\\_point< \\_Clock, \\_Duration >](#)  
*time\_point*
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)  
*treat\_as\_floating\_point*

## Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

## Typedefs

- typedef system\_clock [std::chrono::high\\_resolution\\_clock](#)
- typedef duration< int, ratio< 3600 > > [std::chrono::hours](#)
- typedef duration< int64\_t, micro > [std::chrono::microseconds](#)
- typedef duration< int64\_t, milli > [std::chrono::milliseconds](#)
- typedef duration< int, ratio< 60 > > [std::chrono::minutes](#)
- typedef system\_clock [std::chrono::monotonic\\_clock](#)
- typedef duration< int64\_t, nano > [std::chrono::nanoseconds](#)
- typedef duration< int64\_t > [std::chrono::seconds](#)

## Functions

- template<typename \_ToDuration , typename \_Rep , typename \_Period >  
enable\_if< \_\_is\_duration< \_ToDuration >::value, \_ToDuration >::type  
[std::chrono::duration\\_cast](#) (const duration< \_Rep, \_Period > &\_d)
- template<typename \_Clock, typename \_Duration1 , typename \_Duration2 >  
bool [std::chrono::operator!=](#) (const time\_point< \_Clock, \_Duration1 > &\_\_lhs, const time\_point< \_Clock, \_Duration2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1 , typename \_Rep2 , typename \_Period2 >  
bool [std::chrono::operator!=](#) (const duration< \_Rep1, \_Period1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1 , typename \_Rep2 , typename \_Period2 >  
common\_type< duration< \_Rep1, \_Period1 >, duration< \_Rep2, \_Period2 > >::type [std::chrono::operator%](#) (const duration< \_Rep1, \_Period1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period , typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep1, typename enable\_if<!\_\_is\_duration< \_Rep2 >::value, \_Rep2 >::type >::type, \_Period > [std::chrono::operator%](#) (const duration< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period , typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep1, \_Rep2 >::type, \_Period > [std::chrono::operator\\*](#) (const duration< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period , typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep2, \_Rep1 >::type, \_Period > [std::chrono::operator\\*](#) (const \_Rep1 &\_\_s, const duration< \_Rep2, \_Period > &\_\_d)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >  
>::type std::chrono::operator+ (const duration< _Rep1, _Period1 > &_lhs,  
const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,  
_Period2 > >::type > std::chrono::operator+ (const time_point< _Clock, _-  
Duration1 > &_lhs, const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >  
time_point< _Clock, typename common_type< duration< _Rep1, _Period1  
>, _Duration2 >::type > std::chrono::operator+ (const duration< _Rep1, _-  
Period1 > &_lhs, const time_point< _Clock, _Duration2 > &_rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2,  
_Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _-  
Duration1 > &_lhs, const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
common_type< _Duration1, _Duration2 >::type std::chrono::operator- (const  
time_point< _Clock, _Duration1 > &_lhs, const time_point< _Clock, _-  
Duration2 > &_rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >  
>::type std::chrono::operator- (const duration< _Rep1, _Period1 > &_lhs,  
const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >  
duration< typename __common_rep_type< _Rep1, typename enable_-  
if<!__is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period >  
std::chrono::operator/ (const duration< _Rep1, _Period > &_d, const _Rep2  
&_s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const  
duration< _Rep1, _Period1 > &_lhs, const duration< _Rep2, _Period2 >  
&_rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator< (const time_point< _Clock, _Duration1 > &_-  
lhs, const time_point< _Clock, _Duration2 > &_rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator< (const duration< _Rep1, _Period1 > &_lhs,  
const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &_lhs,  
const duration< _Rep2, _Period2 > &_rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator<= (const time_point< _Clock, _Duration1 > &_-  
lhs, const time_point< _Clock, _Duration2 > &_rhs)`

- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool std::chrono::operator== (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool std::chrono::operator> (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool std::chrono::operator>= (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _ToDuration, typename _Clock, typename _Duration >`  
`enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _ToDuration > >::type std::chrono::time_point_cast (const time_point< _Clock, _Duration > &__t)`

### 6.59.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

## 6.60 cinttypes File Reference

### 6.60.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

## 6.61 cinttypes File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`

- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

## 6.62 `cinttypes` File Reference

### 6.62.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cinttypes](#).

## 6.63 `ciso646` File Reference

### 6.63.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ciso646](#).

## 6.64 `climits` File Reference

### Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.64.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [climits](#).

## 6.65 `climits` File Reference

### Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.65.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).

## 6.66 `clocale` File Reference

### Namespaces

- namespace [std](#)

### 6.66.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [clocale](#).

## 6.67 cmath File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp > _Tp std::__cmath_power (_Tp, unsigned int)`
- `template<typename _Tp > _Tp std::__pow_helper (_Tp __x, int __n)`
- `float std::abs (float __x)`
- `long double std::abs (long double __x)`
- `double std::abs (double __x)`
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::abs (_Tp __x)`
- `long double std::acos (long double __x)`
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acos (_Tp __x)`
- `float std::acos (float __x)`
- `float std::asin (float __x)`
- `long double std::asin (long double __x)`
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asin (_Tp __x)`
- `float std::atan (float __x)`
- `long double std::atan (long double __x)`
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atan (_Tp __x)`
- `float std::atan2 (float __y, float __x)`
- `long double std::atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up > __gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value, _Tp >::__type, _Up >::__type std::atan2 (_Tp __y, _Up __x)`
- `float std::ceil (float __x)`
- `long double std::ceil (long double __x)`
- `template<typename _Tp > __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::ceil (_Tp __x)`

- float **std::cos** (float \_\_x)
- long double **std::cos** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::cos** (\_Tp \_\_x)
- float **std::cosh** (float \_\_x)
- long double **std::cosh** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::cosh** (\_Tp \_\_x)
- float **std::exp** (float \_\_x)
- long double **std::exp** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::exp** (\_Tp \_\_x)
- long double **std::fabs** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::fabs** (\_Tp \_\_x)
- float **std::fabs** (float \_\_x)
- float **std::floor** (float \_\_x)
- long double **std::floor** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::floor** (\_Tp \_\_x)
- float **std::fmod** (float \_\_x, float \_\_y)
- long double **std::fmod** (long double \_\_x, long double \_\_y)
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::frexp** (\_Tp \_\_x, int \*\_\_exp)
- float **std::frexp** (float \_\_x, int \*\_\_exp)
- float **std::ldexp** (float \_\_x, int \_\_exp)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::ldexp** (\_Tp \_\_x, int \_\_exp)
- long double **std::ldexp** (long double \_\_x, int \_\_exp)
- float **std::log** (float \_\_x)
- long double **std::log** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::log** (\_Tp \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::log10 (_Tp __x)`
- `float std::log10 (float __x)`
- `long double std::log10 (long double __x)`
- `long double std::modf (long double __x, long double *__iptr)`
- `float std::modf (float __x, float *__iptr)`
- `long double std::pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`  
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`  
`>::__type, _Up >::__type std::pow (_Tp __x, _Up __y)`
- `float std::pow (float __x, float __y)`
- `long double std::sin (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sin (_Tp __x)`
- `float std::sin (float __x)`
- `long double std::sinh (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sinh (_Tp __x)`
- `float std::sinh (float __x)`
- `float std::sqrt (float __x)`
- `long double std::sqrt (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sqrt (_Tp __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::tan (_Tp __x)`
- `long double std::tan (long double __x)`
- `float std::tan (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::tanh (_Tp __x)`
- `long double std::tanh (long double __x)`
- `float std::tanh (float __x)`

### 6.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [`cmath`](#).

## 6.68 `cmath` File Reference

### Namespaces

- namespace [`std`](#)
- namespace [`std::tr1`](#)

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_laguerre` (unsigned int `__n`, unsigned int `__m`, `_Tp __x`)
- float [std::tr1::assoc\\_laguerref](#) (unsigned int `__n`, unsigned int `__m`, float `__x`)
- long double [std::tr1::assoc\\_laguerrel](#) (unsigned int `__n`, unsigned int `__m`, long double `__x`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre` (unsigned int `__l`, unsigned int `__m`, `_Tp __x`)
- float [std::tr1::assoc\\_legendref](#) (unsigned int `__l`, unsigned int `__m`, float `__x`)
- long double [std::tr1::assoc\\_legendrel](#) (unsigned int `__l`, unsigned int `__m`, long double `__x`)
- `template<typename _Tpx , typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (`_Tpx __x`, `_Tpy __y`)
- float [std::tr1::betaf](#) (float `__x`, float `__y`)
- long double [std::tr1::betal](#) (long double `__x`, long double `__y`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1` (`_Tp __k`)
- float [std::tr1::comp\\_ellint\\_1f](#) (float `__k`)
- long double [std::tr1::comp\\_ellint\\_1l](#) (long double `__k`)

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2 (_Tp __k)`
- `float std::tr1::comp\_ellint\_2f (float __k)`
- `long double std::tr1::comp\_ellint\_2l (long double __k)`
- `template<typename _Tp , typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp\_ellint\_3 (_Tp`  
`__k, _Tpn __nu)`
- `float std::tr1::comp\_ellint\_3f (float __k, float __nu)`
- `long double std::tr1::comp\_ellint\_3l (long double __k, long double __nu)`
- `template<typename _Tpa , typename _Tpc , typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf\_hyperg`  
`(_Tpa __a, _Tpc __c, _Tp __x)`
- `float std::tr1::conf\_hypergf (float __a, float __c, float __x)`
- `long double std::tr1::conf\_hypergl (long double __a, long double __c, long`  
`double __x)`
- `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_i (_Tpnu`  
`__nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_if (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_il (long double __nu, long double __x)`
- `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_j (_Tpnu`  
`__nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_jf (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_jl (long double __nu, long double __x)`
- `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_k (_-`  
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_bessel\_kf (float __nu, float __x)`
- `long double std::tr1::cyl\_bessel\_kl (long double __nu, long double __x)`
- `template<typename _Tpnu , typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_neumann (_-`  
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl\_neumannf (float __nu, float __x)`
- `long double std::tr1::cyl\_neumannl (long double __nu, long double __x)`
- `template<typename _Tp , typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint\_1 (_Tp __k, _-`  
`Tpp __phi)`
- `float std::tr1::ellint\_1f (float __k, float __phi)`
- `long double std::tr1::ellint\_1l (long double __k, long double __phi)`
- `template<typename _Tp , typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint\_2 (_Tp __k, _-`  
`Tpp __phi)`
- `float std::tr1::ellint\_2f (float __k, float __phi)`

- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **std::tr1::hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **std::tr1::hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- float **std::tr1::pow** (float \_\_x, float \_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::tr1::pow** (\_Tp \_\_x, \_Up \_\_y)
- double **std::tr1::pow** (double \_\_x, double \_\_y)
- long double **std::tr1::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)

- `template<typename _Tp > __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel` (unsigned int \_\_n, \_Tp \_\_x)
- `float std::tr1::sph_besself` (unsigned int \_\_n, float \_\_x)
- `long double std::tr1::sph_bessell` (unsigned int \_\_n, long double \_\_x)
- `template<typename _Tp > __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- `float std::tr1::sph_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- `long double std::tr1::sph_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- `template<typename _Tp > __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann` (unsigned int \_\_n, \_Tp \_\_x)
- `float std::tr1::sph_neumannf` (unsigned int \_\_n, float \_\_x)
- `long double std::tr1::sph_neumannl` (unsigned int \_\_n, long double \_\_x)

### 6.68.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

## 6.69 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.69.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cmath](#).

## 6.70 `cmath.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >`  
`_Tp std::__cmath_power (_Tp, unsigned int)`

#### 6.70.1 Detailed Description

This is a Standard C++ Library file.

Definition in file [cmath.tcc](#).

## 6.71 `codecvt.h` File Reference

### Classes

- class [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#)  
*Common base for codecvt functions.*
- class [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#)  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class [std::codecvt< char, char, mbstate\\_t >](#)  
*class `codecvt<char, char, mbstate_t>` specialization.*
- class [std::codecvt< wchar\\_t, char, mbstate\\_t >](#)  
*class `codecvt<wchar_t, char, mbstate_t>` specialization.*
- class [std::codecvt\\_base](#)  
*Empty base class for codecvt facet [22.2.1.5].*
- class [std::codecvt\\_byname< \\_InternT, \\_ExternT, \\_StateT >](#)  
*class `codecvt_byname` [22.2.1.6].*

## Namespaces

- namespace `std`

### 6.71.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [`codecvt.h`](#).

## 6.72 `codecvt_specializations.h` File Reference

### Classes

- struct `__gnu_cxx::encoding_char_traits<_CharT>`  
*[`encoding\_char\_traits`](#)*
- class `__gnu_cxx::encoding_state`  
*Extension to use `iconv` for dealing with character encodings.*
- class `std::codecvt<_InternT, _ExternT, encoding_state>`  
*`codecvt<InternT, _ExternT, encoding_state>` specialization.*

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename _Tp>`  
`size_t std::__iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`

### 6.72.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [`codecvt\_specializations.h`](#).

## 6.73 compatibility.h File Reference

### 6.73.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#).

## 6.74 compatibility.h File Reference

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Tp >`  
`bool \_\_gnu\_parallel::\_\_compare\_and\_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `bool \_\_gnu\_parallel::\_\_compare\_and\_swap\_32 (volatile int32_t *__ptr, int32_t __comparand, int32_t __replacement)`
- `bool \_\_gnu\_parallel::\_\_compare\_and\_swap\_64 (volatile int64_t *__ptr, int64_t __comparand, int64_t __replacement)`
- `template<typename _Tp >`  
`_Tp \_\_gnu\_parallel::\_\_fetch\_and\_add (volatile _Tp *__ptr, _Tp __addend)`
- `int32_t \_\_gnu\_parallel::\_\_fetch\_and\_add\_32 (volatile int32_t *__ptr, int32_t __addend)`
- `int64_t \_\_gnu\_parallel::\_\_fetch\_and\_add\_64 (volatile int64_t *__ptr, int64_t __addend)`
- `void \_\_gnu\_parallel::\_\_yield ()`

### 6.74.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/compatibility.h](#).

## 6.75 compiletime\_settings.h File Reference

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### Defines

- [#define \\_GLIBCXX\\_ASSERTIONS](#)
- [#define \\_GLIBCXX\\_CALL\(\\_\\_n\)](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_L1](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_TLB](#)
- [#define \\_GLIBCXX\\_SCALE\\_DOWN\\_FPU](#)
- [#define \\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

### 6.75.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [compiletime\\_settings.h](#).

### 6.75.2 Define Documentation

#### 6.75.2.1 #define \_GLIBCXX\_ASSERTIONS

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

#### 6.75.2.2 #define \_GLIBCXX\_CALL( *n* )

Macro to produce log message when entering a function.

#### Parameters

*n* Input size.

#### See also

[\\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

Definition at line 44 of file `completetime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_tree_unguarded()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_parallel::__parallel_sort_mwms()`.

### 6.75.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `completetime_settings.h`.

### 6.75.2.4 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 74 of file `completetime_settings.h`.

### 6.75.2.5 `#define _GLIBCXX_SCALE_DOWN_FPU`

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `completetime_settings.h`.

### 6.75.2.6 `#define _GLIBCXX_VERBOSE_LEVEL`

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `completetime_settings.h`.

## 6.76 complex File Reference

### Classes

- struct `std::complex<_Tp>`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename _Tp>`  
`_Tp std::__complex_abs (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`_Tp std::__complex_arg (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cos (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cosh (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_exp (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_log (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_pow (const complex<_Tp> &_x, const complex<_Tp> &_y)`
- `template<typename _Tp>`  
`std::complex<_Tp> std::__complex_proj (const std::complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_sin (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_sinh (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_sqrt (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_tan (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_tanh (const complex<_Tp> &_z)`
- `template<typename _Tp>`  
`_Tp std::abs (const complex<_Tp> &)`

- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::imag (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > & __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > & __x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > & __os, const complex< _Tp > & __x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > & __is, complex< _Tp > & __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp >`  
`&)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::proj (_Tp __x)`
- `template<typename _Tp >`  
`_Tp std::real (const complex< _Tp > & __z)`

- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator\* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator\* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator\* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp >`  
`&__y)`
  
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

### 6.76.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

## 6.77 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

## Functions

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::conj` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type >`  
`std::tr1::conj` (\_Tp \_\_x)
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::polar` (const \_Tp &\_\_rho, const \_Up &\_\_theta)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &\_\_x, const  
\_Tp &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const \_Tp &\_\_x, const `std::complex< _Tp`  
> &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &\_\_x, const  
`std::complex< _Tp >` &\_\_y)

### 6.77.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

## 6.78 complex File Reference

### Namespaces

- namespace `std`
- namespace `std::tr1`

### Functions

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_acos` (const `std::complex< _Tp >`  
&\_\_z)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_acosh` (const `std::complex< _Tp >`  
&\_\_z)

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_asin (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_asinh (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_atan (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::__complex_atanh (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up >`  
`&__y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`

### 6.78.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/complex](#).

## 6.79 `complex.h` File Reference

### 6.79.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

## 6.80 `concept_check.h` File Reference

### Defines

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

### 6.80.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concept\\_check.h](#).

## 6.81 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

## Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

## Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

### 6.81.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concurrency.h](#).

## 6.82 cond\_dealtor.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_COND_DEALTOR_CLASS_C_DEC`
- `#define PB_DS_COND_DEALTOR_CLASS_T_DEC`

### 6.82.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond\\_dealtor.hpp](#).

## 6.83 `condition_variable` File Reference

### Classes

- class `std::condition_variable`  
*`condition_variable`*
- class `std::condition_variable_any`  
*`condition_variable_any`*

### Namespaces

- namespace `std`

### Enumerations

- enum `std::cv_status` { `no_timeout`, `timeout` }

#### 6.83.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `condition_variable`.

## 6.84 `constructors_destructor_fn_imps.hpp` File Reference

### Functions

- `PB_DS_CLASS_NAME` ()
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8 >`  
`PB_DS_CLASS_NAME` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 >`  
`PB_DS_CLASS_NAME` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 >`  
`PB_DS_CLASS_NAME` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >`  
`PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >`  
`PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 >`  
`PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3)`
- `template<typename T0 , typename T1 , typename T2 >`  
`PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2)`
- `template<typename T0 , typename T1 >`  
`PB_DS_CLASS_NAME (T0 t0, T1 t1)`
- `template<typename T0 >`  
`PB_DS_CLASS_NAME (T0 t0)`
- `PB_DS_CLASS_NAME (const PB_DS_CLASS_NAME &other)`

### 6.84.1 Detailed Description

Contains `constructors_destructor_fn_imps` applicable to different containers.

Definition in file [constructors\\_destructor\\_fn\\_imps.hpp](#).

## 6.85 `container_base_dispatch.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.85.1 Detailed Description

Contains an associative container dispatching base.

Definition in file [container\\_base\\_dispatch.hpp](#).

## 6.86 `cpp_type_traits.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### 6.86.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpp\\_type\\_traits.h](#).

## 6.87 `cpu_defines.h` File Reference

### 6.87.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpu\\_defines.h](#).

## 6.88 `csetjmp` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define setjmp(env)`

### 6.88.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csetjmp](#).

## 6.89 csignal File Reference

### Namespaces

- namespace [std](#)

#### 6.89.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csignal](#).

## 6.90 cstdarg File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define va_end(ap)`

#### 6.90.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdarg](#).

## 6.91 `cstdarg` File Reference

### 6.91.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

## 6.92 `cstdbool` File Reference

### 6.92.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdbool](#).

## 6.93 `cstdbool` File Reference

### 6.93.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdbool](#).

## 6.94 `cstddef` File Reference

### Namespaces

- namespace [std](#)

### 6.94.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstddef](#).

## 6.95 `cstdint` File Reference

### 6.95.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

## 6.96 `cstdint` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.96.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 6.97 `cstdint` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.97.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdint](#).

## 6.98 `cstdio` File Reference

### Namespaces

- namespace [std](#)

### 6.98.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdio](#).

## 6.99 `cstdio` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.99.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

## 6.100 `cstdio` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.100.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdio](#).

## 6.101 cstdlib File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `EXIT_FAILURE`
- #define `EXIT_SUCCESS`

### Functions

- void `std::abort` (void) `_GLIBCXX_NORETURN` throw ()
- int `std::atexit` (void(\*)()) throw ()
- void `std::exit` (int) `_GLIBCXX_NORETURN` throw ()

### 6.101.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdlib](#).

## 6.102 cstdlib File Reference

### Defines

- #define `_GLIBCXX_BEGIN_NAMESPACE_TR1`
- #define `_GLIBCXX_END_NAMESPACE_TR1`
- #define `_GLIBCXX_TR1`

### 6.102.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

## 6.103 cstdlib File Reference

### 6.103.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdlib](#).

## 6.104 cstring File Reference

### Namespaces

- namespace [std](#)

### Functions

- void \* **std::memchr** (void \*\_\_s, int \_\_c, size\_t \_\_n)
- char \* **std::strchr** (char \*\_\_s, int \_\_n)
- char \* **std::strpbrk** (char \*\_\_s1, const char \*\_\_s2)
- char \* **std::strrchr** (char \*\_\_s, int \_\_n)
- char \* **std::strstr** (char \*\_\_s1, const char \*\_\_s2)

### 6.104.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstring](#).

## 6.105 `ctgmath` File Reference

### 6.105.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).

## 6.106 `ctgmath` File Reference

### 6.106.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

## 6.107 `ctime` File Reference

### Namespaces

- namespace [std](#)

### 6.107.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctime](#).

## 6.108 `ctime` File Reference

### 6.108.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

## 6.109 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)  
*Base class for ctype.*

### Namespaces

- namespace [std](#)

#### 6.109.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_base.h](#).

## 6.110 ctype\_inline.h File Reference

### Namespaces

- namespace [std](#)

#### 6.110.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_inline.h](#).

## 6.111 ctype\_noninline.h File Reference

#### 6.111.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_noninline.h](#).

## 6.112 `cwchar` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `wchar_t * std::wcschr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wspbrk` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wcsrchr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wcsstr` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wmemchr` (`wchar_t *__p`, `wchar_t __c`, `size_t __n`)

#### 6.112.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwchar](#).

## 6.113 `cwchar` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

#### 6.113.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

## 6.114 `cwchar` File Reference

### Namespaces

- namespace `std`
- namespace `std::tr1`

#### 6.114.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `tr1_impl/cwchar`.

## 6.115 `cwctype` File Reference

### Namespaces

- namespace `std`

#### 6.115.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file `cwctype`.

## 6.116 `cwctype` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.116.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

## 6.117 cwctype File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.117.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cwctype](#).

## 6.118 cxxabi-forced.h File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)  
*Thrown as part of forced unwinding.  
A magic placeholder class that can be caught by reference to recognize forced unwinding.*

### 6.118.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi-forced.h](#).

## 6.119 cxxabi.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

## Namespaces

- namespace `__gnu_cxx`
- namespace `abi`

## Defines

- `#define _GLIBCXX_NOTHROW`

## Typedefs

- typedef `__cxa_ctor_return_type(* __cxxabiv1::__cxa_ctor_type)(void *)`

## Functions

- `int __cxxabiv1::__cxa_atexit (void*)(void *), void *, void *) throw ()`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () __attribute__((__pure__)) throw ()`
- `char * __cxxabiv1::__cxa_demangle (const char *__mangled_name, char *__output_buffer, size_t *__length, int *__status)`
- `int __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_guard_abort (__guard *) throw ()`
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) throw ()`
- `void __cxxabiv1::__cxa_pure_virtual (void) __attribute__((__noreturn__))`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void *dest_array, void *src_array, size_t element_count, size_t element_size, __cxa_ctor_return_type(*constructor)(void *, void *), __cxa_ctor_type destructor)`
- `void __cxxabiv1::__cxa_vec_cleanup (void *__array_address, size_t __element_count, size_t __s, __cxa_ctor_type destructor) throw ()`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void *__array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type constructor, __cxa_ctor_type destructor)`
- `void __cxxabiv1::__cxa_vec_delete (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type destructor)`

- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete2** (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type destructor, void(\*\_\_dealloc)(void \*))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete3** (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type destructor, void(\*\_\_dealloc)(void \*, size\_t))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_dtor** (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_ctor\_type destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type constructor, \_\_cxa\_ctor\_type destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new2** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type constructor, \_\_cxa\_ctor\_type destructor, void>(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*))
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new3** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_ctor\_type constructor, \_\_cxa\_ctor\_type destructor, void(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*, size\_t))
  
- void \* **\_\_cxxabiv1::\_\_dynamic\_cast** (const void \*\_\_src\_ptr, const \_\_class\_type\_info \*\_\_src\_type, const \_\_class\_type\_info \*\_\_dst\_type, ptrdiff\_t \_\_src2dst)

### 6.119.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

## 6.120 cxxabi\_tweaks.h File Reference

### Defines

- #define **\_GLIBCXX\_CXA\_VEC\_CTOR\_RETURN(x)**
- #define **\_GLIBCXX\_GUARD\_BIT**
- #define **\_GLIBCXX\_GUARD\_PENDING\_BIT**
- #define **\_GLIBCXX\_GUARD\_SET(x)**
- #define **\_GLIBCXX\_GUARD\_TEST(x)**
- #define **\_GLIBCXX\_GUARD\_WAITING\_BIT**

### Typedefs

- typedef void **\_\_cxxabiv1::\_\_cxa\_ctor\_return\_type**
- typedef void **\_\_cxxabiv1::\_\_cxa\_vec\_ctor\_return\_type**

## Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI_ - _)))`

### 6.120.1 Detailed Description

The header provides an CPU-variable interface to the C++ ABI.

Definition in file [cxxabi\\_tweaks.h](#).

## 6.121 debug.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Defines

- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_subscript(_N)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`
- `#define _GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define _GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT(_Condition)`

### 6.121.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

## 6.122 `debug_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator<\\_Alloc>](#)  
*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator new
  - all deallocation calls operator delete.

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.122.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. You should only include this header if you are using GCC 3 or later.

Definition in file [debug\\_allocator.h](#).

## 6.123 `debug_map_base.hpp` File Reference

### 6.123.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file [debug\\_map\\_base.hpp](#).

## 6.124 `decimal` File Reference

### Classes

- class [std::decimal::decimal128](#)

3.2.4 Class [decimal128](#).

- class [std::decimal::decimal32](#)

3.2.2 Class [decimal32](#).

- class [std::decimal::decimal64](#)

3.2.3 Class [decimal64](#).

## Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

## Defines

- #define [\\_DECLARE\\_DECIMAL128\\_COMPOUND\\_ASSIGNMENT\(\\_Op\)](#)
- #define [\\_DECLARE\\_DECIMAL32\\_COMPOUND\\_ASSIGNMENT\(\\_Op\)](#)
- #define [\\_DECLARE\\_DECIMAL64\\_COMPOUND\\_ASSIGNMENT\(\\_Op\)](#)
- #define [\\_DECLARE\\_DECIMAL\\_BINARY\\_OP\\_WITH\\_DEC\(\\_Op, \\_T1, \\_T2, \\_T3\)](#)
- #define [\\_DECLARE\\_DECIMAL\\_BINARY\\_OP\\_WITH\\_INT\(\\_Op, \\_Tp\)](#)
- #define [\\_DECLARE\\_DECIMAL\\_COMPARISON\(\\_Op, \\_Tp\)](#)
- #define [\\_GLIBCXX\\_USE\\_DECIMAL\\_](#)

## Functions

- double [std::decimal::decimal128\\_to\\_double](#) (decimal128 \_\_d)
- float [std::decimal::decimal128\\_to\\_float](#) (decimal128 \_\_d)
- long double [std::decimal::decimal128\\_to\\_long\\_double](#) (decimal128 \_\_d)
- long long [std::decimal::decimal128\\_to\\_long\\_long](#) (decimal128 \_\_d)
- double [std::decimal::decimal32\\_to\\_double](#) (decimal32 \_\_d)
- float [std::decimal::decimal32\\_to\\_float](#) (decimal32 \_\_d)
- long double [std::decimal::decimal32\\_to\\_long\\_double](#) (decimal32 \_\_d)
- long long [std::decimal::decimal32\\_to\\_long\\_long](#) (decimal32 \_\_d)
- double [std::decimal::decimal64\\_to\\_double](#) (decimal64 \_\_d)
- float [std::decimal::decimal64\\_to\\_float](#) (decimal64 \_\_d)
- long double [std::decimal::decimal64\\_to\\_long\\_double](#) (decimal64 \_\_d)
- long long [std::decimal::decimal64\\_to\\_long\\_long](#) (decimal64 \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) (decimal32 \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) (decimal64 \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) (decimal128 \_\_d)

- float **std::decimal::decimal\_to\_float** (decimal32 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal64 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal128 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal32 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal64 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal128 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal32 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal64 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal128 \_\_d)
- static decimal128 **std::decimal::make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static decimal128 **std::decimal::make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal32 **std::decimal::make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static decimal32 **std::decimal::make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (long long \_\_coeff, int \_\_exp)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)

- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, unsigned long long \_\_-  
rhs)
- decimal32 **std::decimal::operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal32 \_\_-  
rhs)
- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)

- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)

- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)

- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_-  
rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_-  
rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_-  
rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_-  
rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)

- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)

- `bool std::decimal::operator< (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator< (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator< (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal32 __rhs)`

- `bool std::decimal::operator==(decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator==(long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator==(decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator==(int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator==(decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator==(decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator==(decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator==(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator==(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal128 __rhs)`

- bool **std::decimal::operator**> (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**> (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**> (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)

- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>=` (long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (long long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, decimal64 \_\_rhs)

### 6.124.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

## 6.125 deque File Reference

### 6.125.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

## 6.126 deque File Reference

### Classes

- class [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::\_\_debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

#### 6.126.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

## 6.127 deque File Reference

### Classes

- class [std::\\_\\_profile::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

#### 6.127.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).

## 6.128 deque.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _`  
`Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

### 6.128.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [deque.tcc](#).

## 6.129 enc\_filebuf.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#)  
*class [enc\\_filebuf](#).*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.129.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc\\_filebuf.h](#).

## 6.130 [equally\\_split.h](#) File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _DifferenceType , typename _OutputIterator >  
_OutputIterator \_\_gnu\_parallel::equally\_split (_DifferenceType __n, \_-  
ThreadIndex __num_threads, \_OutputIterator __s)`
- `template<typename _DifferenceType >  
_DifferenceType \_\_gnu\_parallel::equally\_split\_point (_DifferenceType __n, \_-  
ThreadIndex __num_threads, \_ThreadIndex __thread_no)`

### 6.130.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [equally\\_split.h](#).

## 6.131 [error\\_constants.h](#) File Reference

### Namespaces

- namespace [std](#)

### Enumerations

- enum [errc](#) {

---

```
address_family_not_supported, address_in_use, address_not_available,  
already_connected,  
argument_list_too_long, argument_out_of_domain, bad_address, bad_  
file_descriptor,  
bad_message, broken_pipe, connection_aborted, connection_already_in_  
progress,  
connection_refused, connection_reset, cross_device_link, destination_  
address_required,  
device_or_resource_busy, directory_not_empty, executable_format_error,  
file_exists,  
file_too_large, filename_too_long, function_not_supported, host_  
unreachable,  
identifier_removed, illegal_byte_sequence, inappropriate_io_control_  
operation, interrupted,  
invalid_argument, invalid_seek, io_error, is_a_directory,  
message_size, network_down, network_reset, network_unreachable,  
no_buffer_space, no_child_process, no_link, no_lock_available,  
no_message_available, no_message, no_protocol_option, no_space_on_  
device,  
no_stream_resources, no_such_device_or_address, no_such_device, no_  
such_file_or_directory,  
no_such_process, not_a_directory, not_a_socket, not_a_stream,  
not_connected, not_enough_memory, not_supported, operation_canceled,  
operation_in_progress, operation_not_permitted, operation_not_  
supported, operation_would_block,  
owner_dead, permission_denied, protocol_error, protocol_not_supported,  
read_only_file_system, resource_deadlock_would_occur, resource_  
unavailable_try_again, result_out_of_range,  
state_not_recoverable, stream_timeout, text_file_busy, timed_out,  
too_many_files_open_in_system, too_many_files_open, too_many_links,  
too_many_symbolic_link_levels,  
value_too_large, wrong_protocol_type }
```

### 6.131.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [error\\_constants.h](#).

## 6.132 exception File Reference

### Classes

- class [std::bad\\_exception](#)
- class [std::exception](#)

*Base class for all library exceptions.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Typedefs

- typedef void(\* [std::terminate\\_handler](#) )()
- typedef void(\* [std::unexpected\\_handler](#) )()

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- terminate\_handler [std::set\\_terminate](#) (terminate\_handler) throw ()
- unexpected\_handler [std::set\\_unexpected](#) (unexpected\_handler) throw ()
- void [std::terminate](#) () \_\_attribute\_\_((\_\_noreturn\_\_)) throw ()
- bool [std::uncaught\\_exception](#) () \_\_attribute\_\_((\_\_pure\_\_)) throw ()
- void [std::unexpected](#) () \_\_attribute\_\_((\_\_noreturn\_\_))

#### 6.132.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

## 6.133 exception.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Functions

- void `__gnu_pbds::__throw_container_error` (void)
- void `__gnu_pbds::__throw_insert_error` (void)
- void `__gnu_pbds::__throw_join_error` (void)
- void `__gnu_pbds::__throw_resize_error` (void)

### 6.133.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

## 6.134 `exception_ptr.h` File Reference

### Classes

- class `std::__exception_ptr::exception_ptr`  
*An opaque pointer to an arbitrary exception.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Ex >`  
`exception_ptr` `std::copy_exception` (`_Ex __ex`) `throw ()`
- `exception_ptr` `std::current_exception` () `throw ()`
- `bool` `std::__exception_ptr::operator!=` (`const exception_ptr &`, `const exception_ptr &`) `__attribute__((__pure__))` `throw ()`
- `bool` `std::__exception_ptr::operator==` (`const exception_ptr &`, `const exception_ptr &`) `__attribute__((__pure__))` `throw ()`
- `void` `std::rethrow_exception` (`exception_ptr`) `__attribute__((__noreturn__))`

### 6.134.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [exception\\_ptr.h](#).

## 6.135 extptr\_allocator.h File Reference

### Classes

- class `__gnu_cxx::_ExtPtr_allocator<_Tp>`

*An example allocator which uses a non-standard pointer type.*

*This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Tp >`  
`void __gnu_cxx::swap (_ExtPtr_allocator<_Tp > &__larg, _ExtPtr_allocator<_Tp > &__rarg)`

#### 6.135.1 Detailed Description

##### Author

Bob Walters

An example allocator which uses an alternative pointer type from `bits/pointer.h`. Supports test cases which confirm container support for alternative pointers.

Definition in file `extptr_allocator.h`.

## 6.136 features.h File Reference

Defines on whether to include algorithm variants.

### Defines

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`

- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 6.136.1 Detailed Description

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [features.h](#).

### 6.136.2 Define Documentation

#### 6.136.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

##### See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file [features.h](#).

#### 6.136.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

##### See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file [features.h](#).

#### 6.136.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

##### See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file [features.h](#).

**6.136.2.4 #define \_GLIBCXX\_FIND\_GROWING\_BLOCKS**

Include the growing blocks variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

**6.136.2.5 #define \_GLIBCXX\_MERGESORT**

Include parallel multi-way mergesort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

**6.136.2.6 #define \_GLIBCXX\_QUICKSORT**

Include parallel unbalanced quicksort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

**6.136.2.7 #define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iiter beg, _Iiter __end)`.

**See also**

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

---

### 6.136.2.8 `#define _GLIBCXX_TREE_FULL_COPY`

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

#### See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

### 6.136.2.9 `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

#### See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

## 6.137 `fenv.h` File Reference

### 6.137.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fenv.h](#).

## 6.138 `find.h` File Reference

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`

### 6.138.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find.h](#).

## 6.139 find\_selectors.h File Reference

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct `__gnu_parallel::__adjacent_find_selector`  
*Test predicate on two adjacent elements.*
- struct `__gnu_parallel::__find_first_of_selector< _FIterator >`  
*Test predicate on several elements.*
- struct `__gnu_parallel::__find_if_selector`  
*Test predicate on a single element, used for `std::find()` and `std::find_if()`.*

- struct `__gnu_parallel::__generic_find_selector`  
*Base class of all `__gnu_parallel::__find_template` selectors.*
- struct `__gnu_parallel::__mismatch_selector`  
*Test inverted predicate on a single element.*

## Namespaces

- namespace `__gnu_parallel`

### 6.139.1 Detailed Description

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find\\_selectors.h](#).

## 6.140 `for_each.h` File Reference

Main interface for embarrassingly parallel functions.

## Namespaces

- namespace `__gnu_parallel`

## Functions

- `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result >`  
`__gnu_parallel::__for_each_template_random_access` (`_Iter __begin`,  
`_Iter __end`, `_UserOp __user_op`, `_Functionality &__functionality`, `_Red __reduction`, `_Result __reduction_start`, `_Result &__output`, `typename`  
`std::iterator_traits< _Iter >::difference_type __bound`, `_Parallelism __parallelism_tag`)

### 6.140.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like [workstealing.h](#), [par\\_loop.h](#), [omp\\_loop.h](#), and [omp\\_loop\\_static.h](#). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each.h](#).

## 6.141 `for_each_selectors.h` File Reference

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct<\\_BinOp>](#)  
*General reduction, using a binary operator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector<\\_It>](#)  
*`std::accumulate()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector<\\_It>](#)  
*Selector that returns the difference between two adjacent `__elements`.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector<\\_It, \\_Diff>](#)  
*`std::count_if()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector<\\_It, \\_Diff>](#)  
*`std::count()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector<\\_It>](#)  
*`std::fill()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector<\\_It>](#)  
*`std::for_each()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector<\\_It>](#)  
*`std::generate()` selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector<\\_It>](#)  
*Generic `__selector` for embarrassingly parallel functions.*
- struct [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector<\\_It>](#)  
*Selector that just returns the passed iterator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector<\\_It, \\_It2, \\_Tp>](#)

*std::inner\_product()* selector.

- struct `__gnu_parallel::__max_element_reduct<_Compare, _It >`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__gnu_parallel::__min_element_reduct<_Compare, _It >`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp >`  
*std::replace() selector.*
- struct `__gnu_parallel::__replace_selector<_It, _Tp >`  
*std::replace() selector.*
- struct `__gnu_parallel::__transform1_selector<_It >`  
*std::transform() \_\_selector, one input sequence variant.*
- struct `__gnu_parallel::__transform2_selector<_It >`  
*std::transform() \_\_selector, two input sequences variant.*
- struct `__gnu_parallel::__DummyReduct`  
*Reduction function doing nothing.*
- struct `__gnu_parallel::__Nothing`  
*Functor doing nothing.*

## Namespaces

- namespace `__gnu_parallel`

### 6.141.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each\\_selectors.h](#).

## 6.142 formatter.h File Reference

### Classes

- struct [\\_\\_gnu\\_debug::\\_\\_is\\_same<\\_Type1, \\_Type2 >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Enumerations

- enum [\\_Debug\\_msg\\_id](#) {  
[\\_\\_msg\\_valid\\_range](#), [\\_\\_msg\\_insert\\_singular](#), [\\_\\_msg\\_insert\\_different](#), [\\_\\_msg\\_erase\\_bad](#),  
[\\_\\_msg\\_erase\\_different](#), [\\_\\_msg\\_subscript\\_oob](#), [\\_\\_msg\\_empty](#), [\\_\\_msg\\_unpartitioned](#),  
[\\_\\_msg\\_unpartitioned\\_pred](#), [\\_\\_msg\\_unsorted](#), [\\_\\_msg\\_unsorted\\_pred](#), [\\_\\_msg\\_not\\_heap](#),  
[\\_\\_msg\\_not\\_heap\\_pred](#), [\\_\\_msg\\_bad\\_bitset\\_write](#), [\\_\\_msg\\_bad\\_bitset\\_read](#),  
[\\_\\_msg\\_bad\\_bitset\\_flip](#),  
[\\_\\_msg\\_self\\_splice](#), [\\_\\_msg\\_splice\\_alloc](#), [\\_\\_msg\\_splice\\_bad](#), [\\_\\_msg\\_splice\\_other](#),  
[\\_\\_msg\\_splice\\_overlap](#), [\\_\\_msg\\_init\\_singular](#), [\\_\\_msg\\_init\\_copy\\_singular](#), [\\_\\_msg\\_init\\_const\\_singular](#),  
[\\_\\_msg\\_copy\\_singular](#), [\\_\\_msg\\_bad\\_deref](#), [\\_\\_msg\\_bad\\_inc](#), [\\_\\_msg\\_bad\\_dec](#),  
[\\_\\_msg\\_iter\\_subscript\\_oob](#), [\\_\\_msg\\_advance\\_oob](#), [\\_\\_msg\\_retreat\\_oob](#), [\\_\\_msg\\_iter\\_compare\\_bad](#),  
[\\_\\_msg\\_compare\\_different](#), [\\_\\_msg\\_iter\\_order\\_bad](#), [\\_\\_msg\\_order\\_different](#),  
[\\_\\_msg\\_distance\\_bad](#),  
[\\_\\_msg\\_distance\\_different](#), [\\_\\_msg\\_deref\\_istream](#), [\\_\\_msg\\_inc\\_istream](#), [\\_\\_msg\\_output\\_ostream](#),  
[\\_\\_msg\\_deref\\_istreambuf](#), [\\_\\_msg\\_inc\\_istreambuf](#) }

#### 6.142.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

## 6.143 forward\_list.h File Reference

### Classes

- struct `std::_Fwd_list_base<_Tp, _Alloc >`  
*Base class for forward\_list.*
- struct `std::_Fwd_list_const_iterator<_Tp >`  
*A forward\_list::const\_iterator.*
- struct `std::_Fwd_list_iterator<_Tp >`  
*A forward\_list::iterator.*
- struct `std::_Fwd_list_node<_Tp >`  
*A helper node class for forward\_list. This is just a linked list with a data value in each node. There is a sorting utility method.*
- struct `std::_Fwd_list_node_base`  
*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- class `std::forward_list<_Tp, _Alloc >`  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp >`  
`bool std::operator!=(const _Fwd_list_iterator<_Tp > &__x, const _Fwd_list_const_iterator<_Tp > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!=(const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<(const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`  
`bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`

### 6.143.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.h](#).

## 6.144 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

### 6.144.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.tcc](#).

## 6.145 fstream File Reference

### Classes

- class [std::basic\\_filebuf<\\_CharT, \\_Traits >](#)

*The actual work of input and output (for files). This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.*

- class [std::basic\\_fstream<\\_CharT, \\_Traits >](#)

*Controlling input and output for files. This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*

- class [std::basic\\_ifstream<\\_CharT, \\_Traits >](#)

*Controlling input for files. This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*

- class [std::basic\\_ofstream<\\_CharT, \\_Traits >](#)

*Controlling output for files. This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*

### Namespaces

- namespace [std](#)

#### 6.145.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fstream](#).

## 6.146 `fstream.tcc` File Reference

### Namespaces

- namespace [std](#)

### 6.146.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [fstream.tcc](#).

## 6.147 `functexcept.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_cast** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_exception** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_function\_call** () `__attribute__((__noreturn__))`
- void **std::\_\_throw\_bad\_typeid** (void) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_domain\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_future\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_invalid\_argument** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_ios\_failure** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_length\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_logic\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_out\_of\_range** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_overflow\_error** (const char \*) `__attribute__((__noreturn__-))`
- void **std::\_\_throw\_range\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_runtime\_error** (const char \*) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_system\_error** (int) `__attribute__((__noreturn__))`
- void **std::\_\_throw\_underflow\_error** (const char \*) `__attribute__((__noreturn__-))`

### 6.147.1 Detailed Description

This header provides support for `-fno-exceptions`.

Definition in file [functexcept.h](#).

## 6.148 functional File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant< \\_Tp >](#)
- struct [std::\\_Build\\_index\\_tuple< \\_Num >](#)  
*Builds an [\\_Index\\_tuple](#)<0, 1, 2, ..., [\\_Num-1](#)>.*
- struct [std::\\_Derives\\_from\\_binary\\_function< \\_Tp >](#)  
*Determines if the type [\\_Tp](#) derives from [binary\\_function](#).*
- struct [std::\\_Derives\\_from\\_unary\\_function< \\_Tp >](#)  
*Determines if the type [\\_Tp](#) derives from [unary\\_function](#).*
- class [std::\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [std::\\_Function\\_to\\_function\\_pointer< \\_Tp, \[\\\_IsFunctionType\]\(#\) >](#)  
*Turns a function type into a function pointer type.*
- class [std::\\_Has\\_result\\_type\\_helper< \\_Tp >](#)
- struct [std::\\_Index\\_tuple< \[\\\_Indexes\]\(#\) >](#)
- struct [std::\\_Maybe\\_get\\_result\\_type< \[\\\_Has\\\_result\\\_type\]\(#\), \[\\\_Functor\]\(#\) >](#)  
*If we have found a [result\\_type](#), extract it.*
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \[\\\_Res\]\(#\), \[\\\_ArgTypes\]\(#\) >](#)
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \[\\\_Res\]\(#\), \[\\\_T1\]\(#\) >](#)  
*Derives from [unary\\_function](#), as appropriate.*
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function< \[\\\_Res\]\(#\), \[\\\_T1\]\(#\), \[\\\_T2\]\(#\) >](#)  
*Derives from [binary\\_function](#), as appropriate.*
- struct [std::\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp >](#)
- struct [std::\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp \[\\\_Class\]\(#\)::\\* >](#)
- class [std::\\_Mem\\_fn< \[\\\_Res\]\(#\)\(\[\\\_Class\]\(#\)::\\*\)\(\[\\\_ArgTypes\]\(#\)...\) const >](#)  
*Implementation of `mem_fn` for const member function pointers.*

- class `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`  
*Implementation of `mem_fn` for volatile member function pointers.*
- class `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) >`  
*Implementation of `mem_fn` for member function pointers.*
- class `std::_Mu<_Arg, false, false >`
- class `std::_Mu<_Arg, false, true >`
- class `std::_Mu<_Arg, true, false >`
- class `std::_Mu<reference_wrapper<_Tp >, false, false >`
- struct `std::_Placeholder<_Num >`  
*The type of placeholder objects defined by `libstdc++`.*
- struct `std::_Reference_wrapper_base<_Tp >`
- struct `std::_Safe_tuple_element<__i, _Tuple >`
- struct `std::_Safe_tuple_element_impl<__i, _Tuple, _IsSafe >`
- struct `std::_Safe_tuple_element_impl<__i, _Tuple, false >`
- struct `std::_Weak_result_type<_Functor >`
- struct `std::_Weak_result_type_impl<_Functor >`
- struct `std::_Weak_result_type_impl<_Res(&)(_ArgTypes...) >`  
*Retrieve the result type for a function reference.*
- struct `std::_Weak_result_type_impl<_Res(*)(_ArgTypes...) >`  
*Retrieve the result type for a function pointer.*
- struct `std::_Weak_result_type_impl<_Res(_ArgTypes...) >`  
*Retrieve the result type for a function type.*
- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`  
*Retrieve result type for a const member function pointer.*
- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >`  
*Retrieve result type for a const volatile member function pointer.*
- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile >`  
*Retrieve result type for a volatile member function pointer.*

- struct [std::Weak\\_result\\_type\\_impl<\\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve result type for a member function pointer.*
- class [std::bad\\_function\\_call](#)  
*Exception class thrown when class template function's operator() is called with an empty target.*
- class [std::function<\\_Res\(\\_ArgTypes...\)>](#)  
*Primary class template for std::function.  
Polymorphic function wrapper.*
- struct [std::is\\_bind\\_expression<\\_Tp >](#)  
*Determines if the given type \_Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].*
- struct [std::is\\_bind\\_expression<\\_Bind<\\_Signature > >](#)  
*Class template \_Bind is always a bind expression.*
- struct [std::is\\_bind\\_expression<\\_Bind\\_result<\\_Result, \\_Signature > >](#)  
*Class template \_Bind is always a bind expression.*
- struct [std::is\\_placeholder<\\_Tp >](#)  
*Determines if the given type \_Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct [std::is\\_placeholder<\\_Placeholder<\\_Num > >](#)
- class [std::reference\\_wrapper<\\_Tp >](#)  
*Primary class template for reference\_wrapper.*

## Namespaces

- namespace [std](#)
- namespace [std::placeholders](#)

## Enumerations

- enum [\\_Manager\\_operation](#) { [\\_\\_get\\_type\\_info](#), [\\_\\_get\\_functor\\_ptr](#), [\\_\\_clone\\_functor](#), [\\_\\_destroy\\_functor](#) }

## Functions

- `template<typename _Functor >`  
`_Functor & std::callable_function (_Functor &__f)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > std::callable_function (_Member _-`  
`Class::*&__p)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > std::callable_function (_Member _-`  
`Class::*const &__p)`
- `template<typename _Functor, typename... _ArgTypes>`  
`_Bind< typename _Maybe_wrap_member_pointer< _Functor >::type(_-`  
`ArgTypes...)> std::bind (_Functor __f, _ArgTypes...__args)`
- `template<typename _Result, typename _Functor, typename... _ArgTypes>`  
`_Bind_result< _Result, typename _Maybe_wrap_member_pointer< _Functor`  
`>::type(_ArgTypes...)> std::bind (_Functor __f, _ArgTypes...__args)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (const function< _Res(_Args...)> &__f, _M_clear_type *)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (_M_clear_type *, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (const function< _Res(_Args...)> &__f, _M_clear_type *)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (_M_clear_type *, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>`  
`void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)>`  
`&__y)`
  
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (_Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (const _Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (reference_wrapper< _Tp > __t)`

## Variables

- `enable_if< (!is_member_pointer< _Functor >::value &&!is_function< _-`  
`Functor >::value &&!is_function< typename remove_pointer< _Functor`  
`>::type >::value), typename result_of< _Functor(_Args...)>::type >::type`  
`std::invoke (_Functor &__f, _Args &&...__args)`

### 6.148.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

## 6.149 functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun< \\_Result, \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun< \\_Result, \\_Argument >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun< \\_Result >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project1st< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project2nd< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select1st< \\_Pair >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select2nd< \\_Pair >](#)  
*An SGI extension .*
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose< \\_Operation1, \\_Operation2 >](#)  
*An SGI extension .*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<class _Operation1 , class _Operation2 >`  
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _-`  
`Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`  
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_-`  
`cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const`  
`_Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result`  
`&__val)`
- `template<class _Result >`  
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const`  
`_Result &__val)`
- `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1 (_Ret(_Tp::*__f)(_-`  
`Arg))`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1_ref (_Ret(_-`  
`Tp::*__f)(_Arg))`

### 6.149.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/functional](#).

## 6.150 functional\_hash.h File Reference

### Classes

- struct [std::hash< \\_Tp >](#)  
*Primary class template hash.*
- struct [std::hash< \\_Tp \\* >](#)

*Partial specializations for pointer types.*

## Namespaces

- namespace [std](#)

## Defines

- #define [\\_Cxx\\_hashtable\\_define\\_trivial\\_hash\(\\_Tp\)](#)

### 6.150.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [functional\\_hash.h](#).

## 6.151 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (_Iterator &)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`

- `template<typename _ForwardIterator, typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _-`  
`ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_singular (_Iterator &__x)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Safe_iterator< _Iterator, _-`  
`Sequence > &__x)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Tp *__ptr)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _-`  
`InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _-`  
`InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, _Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _-`  
`InputIterator &, std::\_\_false\_type)`

- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_set\_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT >`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT *__s)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT *__s, const _-`  
`Integer &__n \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _InputIterator >`  
`_InputIterator \_\_gnu\_debug::\_\_check\_valid\_range (const _InputIterator &__-`  
`first, const _InputIterator &__last \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence`  
`> &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _InputIterator &__first, const _-`  
`InputIterator &__last, std::__false_type)`
- `template<typename _Integral >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _Integral &, const _Integral &`  
`, std::__true_type)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux2 (const _InputIterator &, const _-`  
`InputIterator &, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux2 (const _RandomAccessIterator &__-`  
`first, const _RandomAccessIterator &__last, std::random\_access\_iterator\_tag)`

### 6.151.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).

## 6.152 future File Reference

### Classes

- class `std::\_\_basic\_future< \_Res >`  
*Common implementation for [future](#) and [shared\\_future](#).*

- struct `std::__future_base`  
*Base class and enclosing scope.*
- struct `std::__future_base::Ptr<_Res>`  
*A `unique_ptr` based on the instantiating type.*
- struct `std::__future_base::Result<_Res>`  
*Result.*
- struct `std::__future_base::Result<_Res &>`  
*Partial specialization for reference types.*
- struct `std::__future_base::Result<void>`  
*Explicit specialization for void.*
- struct `std::__future_base::Result_base`  
*Base class for results.*
- class `std::__future_base::State`  
*Shared state between a promise and one or more associated futures.*
- class `std::future<_Res>`  
*Primary template for future.*
- class `std::future<_Res &>`  
*Partial specialization for `future<R&>`*
- class `std::future<void>`  
*Explicit specialization for `future<void>`*
- class `std::future_error`  
*Exception type thrown by futures.*
- class `std::packaged_task<_Res(_ArgTypes...)>`  
*`packaged_task`*
- class `std::promise<_Res>`  
*Primary template for promise.*
- class `std::promise<_Res &>`  
*Partial specialization for `promise<R&>`*

- class `std::promise< void >`  
*Explicit specialization for `promise<void>`*
- class `std::shared_future< _Res >`  
*Primary template for `shared_future`.*
- class `std::shared_future< _Res & >`  
*Partial specialization for `shared_future<R&>`*
- class `std::shared_future< void >`  
*Explicit specialization for `shared_future<void>`*

## Namespaces

- namespace `std`

## Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

## Functions

- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::type > std::async (launch __policy, _Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`  
`enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future< decltype(std::declval< _Fn >)(std::declval< _Args >)...> >::type std::async (_Fn &&__fn, _Args &&...__args)`
- `error_code std::make_error_code (future_errc __errc)`
- `error_condition std::make_error_condition (future_errc __errc)`
- `template<typename _Res >`  
`void std::swap (promise< _Res > &__x, promise< _Res > &__y)`
- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`

## Variables

- const error\_category \*const [std::future\\_category](#)

### 6.152.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).

## 6.153 [gslice.h](#) File Reference

### Classes

- class [std::gslice](#)  
*Class defining multi-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

#### 6.153.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice.h](#).

## 6.154 [gslice\\_array.h](#) File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)  
*Reference to multi-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

## Defines

- #define `_DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.154.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice\\_array.h](#).

## 6.155 hash\_fun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string` (const char \* \_\_s)

### 6.155.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_fun.h](#).

## 6.156 hash\_map File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_map< \\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap< \\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator!=(const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator!=(const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator==(const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator==(const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void __gnu_cxx::swap(hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void __gnu_cxx::swap(hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

### 6.156.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_map](#).

## 6.157 hash\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`

- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`
- #define `PB_DS_SIZE_BASE_C_DEC`

### 6.157.1 Detailed Description

Contains hash-related policies.

Definition in file [hash\\_policy.hpp](#).

## 6.158 hash\_set File Reference

### Classes

- class `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Functions

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &_hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &_hs2)`

### 6.158.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_set](#).

## 6.159 hashtable.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum { [\\_S\\_num\\_primes](#) }

## Functions

- unsigned long `__gnu_cxx::__stl_next_prime` (unsigned long `__n`)
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
bool `__gnu_cxx::operator!=` (const hashtable< `_Val, _Key, _HF, _Ex, _Eq, _All` > &`__ht1`, const hashtable< `_Val, _Key, _HF, _Ex, _Eq, _All` > &`__ht2`)
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
bool `__gnu_cxx::operator==` (const hashtable< `_Val, _Key, _HF, _Ex, _Eq, _All` > &`__ht1`, const hashtable< `_Val, _Key, _HF, _Ex, _Eq, _All` > &`__ht2`)
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
void `__gnu_cxx::swap` (hashtable< `_Val, _Key, _HF, _Extract, _EqKey, _All` > &`__ht1`, hashtable< `_Val, _Key, _HF, _Extract, _EqKey, _All` > &`__ht2`)

## Variables

- static const unsigned long `__gnu_cxx::__stl_prime_list` [`_S_num_primes`]

### 6.159.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [backward/hashtable.h](#).

## 6.160 hashtable.h File Reference

### Namespaces

- namespace `std`

### 6.160.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/hashtable.h](#).

## 6.161 `hashtable_policy.h` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<class _Iterator >`  
[std::iterator\\_traits](#)< \_Iterator >::difference\_type [std::\\_\\_detail::\\_\\_distance\\_fw](#) (\_Iterator \_\_first, \_Iterator \_\_last, [std::input\\_iterator\\_tag](#))
- `template<class _Iterator >`  
[std::iterator\\_traits](#)< \_Iterator >::difference\_type [std::\\_\\_detail::\\_\\_distance\\_fw](#) (\_Iterator \_\_first, \_Iterator \_\_last, [std::forward\\_iterator\\_tag](#))
- `template<class _Iterator >`  
[std::iterator\\_traits](#)< \_Iterator >::difference\_type [std::\\_\\_detail::\\_\\_distance\\_fw](#) (\_Iterator \_\_first, \_Iterator \_\_last)
- `template<typename _Value, bool __cache>`  
`bool std::\_\_detail::operator!= (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`  
`bool std::\_\_detail::operator!= (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`  
`bool std::\_\_detail::operator== (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`  
`bool std::\_\_detail::operator== (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`

### Variables

- const unsigned long [std::\\_\\_detail::\\_\\_prime\\_list](#) []

#### 6.161.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [hashtable\\_policy.h](#).

## 6.162 indirect\_array.h File Reference

### Classes

- class [std::indirect\\_array< \\_Tp >](#)  
*Reference to arbitrary subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define \_DEFINE\_VALARRAY\_OPERATOR\(\_Op, \_Name\)`

#### 6.162.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [indirect\\_array.h](#).

## 6.163 initializer\_list File Reference

### Classes

- class [std::initializer\\_list< \\_E >](#)  
*initializer\_list*

### Namespaces

- namespace [std](#)

#### 6.163.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer\\_list](#).

## 6.164 iomanip File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __-`  
`intl=false)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setprecision __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __-`  
`intl=false)`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

### 6.164.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).

## 6.165 ios File Reference

### 6.165.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

## 6.166 ios\_base.h File Reference

### Classes

- class [std::ios\\_base](#)  
*The base of the I/O class hierarchy.*

*This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).*

- class `std::ios_base::failure`

*These are thrown to indicate problems with io.  
27.4.2.1.1 Class `ios_base::failure`.*

## Namespaces

- namespace `std`

## Defines

- `#define _IOS_BASE_SEEK_CUR`
- `#define _IOS_BASE_SEEK_END`

## Enumerations

- enum `_Ios_Fmtflags` {  
`_S_boolalpha, _S_dec, _S_fixed, _S_hex,`  
`_S_internal, _S_left, _S_oct, _S_right,`  
`_S_scientific, _S_showbase, _S_showpoint, _S_showpos,`  
`_S_skipws, _S_unitbuf, _S_uppercase, _S_adjustfield,`  
`_S_basefield, _S_floatfield, _S_ios_fmtflags_end }`
- enum `_Ios_Iostate` {  
`_S_goodbit, _S_badbit, _S_eofbit, _S_failbit,`  
`_S_ios_iostate_end }`
- enum `_Ios_Openmode` {  
`_S_app, _S_ate, _S_bin, _S_in,`  
`_S_out, _S_trunc, _S_ios_openmode_end }`
- enum `_Ios_Seekdir` { `_S_beg, _S_cur, _S_end, _S_ios_seekdir_end }`

## Functions

- `ios_base & std::boolalpha (ios_base &__base)`
- `ios_base & std::dec (ios_base &__base)`
- `ios_base & std::fixed (ios_base &__base)`

- ios\_base & [std::hex](#) (ios\_base & \_\_base)
- ios\_base & [std::internal](#) (ios\_base & \_\_base)
- ios\_base & [std::left](#) (ios\_base & \_\_base)
- ios\_base & [std::noboolalpha](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowbase](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::noshowpos](#) (ios\_base & \_\_base)
- ios\_base & [std::noskipws](#) (ios\_base & \_\_base)
- ios\_base & [std::nounitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::nouppercase](#) (ios\_base & \_\_base)
- ios\_base & [std::oct](#) (ios\_base & \_\_base)
- `_Ios_Fmtflags` [std::operator&](#) (`_Ios_Fmtflags __a`, `_Ios_Fmtflags __b`)
- `_Ios_Openmode` [std::operator&](#) (`_Ios_Openmode __a`, `_Ios_Openmode __b`)
- `_Ios_Iostate` [std::operator&](#) (`_Ios_Iostate __a`, `_Ios_Iostate __b`)
- `_Ios_Fmtflags` & [std::operator&=](#) (`_Ios_Fmtflags &__a`, `_Ios_Fmtflags __b`)
- `_Ios_Iostate` & [std::operator&=](#) (`_Ios_Iostate &__a`, `_Ios_Iostate __b`)
- `_Ios_Openmode` & [std::operator&=](#) (`_Ios_Openmode &__a`, `_Ios_Openmode __b`)
- `_Ios_Iostate` [std::operator^](#) (`_Ios_Iostate __a`, `_Ios_Iostate __b`)
- `_Ios_Openmode` [std::operator^](#) (`_Ios_Openmode __a`, `_Ios_Openmode __b`)
- `_Ios_Fmtflags` [std::operator^](#) (`_Ios_Fmtflags __a`, `_Ios_Fmtflags __b`)
- `_Ios_Iostate` & [std::operator^=](#) (`_Ios_Iostate &__a`, `_Ios_Iostate __b`)
- `_Ios_Fmtflags` & [std::operator^=](#) (`_Ios_Fmtflags &__a`, `_Ios_Fmtflags __b`)
- `_Ios_Openmode` & [std::operator^=](#) (`_Ios_Openmode &__a`, `_Ios_Openmode __b`)
- `_Ios_Iostate` [std::operator|](#) (`_Ios_Iostate __a`, `_Ios_Iostate __b`)
- `_Ios_Fmtflags` [std::operator|](#) (`_Ios_Fmtflags __a`, `_Ios_Fmtflags __b`)
- `_Ios_Openmode` [std::operator|](#) (`_Ios_Openmode __a`, `_Ios_Openmode __b`)
- `_Ios_Fmtflags` & [std::operator|=](#) (`_Ios_Fmtflags &__a`, `_Ios_Fmtflags __b`)
- `_Ios_Openmode` & [std::operator|=](#) (`_Ios_Openmode &__a`, `_Ios_Openmode __b`)
- `_Ios_Iostate` & [std::operator|=](#) (`_Ios_Iostate &__a`, `_Ios_Iostate __b`)
- `_Ios_Iostate` [std::operator~](#) (`_Ios_Iostate __a`)
- `_Ios_Fmtflags` [std::operator~](#) (`_Ios_Fmtflags __a`)
- `_Ios_Openmode` [std::operator~](#) (`_Ios_Openmode __a`)
- ios\_base & [std::right](#) (ios\_base & \_\_base)
- ios\_base & [std::scientific](#) (ios\_base & \_\_base)
- ios\_base & [std::showbase](#) (ios\_base & \_\_base)
- ios\_base & [std::showpoint](#) (ios\_base & \_\_base)
- ios\_base & [std::showpos](#) (ios\_base & \_\_base)
- ios\_base & [std::skipws](#) (ios\_base & \_\_base)
- ios\_base & [std::unitbuf](#) (ios\_base & \_\_base)
- ios\_base & [std::uppercase](#) (ios\_base & \_\_base)

### 6.166.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ios\\_base.h](#).

## 6.167 iosfwd File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef basic\_filebuf< char > [std::filebuf](#)
- typedef basic\_fstream< char > [std::fstream](#)
- typedef basic\_ifstream< char > [std::ifstream](#)
- typedef basic\_ios< char > [std::ios](#)
- typedef basic\_iostream< char > [std::iostream](#)
- typedef basic\_istream< char > [std::istream](#)
- typedef basic\_istreamstream< char > [std::istreamstream](#)
- typedef basic\_ofstream< char > [std::ofstream](#)
- typedef basic\_ostream< char > [std::ostream](#)
- typedef basic\_ostringstream< char > [std::ostringstream](#)
- typedef basic\_streambuf< char > [std::streambuf](#)
- typedef basic\_stringbuf< char > [std::stringbuf](#)
- typedef basic\_stringstream< char > [std::stringstream](#)
- typedef basic\_filebuf< wchar\_t > [std::wfilebuf](#)
- typedef basic\_fstream< wchar\_t > [std::wfstream](#)
- typedef basic\_ifstream< wchar\_t > [std::wifstream](#)
- typedef basic\_ios< wchar\_t > [std::wios](#)
- typedef basic\_iostream< wchar\_t > [std::wiostream](#)
- typedef basic\_istream< wchar\_t > [std::wistream](#)
- typedef basic\_istreamstream< wchar\_t > [std::wistreamstream](#)
- typedef basic\_ofstream< wchar\_t > [std::wofstream](#)
- typedef basic\_ostream< wchar\_t > [std::wostream](#)
- typedef basic\_ostringstream< wchar\_t > [std::wostringstream](#)
- typedef basic\_streambuf< wchar\_t > [std::wstreambuf](#)
- typedef basic\_stringbuf< wchar\_t > [std::wstringbuf](#)
- typedef basic\_stringstream< wchar\_t > [std::wstringstream](#)

### 6.167.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

## 6.168 iostream File Reference

### Namespaces

- namespace [std](#)

### Variables

- static ios\_base::Init **std::\_\_ioinit**

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

*They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.*

- istream [std::cin](#)
- ostream [std::cout](#)
- ostream [std::cerr](#)
- ostream [std::clog](#)
- wistream [std::wcin](#)
- wostream [std::wcout](#)
- wostream [std::wcerr](#)
- wostream [std::wclog](#)

### 6.168.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).

## 6.169 istream File Reference

### Classes

- class [std::basic\\_iostream<\\_CharT, \\_Traits>](#)

*Merging istream and ostream capabilities.*  
*This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)

*Controlling input.*  
*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.*
- class [std::basic\\_istream<\\_CharT, \\_Traits>::sentry](#)

*Performs setup work for input streams.*

### Namespaces

- namespace [std](#)

### Functions

- [template<typename \\_CharT, typename \\_Traits, typename \\_Tp> basic\\_istream<\\_CharT, \\_Traits> & std::operator>> \(basic\\_istream<\\_CharT, \\_Traits> &&\\_\\_is, \\_Tp &\\_\\_x\)](#)
- [template<typename \\_CharT, typename \\_Traits> basic\\_istream<\\_CharT, \\_Traits> & std::ws \(basic\\_istream<\\_CharT, \\_Traits> &\\_\\_is\)](#)
- [template<typename \\_CharT, typename \\_Traits> basic\\_istream<\\_CharT, \\_Traits> & std::operator>> \(basic\\_istream<\\_CharT, \\_Traits> &\\_\\_in, \\_CharT &\\_\\_c\)](#)
- [template<class \\_Traits> basic\\_istream<char, \\_Traits> & std::operator>> \(basic\\_istream<char, \\_Traits> &\\_\\_in, unsigned char &\\_\\_c\)](#)
- [template<class \\_Traits> basic\\_istream<char, \\_Traits> & std::operator>> \(basic\\_istream<char, \\_Traits> &\\_\\_in, signed char &\\_\\_c\)](#)

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _-`  
`CharT, _Traits > &__in, _CharT *__s`)
- `template<>`  
`basic_istream< char > & std::operator>>` (`basic_istream< char > &__in,`  
`char *__s`)
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>>` (`basic_istream< char, _-`  
`Traits > &__in, unsigned char *__s`)
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>>` (`basic_istream< char, _-`  
`Traits > &__in, signed char *__s`)

### 6.169.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).

## 6.170 istream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws` (`basic_istream< _CharT, _Traits`  
`> &__is`)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _-`  
`CharT, _Traits > &__in, _CharT &__c`)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _-`  
`CharT, _Traits > &__in, _CharT *__s`)

### 6.170.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [istream.tcc](#).

## 6.171 iterator File Reference

### 6.171.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

## 6.172 iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _-  
Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _-  
RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_-  
tag)`
- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _-  
Distance &__n)`

### 6.172.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/iterator](#).

## 6.173 iterator.h File Reference

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple< \\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.173.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [iterator.h](#).

## 6.174 iterator\_tracker.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (type-`  
`name __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const`  
`__iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &_`  
`__rhs)`

### 6.174.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator\\_tracker.h](#).

## 6.175 limits File Reference

### Classes

- struct `std::__numeric_limits_base`  
*Part of `std::numeric_limits`.*
- struct `std::numeric_limits< _Tp >`  
*Properties of fundamental types.*
- struct `std::numeric_limits< bool >`  
*`numeric_limits<bool>` specialization.*
- struct `std::numeric_limits< char >`  
*`numeric_limits<char>` specialization.*
- struct `std::numeric_limits< char16_t >`  
*`numeric_limits<char16_t>` specialization.*
- struct `std::numeric_limits< char32_t >`  
*`numeric_limits<char32_t>` specialization.*
- struct `std::numeric_limits< double >`  
*`numeric_limits<double>` specialization.*
- struct `std::numeric_limits< float >`  
*`numeric_limits<float>` specialization.*
- struct `std::numeric_limits< int >`  
*`numeric_limits<int>` specialization.*

- struct `std::numeric_limits< long >`  
*numeric\_limits<long> specialization.*
- struct `std::numeric_limits< long double >`  
*numeric\_limits<long double> specialization.*
- struct `std::numeric_limits< long long >`  
*numeric\_limits<long long> specialization.*
- struct `std::numeric_limits< short >`  
*numeric\_limits<short> specialization.*
- struct `std::numeric_limits< signed char >`  
*numeric\_limits<signed char> specialization.*
- struct `std::numeric_limits< unsigned char >`  
*numeric\_limits<unsigned char> specialization.*
- struct `std::numeric_limits< unsigned int >`  
*numeric\_limits<unsigned int> specialization.*
- struct `std::numeric_limits< unsigned long >`  
*numeric\_limits<unsigned long> specialization.*
- struct `std::numeric_limits< unsigned long long >`  
*numeric\_limits<unsigned long long> specialization.*
- struct `std::numeric_limits< unsigned short >`  
*numeric\_limits<unsigned short> specialization.*
- struct `std::numeric_limits< wchar_t >`  
*numeric\_limits<wchar\_t> specialization.*

## Namespaces

- namespace `std`

## Defines

- #define `__glibcxx_digits(T)`
- #define `__glibcxx_digits10(T)`
- #define `__glibcxx_double_has_denorm_loss`
- #define `__glibcxx_double_tinyness_before`
- #define `__glibcxx_double_traps`
- #define `__glibcxx_float_has_denorm_loss`
- #define `__glibcxx_float_tinyness_before`
- #define `__glibcxx_float_traps`
- #define `__glibcxx_integral_traps`
- #define `__glibcxx_long_double_has_denorm_loss`
- #define `__glibcxx_long_double_tinyness_before`
- #define `__glibcxx_long_double_traps`
- #define `__glibcxx_max(T)`
- #define `__glibcxx_max_digits10(T)`
- #define `__glibcxx_min(T)`
- #define `__glibcxx_signed(T)`

## Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` {  
`std::round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`,  
`std::round_toward_infinity`,  
`std::round_toward_neg_infinity` }

### 6.175.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

## 6.176 list File Reference

### 6.176.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).

## 6.177 list File Reference

### Classes

- class [std::\\_\\_debug::list< \\_Tp, \\_Allocator >](#)  
*Class [std::list](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

#### 6.177.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

## 6.178 list File Reference

### Classes

- class [std::\\_\\_profile::list< \\_Tp, \\_Allocator >](#)  
*List wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

#### 6.178.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

## 6.179 list.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.179.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [list.tcc](#).

## 6.180 list\_partition.h File Reference

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter >`  
`void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`  
`void \_\_gnu\_parallel::\_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t \_\_gnu\_parallel::list\_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`

### 6.180.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [list\\_partition.h](#).

## 6.181 list\_update\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`
- #define `PB_DS_CLASS_T_DEC`

#### 6.181.1 Detailed Description

Contains policies for list update containers.

Definition in file [list\\_update\\_policy.hpp](#).

## 6.182 locale File Reference

#### 6.182.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

## 6.183 locale\_classes.h File Reference

### Classes

- class [std::collate<\\_CharT>](#)  
*Facet for localized string comparison.*
- class [std::collate\\_byname<\\_CharT>](#)  
*class [collate\\_byname](#) [22.2.4.2].*
- class [std::locale](#)

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

- class [std::locale::facet](#)

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class [std::locale::id](#)

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >  
bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >  
const _Facet & std::use\_facet (const locale &__loc)`

### 6.183.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_classes.h](#).

## 6.184 locale\_classes.tcc File Reference

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >`  
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use_facet (const locale &__loc)`

### 6.184.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `locale_classes.tcc`.

## 6.185 locale\_facets.h File Reference

### Classes

- class `std::__ctype_abstract_base< _CharT >`  
*Common base for ctype facet.*
- class `std::ctype< _CharT >`  
*Primary class template ctype facet.*  
*This template class defines classification and conversion functions for character sets.*  
*It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class `std::ctype< char >`  
*The ctype<char> specialization.*  
*This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class `std::ctype< wchar_t >`  
*The ctype<wchar\_t> specialization.*  
*This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- class `std::ctype_byname< _CharT >`  
*class ctype\_byname [22.2.1.2].*
- class `std::ctype_byname< char >`  
*22.2.1.4 Class ctype\_byname specializations.*

- class `std::num_get<_CharT, _InIter >`  
*Primary class template `num_get`.  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class `std::num_put<_CharT, _OutIter >`  
*Primary class template `num_put`.  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- class `std::numpunct<_CharT >`  
*Primary class template `numpunct`.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.*
- class `std::numpunct_byname<_CharT >`  
*class `numpunct_byname` [22.2.3.2].*

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_NUM_FACETS`

## Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-  
gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __-  
c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __-  
c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __-  
c_locale &) throw ()`

- `template<>`  
void **std::\_\_convert\_to\_v** (const char \*, double &, ios\_base::iostate &, const \_\_c\_locale &) throw ()
- `template<typename _CharT >`  
ostreambuf\_iterator< \_CharT > **std::\_\_write** (ostreambuf\_iterator< \_CharT > \_\_s, const \_CharT \*\_\_ws, int \_\_len)
- `template<typename _CharT, typename _OutIter >`  
\_OutIter **std::\_\_write** (\_OutIter \_\_s, const \_CharT \*\_\_ws, int \_\_len)
- `template<typename _CharT >`  
bool **std::isalnum** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isalpha** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::iscntrl** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isdigit** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isgraph** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::islower** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isprint** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::ispunct** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isspace** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isupper** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
bool **std::isxdigit** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
\_CharT **std::tolower** (\_CharT \_\_c, const locale &\_\_loc)
- `template<typename _CharT >`  
\_CharT **std::toupper** (\_CharT \_\_c, const locale &\_\_loc)

### 6.185.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.h](#).

## 6.186 locale\_facets.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-`  
`gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT >`  
`_GLIBCXX_END_LDBL_NAMESPACE int std::__int_to_char (_CharT *__-`  
`bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __-`  
`dec)`
- `_GLIBCXX_PURE bool std::__verify_grouping (const char *__grouping,`  
`size_t __grouping_size, const string &__grouping_tmp) throw ()`

### 6.186.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.tcc](#).

## 6.187 locale\_facets\_nonio.h File Reference

### Classes

- class [std::messages<\\_CharT >](#)  
*Primary class template messages.*  
*This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [std::messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [std::messages\\_byname<\\_CharT >](#)  
*class messages\_byname [22.2.7.2].*
- class [std::money\\_base](#)

*Money format ordering data.*

*This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*

- class `std::money_get< _CharT, _InIter >`

*Primary class template `money_get`.*

*This facet encapsulates the code to parse and return a monetary amount from a string.*

- class `std::money_put< _CharT, _OutIter >`

*Primary class template `money_put`.*

*This facet encapsulates the code to format and output a monetary amount.*

- class `std::moneypunct< _CharT, _Intl >`

*Primary class template `moneypunct`.*

*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*

- class `std::moneypunct_byname< _CharT, _Intl >`

*class `moneypunct_byname` [22.2.6.4].*

- class `std::time_base`

*Time format ordering data.*

*This class provides an enum representing different orderings of time: day, month, and year.*

- class `std::time_get< _CharT, _InIter >`

*Primary class template `time_get`.*

*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*

- class `std::time_get_byname< _CharT, _InIter >`

*class `time_get_byname` [22.2.5.2].*

- class `std::time_put< _CharT, _OutIter >`

*Primary class template `time_put`.*

*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

- class `std::time_put_byname< _CharT, _OutIter >`

*class `time_put_byname` [22.2.5.4].*

## Namespaces

- namespace [std](#)

### 6.187.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.h](#).

## 6.188 locale\_facets\_nonio.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.188.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.tcc](#).

## 6.189 localefwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isctrl (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 6.189.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [localefwd.h](#).

## 6.190 losertree.h File Reference

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class `\_\_gnu\_parallel::\_LoserTree< __stable, _Tp, _Compare >`  
*Stable [\\_LoserTree](#) variant.*
- class `\_\_gnu\_parallel::\_LoserTree< false, _Tp, _Compare >`  
*Unstable [\\_LoserTree](#) variant.*

- class `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`  
*Guarded loser/tournament tree.*
- struct `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`  
*Internal representation of a `_LoserTree` element.*
- class `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`  
*Stable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`  
*Base class of `_LoserTree` implementation using pointers.*
- struct `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`  
*Internal representation of `_LoserTree` `__elements`.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- class `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`  
*Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`  
*Non-Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`  
*Base class for unguarded `_LoserTree` implementation.*

## Namespaces

- namespace `__gnu_parallel`

### 6.190.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

## 6.191 macros.h File Reference

### Defines

- #define `__glibcxx_check_erase`(\_Position)
- #define `__glibcxx_check_erase_range`(\_First, \_Last)
- #define `__glibcxx_check_heap`(\_First, \_Last)
- #define `__glibcxx_check_heap_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_check_insert`(\_Position)
- #define `__glibcxx_check_insert_range`(\_Position, \_First, \_Last)
- #define `__glibcxx_check_nonempty`()
- #define `__glibcxx_check_partitioned_lower`(\_First, \_Last, \_Value)
- #define `__glibcxx_check_partitioned_lower_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_check_partitioned_upper`(\_First, \_Last, \_Value)
- #define `__glibcxx_check_partitioned_upper_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_check_sorted`(\_First, \_Last)
- #define `__glibcxx_check_sorted_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_check_sorted_set`(\_First1, \_Last1, \_First2)
- #define `__glibcxx_check_sorted_set_pred`(\_First1, \_Last1, \_First2, \_Pred)
- #define `__glibcxx_check_string`(\_String)
- #define `__glibcxx_check_string_len`(\_String, \_Len)
- #define `__glibcxx_check_subscript`(\_N)
- #define `__glibcxx_check_valid_range`(\_First, \_Last)
- #define `_GLIBCXX_DEBUG_VERIFY`(\_Condition, \_ErrorMessage)

### 6.191.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

## 6.191.2 Define Documentation

### 6.191.2.1 #define \_\_glibcxx\_check\_erase( *\_Position* )

Verify that we can erase the element referenced by the iterator *\_Position*. We can erase the element if the *\_Position* iterator is dereferenceable and references this sequence.

Definition at line 102 of file macros.h.

### 6.191.2.2 #define \_\_glibcxx\_check\_erase\_range( *\_First*, *\_Last* )

Verify that we can erase the elements in the iterator range [*\_First*, *\_Last*). We can erase the elements if [*\_First*, *\_Last*) is a valid iterator range within this sequence.

Definition at line 116 of file macros.h.

### 6.191.2.3 #define \_\_glibcxx\_check\_heap\_pred( *\_First*, *\_Last*, *\_Pred* )

Verify that the iterator range [*\_First*, *\_Last*) is a heap w.r.t. the predicate *\_Pred*.

Definition at line 229 of file macros.h.

### 6.191.2.4 #define \_\_glibcxx\_check\_insert( *\_Position* )

Verify that we can insert into *\*this* with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end) and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 64 of file macros.h.

### 6.191.2.5 #define \_\_glibcxx\_check\_insert\_range( *\_Position*, *\_First*, *\_Last* )

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into *\*this* with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

## Todo

We would like to be able to check for noninterference of *\_Position* and the range [*\_First*, *\_Last*), but that can't (in general) be done.

Definition at line 87 of file macros.h.

#### 6.191.2.6 #define \_\_glibcxx\_check\_partitioned\_lower( *\_First*, *\_Last*, *\_Value* )

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value*.

Definition at line 177 of file macros.h.

#### 6.191.2.7 #define \_\_glibcxx\_check\_partitioned\_lower\_pred( *\_First*, *\_Last*, *\_Value*, *\_Pred* )

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value* and predicate *\_Pred*.

Definition at line 197 of file macros.h.

#### 6.191.2.8 #define \_\_glibcxx\_check\_partitioned\_upper\_pred( *\_First*, *\_Last*, *\_Value*, *\_Pred* )

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value* and predicate *\_Pred*.

Definition at line 209 of file macros.h.

#### 6.191.2.9 #define \_\_glibcxx\_check\_sorted\_pred( *\_First*, *\_Last*, *\_Pred* )

Verify that the iterator range [*\_First*, *\_Last*) is sorted by the predicate *\_Pred*.

Definition at line 148 of file macros.h.

#### 6.191.2.10 #define \_GLIBCXX\_DEBUG\_VERIFY( *\_Condition*, *\_ErrorMessage* )

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 42 of file macros.h.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::_Safe_iterator< _`

Iterator, \_Sequence >::operator->(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::operator=().

## 6.192 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator< \\_Tp >](#)

*An allocator that uses malloc.*

*This is precisely the allocator defined in the C++ Standard.*

- all allocation calls `malloc`
- all deallocation calls `free`.

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp >`  
`bool \_\_gnu\_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool \_\_gnu\_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

#### 6.192.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc\\_allocator.h](#).

## 6.193 map File Reference

### 6.193.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

## 6.194 map File Reference

### 6.194.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).

## 6.195 map File Reference

### 6.195.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

## 6.196 map.h File Reference

### Classes

- class [std::\\_\\_debug::map<\\_Key, \\_Tp, \\_Compare, \\_Allocator>](#)  
*Class [std::map](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::\_\_debug::operator!= (const map<_Key, _Tp, _Compare, _Allocator > &__lhs, const map<_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::\_\_debug::operator< (const map<_Key, _Tp, _Compare, _Allocator > &__lhs, const map<_Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::\_\_debug::operator<= (const map<_Key, _Tp, _Compare, _Allocator > &__lhs, const map<_Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.196.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

## 6.197 map.h File Reference

### Classes

- class [std::\\_\\_profile::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::map](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<=` (const map< \_Key, \_Tp, \_Compare, \_-  
Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator==` (const map< \_Key, \_Tp, \_Compare, \_-  
Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>` (const map< \_Key, \_Tp, \_Compare, \_Allocator  
> &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>=` (const map< \_Key, \_Tp, \_Compare, \_-  
Allocator > &\_\_lhs, const map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap` (map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs,  
map< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)

### 6.197.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

## 6.198 mask\_array.h File Reference

### Classes

- class [std::mask\\_array< \\_Tp >](#)  
*Reference to selected subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.198.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [mask\\_array.h](#).

## 6.199 memory File Reference

### 6.199.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [memory](#).

## 6.200 memory File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter >  
pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::\_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter __result, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >  
pair< _RandomAccessIter, _ForwardIter > \_\_gnu\_cxx::\_\_uninitialized\_copy\_n (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >  
pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::\_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >  
pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::\_\_uninitialized\_copy\_n\_a (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator<\_Tp >)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a (-`  
`InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`  
`__first, _Size __count, _ForwardIter __result)`

### 6.200.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/memory](#).

## 6.201 merge.h File Reference

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __-`  
`target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &_-`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &_-`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1 , typename _RAIter3 , typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target,`  
`typename std::iterator\_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target,`  
`typename std::iterator\_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`

### 6.201.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

## 6.202 `messages_members.h` File Reference

### Namespaces

- namespace [std](#)

### 6.202.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [messages\\_members.h](#).

## 6.203 `move.h` File Reference

### Classes

- struct [std::identity](#)< `_Tp` >  
*identity*

## Namespaces

- namespace [std](#)

## Defines

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_MOVE(_Tp)`

## Functions

- `template<typename _Tp >  
enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward  
(typename std::identity< _Tp >::type &__t)`
- `template<typename _Tp >  
enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward  
(typename std::identity< _Tp >::type &&__t)`
- `template<typename _Tp >  
enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward  
(typename std::remove\_reference< _Tp >::type &&__t)`
- `template<typename _Tp >  
enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward  
(typename std::identity< _Tp >::type __t)`
- `template<typename _Tp >  
std::remove\_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _Tp >  
void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _Tp, size_t _Nm>  
void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`

### 6.203.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [move.h](#).

## 6.204 mt\_allocator.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_common\\_pool\\_policy](#)< [\\_PoolTp](#), [\\_Thread](#) >

*Policy for shared \_\_pool objects.*

- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class `__gnu_cxx::__mt_alloc_base< _Tp >`  
*Base class for \_Tp dependent member functions.*
- struct `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`  
*Policy for individual \_\_pool objects.*
- class `__gnu_cxx::__pool< false >`  
*Specialization for single thread.*
- class `__gnu_cxx::__pool< true >`  
*Specialization for thread enabled, via gthreads.h.*
- struct `__gnu_cxx::__pool_base`  
*Base class for pool object.*

## Namespaces

- namespace `__gnu_cxx`

## Defines

- `#define __thread_default`

## Typedefs

- `typedef void(* __gnu_cxx::__destroy_handler)(void *)`

## Functions

- `template<typename _Tp, typename _Poolp >`  
`bool __gnu_cxx::operator!=(const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`

- `template<typename _Tp, typename _Poolp >`  
`bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __-`  
`mt_alloc< _Tp, _Poolp > &)`

### 6.204.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt\\_allocator.h](#).

## 6.205 multimap.h File Reference

### Classes

- class `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`  
*Class `std::multimap` with safety/checking/debug instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.205.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

## 6.206 multimap.h File Reference

### Classes

- class [std::\\_\\_profile::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::multimap](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator >`  
`&__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.206.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

## 6.207 multiseq\_selection.h File Reference

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, descending.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- `#define __S(__i)`
- `#define __S(__i)`

## Functions

- `template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator\_traits< typename std::iterator\_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 6.207.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiseq\\_selection.h](#).

## 6.208 multiset.h File Reference

### Classes

- class `std::\_\_debug::multiset< _Key, _Compare, _Allocator >`  
*Class [std::multiset](#) with safety/checking/debug instrumentation.*

## Namespaces

- namespace `std`
- namespace `std::__debug`

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

### 6.208.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).

## 6.209 multiset.h File Reference

### Classes

- class `std::__profile::multiset< _Key, _Compare, _Allocator >`  
*Class `std::multiset` wrapper with performance instrumentation.*

## Namespaces

- namespace `std`
- namespace `std::__profile`

## Functions

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`

### 6.209.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

## 6.210 multiway\_merge.h File Reference

Implementation of sequential and parallel multiway merge.

### Classes

- struct `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __-sentinels, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 3-way merging with `__sentinels` turned off.*

- struct `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 3-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with `__sentinels` turned off.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with `__sentinels` turned off.*

- class `__gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*

- struct `__gnu_parallel::_LoserTreeTraits< _Tp >`

*Traits for determining whether the loser tree should use pointers or copies.*

- struct `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

*Stable sorting functor.*

- struct `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

*Non-`__stable` sorting functor.*

## Namespaces

- namespace `__gnu_parallel`

## Defines

- #define `_GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`
- #define `_GLIBCXX_PARALLEL_LENGTH(__s)`
- #define `_GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- #define `_GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 \_\_gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType > void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp , typename _Compare > _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator\_traits< \_RAIterIterator >::value\_type::first\_type >::value\_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator\_traits< \_RAIterIterator >::value\_type::first\_type >::value\_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType > void \_\_gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare > _RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`

DifferenceTp \_\_length, \_Compare \_\_comp, parallel\_tag \_\_tag=parallel\_tag(0))

- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator _`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , type-`  
`name _DifferenceTp , typename _Splitter , typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::parallel\_multiway\_merge (_RAIterIterator __seqs_`  
`begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _`  
`DifferenceTp __length, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator _`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator _`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator _`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-`  
`name _Compare >`

```

_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _
_seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _
DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

```

- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
**\_\_gnu\_parallel::stable\_multiway\_merge** (\_RAIterPairIterator \_  
\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_RAIterOut \_\_target, \_  
DifferenceTp \_\_length, \_Compare \_\_comp, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
\_RAIterOut **\_\_gnu\_parallel::stable\_multiway\_merge\_sentinels** (\_  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_RAIterOut  
\_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, sampling\_tag \_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
\_RAIterOut **\_\_gnu\_parallel::stable\_multiway\_merge\_sentinels** (\_  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_RAIterOut  
\_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, default\_parallel\_tag  
\_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
\_RAIterOut **\_\_gnu\_parallel::stable\_multiway\_merge\_sentinels** (\_  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_RAIterOut  
\_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, parallel\_tag \_\_  
tag=parallel\_tag(0))
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
\_RAIterOut **\_\_gnu\_parallel::stable\_multiway\_merge\_sentinels** (\_  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_RAIterOut  
\_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, [\\_\\_gnu\\_parallel::exact\\_  
tag](#) \_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
\_RAIterOut **\_\_gnu\_parallel::stable\_multiway\_merge\_sentinels** (\_  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_  
RAIterOut \_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, [\\_\\_gnu\\_  
parallel::sequential\\_tag](#))

### 6.210.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_merge.h](#).

## 6.210.2 Define Documentation

### 6.210.2.1 #define \_GLIBCXX\_PARALLEL\_LENGTH( \_\_s )

Length of a sequence described by a pair of iterators.

Definition at line 53 of file [multiway\\_merge.h](#).

Referenced by `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

## 6.211 multiway\_mergesort.h File Reference

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Piece<\\_DifferenceTp>](#)  
*Subsequence description.*
- struct [\\_\\_gnu\\_parallel::\\_PMWMSortingData<\\_RAIter>](#)  
*Data accessed by all threads.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently<\\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)  
*Split consistently.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently<false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)  
*Split by sampling.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently<>true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)

*Split by exact splitting.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >`  
`void __gnu_parallel::__determine_samples` (`_PMWMSSortingData< _RAIter`  
`> *__sd, _DifferenceTp __num_samples`)
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::parallel_sort_mwms` (`_RAIter __begin, _RAIter __end, _-`  
`Compare __comp, _ThreadIndex __num_threads`)
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::parallel_sort_mwms_pu` (`_PMWMSSortingData< _-`  
`RAIter > *__sd, _Compare &__comp`)

### 6.211.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_mergesort.h](#).

## 6.212 mutex File Reference

### Classes

- struct [std::adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [std::defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [std::lock\\_guard< \\_Mutex >](#)  
*Scoped lock idiom.*
- class [std::mutex](#)  
*mutex*

- struct `std::once_flag`  
*once\_flag*
- class `std::recursive_mutex`  
*recursive\_mutex*
- class `std::recursive_timed_mutex`  
*recursive\_timed\_mutex*
- class `std::timed_mutex`  
*timed\_mutex*
- struct `std::try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `std::unique_lock< _Mutex >`  
*unique\_lock*

## Namespaces

- namespace `std`

## Functions

- mutex & `std::__get_once_mutex ()`
- void `std::__once_proxy ()`
- void `std::__set_once_functor_lock_ptr (unique_lock< mutex > *)`
- template<typename \_Callable, typename... \_Args>  
void `std::call_once` (once\_flag &\_\_once, \_Callable \_\_f, \_Args &&...\_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void `std::lock` (\_L1 &, \_L2 &, \_L3 &...)
- template<typename \_Mutex >  
void `std::swap` (unique\_lock< \_Mutex > &\_\_x, unique\_lock< \_Mutex > &\_\_y)
- template<typename \_Lock1, typename \_Lock2, typename... \_Lock3>  
int `std::try_lock` (\_Lock1 &\_\_l1, \_Lock2 &\_\_l2, \_Lock3 &...\_\_l3)

## Variables

- function< void()> **std::\_\_once\_functor**
- const adopt\_lock\_t **std::adopt\_lock**
- const defer\_lock\_t **std::defer\_lock**
- const try\_to\_lock\_t **std::try\_to\_lock**

### 6.212.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).

## 6.213 nested\_exception.h File Reference

### Classes

- class [std::nested\\_exception](#)  
*Exception class with exception\_ptr data member.*

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Ex >  
const nested\_exception \* **std::\_\_get\_nested\_exception** (const \_Ex &\_\_ex)
- template<typename \_Ex >  
void **std::\_\_throw\_with\_nested** (\_Ex &&, const nested\_exception \*e) **\_\_-**  
attribute\_\_((\_\_noreturn\_\_))
- template<typename \_Ex >  
void **std::\_\_throw\_with\_nested** (\_Ex &&,...) **\_\_attribute\_\_((\_\_noreturn\_\_))**
- void [std::rethrow\\_if\\_nested](#) (const nested\_exception &\_\_ex)
- template<typename \_Ex >  
void [std::rethrow\\_if\\_nested](#) (const \_Ex &\_\_ex)
- template<typename \_Ex >  
void [std::throw\\_with\\_nested](#) (\_Ex \_\_ex)

### 6.213.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [nested\\_exception.h](#).

## 6.214 new File Reference

### Classes

- class [std::bad\\_alloc](#)

*Exception possibly thrown by `new`.*

*`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*

### Namespaces

- namespace [std](#)

### Typedefs

- typedef `void(* std::new\_handler )()`

### Functions

- `new_handler std::set\_new\_handler (new_handler) throw ()`
- `void * operator new (std::size_t) throw (std::bad_alloc)`
- `void * operator new [ ] (std::size_t) throw (std::bad_alloc)`
- `void operator delete (void *) throw ()`
- `void operator delete [ ] (void *) throw ()`
- `void * operator new (std::size_t, const std::nothrow_t &) throw ()`
- `void * operator new [ ] (std::size_t, const std::nothrow_t &) throw ()`
- `void operator delete (void *, const std::nothrow_t &) throw ()`
- `void operator delete [ ] (void *, const std::nothrow_t &) throw ()`
- `void * operator new (std::size_t, void *__p) throw ()`
- `void * operator new [ ] (std::size_t, void *__p) throw ()`
- `void operator delete (void *, void *) throw ()`
- `void operator delete [ ] (void *, void *) throw ()`

## Variables

- `const nothrow_t std::nothrow`

### 6.214.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [http://gcc.gnu.org/onlinedocs/libstdc++/18\\_support/howto.html#4](http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4) for more.

Definition in file [new](#).

### 6.214.2 Function Documentation

#### 6.214.2.1 `void operator delete ( void * ) throw ()`

These are replaceable signatures:

- normal single `new` and `delete` (no arguments, throw `bad_alloc` on error)
- normal array `new` and `delete` (same)
- `nothrow` single `new` and `delete` (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array `new` and `delete` (same)

Placement `new` and `delete` signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.214.2.2 `void operator delete ( void *, const std::nothrow_t & ) throw ()`

These are replaceable signatures:

- normal single `new` and `delete` (no arguments, throw `bad_alloc` on error)
- normal array `new` and `delete` (same)
- `nothrow` single `new` and `delete` (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array `new` and `delete` (same)

Placement `new` and `delete` signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.3 void operator delete ( void \*, void \* ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 107 of file `new`.

**6.214.2.4 void operator delete[] ( void \* ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.5 void operator delete[] ( void \*, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.6 void operator delete[] ( void \*, void \* ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 108 of file `new`.

**6.214.2.7 void\* operator new ( std::size\_t, void \* \_\_p ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 103 of file `new`.

**6.214.2.8 void\* operator new ( std::size\_t, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.9 void\* operator new ( std::size\_t ) throw (std::bad\_alloc)**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.10 void\* operator new[] ( std::size\_t , const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.214.2.11 void\* operator new[] ( std::size\_t ) throw (std::bad\_alloc)**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

### 6.214.2.12 void\* operator new[] ( std::size\_t, void \* \_\_p ) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 104 of file `new`.

## 6.215 new\_allocator.h File Reference

### Classes

- class `__gnu_cxx::new_allocator<_Tp>`  
*An allocator that uses global new, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator new
  - all deallocation calls operator delete.

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`

### 6.215.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new\\_allocator.h](#).

## 6.216 numeric File Reference

### 6.216.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

## 6.217 numeric File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _ForwardIter, typename _Tp >  
void \_\_gnu\_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`

### 6.217.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/numeric](#).

## 6.218 numeric File Reference

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename __RAIter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate_switch (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Tp std::parallel::inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::parallel::partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`

### 6.218.1 Detailed Description

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/numeric](#).

## 6.219 numeric\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define [\\_\\_glibcxx\\_digits](#)(\_Tp)
- #define [\\_\\_glibcxx\\_digits10](#)(\_Tp)
- #define [\\_\\_glibcxx\\_floating](#)(\_Tp, \_Fval, \_Dval, \_LDval)
- #define [\\_\\_glibcxx\\_max](#)(\_Tp)
- #define [\\_\\_glibcxx\\_max\\_digits10](#)(\_Tp)
- #define [\\_\\_glibcxx\\_max\\_exponent10](#)(\_Tp)
- #define [\\_\\_glibcxx\\_min](#)(\_Tp)
- #define [\\_\\_glibcxx\\_signed](#)(\_Tp)

### 6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric\\_traits.h](#).

## 6.220 numericfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Functions

- `template<typename _IIter, typename _Tp, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_IIter, _IIter, _Tp, _Tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_IIter, _IIter, _Tp, _BinaryOper,`  
`_Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _-`  
`BinaryOper, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __-`  
`parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`  
`_Tag2 >`  
`_OIter std::__parallel::__adjacent_difference_switch (_IIter, _IIter, _OIter, _-`  
`BinaryOper, _Tag1, _Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference_switch (_IIter, _IIter, _OIter, _-`  
`BinaryOper, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu-`  
`parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1,`  
`typename BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _-`  
`RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_-`  
`tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism=__gnu_-`  
`parallel::parallel_unbalanced)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2, typename _Tag1, typename _Tag2 >`  
`_Tp std::__parallel::__inner_product_switch (_IIter1, _IIter1, _IIter2, _Tp,`  
`_BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`  
`_Tag2 >`  
`_OIter std::__parallel::__partial_sum_switch (_IIter, _IIter, _OIter, _-`  
`BinaryOper, _Tag1, _Tag2)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__partial_sum_switch (_IIter, _IIter, _OIter, _-`  
`BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_IIter __begin, _IIter __end, _Tp __init, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu-`  
`parallel::Parallelism)`
- `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_IIter __begin, _IIter __end, _Tp __init)`

- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_IIter, _IIter, _Tp, _BinaryOper)`
- `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_IIter __begin, _IIter __end, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_IIter, _IIter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::Parallelism, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::Parallelism, _BinaryOper)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::sequential\_tag, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_IIter, _IIter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_IIter1, _IIter1, _IIter2, _Tp, BinaryFunction1, BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _IIter1, typename _IIter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >
 _Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >
 _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >
 _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >
 _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >
 _OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`

### 6.220.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

## 6.221 omp\_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
 _Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter >::difference_type __bound)`

### 6.221.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop.h](#).

## 6.222 `omp_loop_static.h` File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop\_static  
(_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _-  
Result __base, _Result &__output, typename std::iterator\_traits< _RAIter  
>::difference_type __bound)`

#### 6.222.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop\\_static.h](#).

## 6.223 `os_defines.h` File Reference

### Defines

- `#define \_\_NO\_CTYPE`

#### 6.223.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [os\\_defines.h](#).

## 6.224 ostream File Reference

### Classes

- class `std::basic_ostream<_CharT, _Traits>`  
*Controlling output.*  
*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.*
- class `std::basic_ostream<_CharT, _Traits>::sentry`  
*Performs setup work for output streams.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::endl (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::ends (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::flush (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits, typename _Tp>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &&__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &&__out, char __c)`
- `template<class _Traits>`  
`basic_ostream<char, _Traits> & std::operator<< (basic_ostream<char, _Traits> &&__out, char __c)`

- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

### 6.224.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

## 6.225 ostream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`

### 6.225.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream.tcc](#).

## 6.226 ostream\_insert.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template wostream & std::__ostream_insert (wostream &, const wchar_t *, streamsize)`
- `template ostream & std::__ostream_insert (ostream &, const char *, streamsize)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >  
void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

### 6.226.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream\\_insert.h](#).

## 6.227 par\_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits<_RAIter>::difference_type __bound)`

### 6.227.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [par\\_loop.h](#).

## 6.228 [parallel.h](#) File Reference

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

### 6.228.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel.h](#).

## 6.229 [partial\\_sum.h](#) File Reference

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator\_traits<_Iter>::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator\_traits<_Iter>::difference_type __n)`

### 6.229.1 Detailed Description

Parallel implementation of [std::partial\\_sum\(\)](#), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partial\\_sum.h](#).

## 6.230 partition.h File Reference

Parallel implementation of [std::partition\(\)](#), [std::nth\\_element\(\)](#), and [std::partial\\_sort\(\)](#). This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- `#define \_GLIBCXX\_VOLATILE`

## Functions

- `template<typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`

- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_`  
`partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex`  
`__num_threads)`

### 6.230.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partition.h](#).

### 6.230.2 Define Documentation

#### 6.230.2.1 #define \_GLIBCXX\_VOLATILE

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by `__gnu_parallel::__parallel_partition()`.

## 6.231 pod\_char\_traits.h File Reference

### Classes

- struct `__gnu_cxx::character< V, I, S >`  
*A POD class that serves as a character abstraction class.*
- struct `std::char_traits< __gnu_cxx::character< V, I, S > >`  
*char\_traits<\_\_gnu\_cxx::character> specialization.*

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename V, typename I, typename S >`  
`bool __gnu_cxx::operator< (const character< V, I, S > &lhs, const character<`  
`V, I, S > &rhs)`

- `template<typename V, typename I, typename S >`  
`bool __gnu_cxx::operator== (const character< V, I, S > &lhs, const`  
`character< V, I, S > &rhs)`

### 6.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pod\\_char\\_traits.h](#).

## 6.232 pointer.h File Reference

### Classes

- `struct __gnu_cxx::_Invalid_type`
- `class __gnu_cxx::_Pointer_adapter< _Storage_policy >`
- `class __gnu_cxx::_Relative_pointer_impl< _Tp >`  
*A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.*
- `class __gnu_cxx::_Relative_pointer_impl< const _Tp >`
- `class __gnu_cxx::_Std_pointer_impl< _Tp >`  
*A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.*
- `struct __gnu_cxx::_Unqualified_type< _Tp >`

### Namespaces

- namespace `__gnu_cxx`

### Defines

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_-`  
`SET(OPERATOR)`

### Functions

- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`

- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT , typename _Traits , typename _StoreT >`  
`std::basic\_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator==( _Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp > &__lhs, int __-`  
`rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator==( int __lhs, const _Pointer_adapter< _Tp > &__-`  
`rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>( const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>( const _Pointer_adapter< _Tp > &__lhs, const _-`  
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>( _Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>( const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>=( const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>=( const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>=( _Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>=( const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`

### 6.232.1 Detailed Description

#### Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

Definition in file [pointer.h](#).

## 6.233 `pool_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc< \\_Tp >](#)  
*Allocator using a memory pool with a single lock.*
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)  
*Base class for [\\_\\_pool\\_alloc](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp >`  
`bool \_\_gnu\_cxx::operator!= (const \_\_pool\_alloc< \_Tp > &, const \_\_pool\_alloc< \_Tp > &)`
- `template<typename _Tp >`  
`bool \_\_gnu\_cxx::operator== (const \_\_pool\_alloc< \_Tp > &, const \_\_pool\_alloc< \_Tp > &)`

#### 6.233.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool\\_allocator.h](#).

## 6.234 `postypes.h` File Reference

### Classes

- class `std::fpos< _StateT >`

*Class representing stream positions.*

## Namespaces

- namespace [std](#)

## Typedefs

- typedef long long [std::streamoff](#)
- typedef `fpos< mbstate_t >` [std::streampos](#)
- typedef `ptrdiff_t` [std::streamsize](#)
- typedef `fpos< mbstate_t >` [std::u16streampos](#)
- typedef `fpos< mbstate_t >` [std::u32streampos](#)
- typedef `fpos< mbstate_t >` [std::wstreampos](#)

## Functions

- `template<typename _StateT >`  
`bool std::operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _StateT >`  
`bool std::operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`

### 6.234.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [postypes.h](#).

## 6.235 `priority_queue.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.235.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority\\_queue.hpp](#).

## 6.236 `priority_queue_base_dispatch.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.236.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority\\_queue\\_base\\_dispatch.hpp](#).

## 6.237 `profiler.h` File Reference

Interface of the profiling runtime library.

### Classes

- struct [\\_\\_gnu\\_profile::\\_\\_reentrance\\_guard](#)  
*Reentrance guard.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Defines

- #define [\\_\\_profcxx\\_hashtable\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_construct2](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_destruct2](#)(\_\_x...)

- #define **\_\_profctx\_hashtable\_resize**(\_\_x...)
- #define **\_\_profctx\_is\_invalid**()
- #define **\_\_profctx\_is\_off**()
- #define **\_\_profctx\_is\_on**()
- #define **\_\_profctx\_list\_construct**(\_\_x...)
- #define **\_\_profctx\_list\_construct2**(\_\_x...)
- #define **\_\_profctx\_list\_destruct**(\_\_x...)
- #define **\_\_profctx\_list\_destruct2**(\_\_x...)
- #define **\_\_profctx\_list\_insert**(\_\_x...)
- #define **\_\_profctx\_list\_invalid\_operator**(\_\_x...)
- #define **\_\_profctx\_list\_iterate**(\_\_x...)
- #define **\_\_profctx\_list\_operation**(\_\_x...)
- #define **\_\_profctx\_list\_rewind**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_construct**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_destruct**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_erase**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_find**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_insert**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_invalidate**(\_\_x...)
- #define **\_\_profctx\_map\_to\_unordered\_map\_iterate**(\_\_x...)
- #define **\_\_profctx\_report**()
- #define **\_\_profctx\_turn\_off**()
- #define **\_\_profctx\_turn\_on**()
- #define **\_\_profctx\_vector\_construct**(\_\_x...)
- #define **\_\_profctx\_vector\_construct2**(\_\_x...)
- #define **\_\_profctx\_vector\_destruct**(\_\_x...)
- #define **\_\_profctx\_vector\_destruct2**(\_\_x...)
- #define **\_\_profctx\_vector\_find**(\_\_x...)
- #define **\_\_profctx\_vector\_insert**(\_\_x...)
- #define **\_\_profctx\_vector\_invalid\_operator**(\_\_x...)
- #define **\_\_profctx\_vector\_iterate**(\_\_x...)
- #define **\_\_profctx\_vector\_resize**(\_\_x...)
- #define **\_\_profctx\_vector\_resize2**(\_\_x...)
- #define **\_GLIBCXX\_PROFILE\_DATA**(\_\_name)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_DATA**(\_\_type, \_\_name, \_\_initial\_value...)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA**(\_\_type, \_\_name)
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH**
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH\_ENV\_VAR**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT\_ENV\_VAR**
- #define **\_GLIBCXX\_PROFILE\_MEM\_PER\_DIAGNOSTIC**
- #define **\_GLIBCXX\_PROFILE\_MEM\_PER\_DIAGNOSTIC\_ENV\_VAR**

- #define `_GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- #define `_GLIBCXX_PROFILE_TRACE_ENV_VAR`
- #define `_GLIBCXX_PROFILE_TRACE_PATH_ROOT`

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, size_t, size_t, size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, size_t, size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, size_t, size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, size_t, size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::__trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_insert (const void *, size_t, size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (const void *, size_t)`
- `void __gnu_profile::__trace_list_to_vector_resize (const void *, size_t, size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_construct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_destruct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_erase (const void *, size_t, size_t)`

- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void \*, size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void \*, size\_t, size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void \*, size\_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void \*, size\_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void \*, size\_t, size\_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, size\_t, size\_t)
- void `__gnu_profile::__trace_vector_to_list_construct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_find` (const void \*, size\_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void \*, size\_t, size\_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void \*, size\_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void \*, size\_t, size\_t)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()

### 6.237.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

## 6.238 profiler\_hashtable\_size.h File Reference

Collection of hashtable size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#)  
*Hashtable size instrumentation trace producer.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_construct](#) (const void \*, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_destruct](#) (const void \*, size\_t, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_report](#) (FILE \*\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_resize](#) (const void \*, size\_t, size\_t)

### 6.238.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler\\_hashtable\\_size.h](#).

## 6.239 profiler\_list\_to\_slist.h File Reference

Diagnostics for list to slist.

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_operation](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_report](#) (FILE \*\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_rewind](#) (const void \*)

### 6.239.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler\\_list\\_to\\_slist.h](#).

## 6.240 profiler\_list\_to\_vector.h File Reference

diagnostics for list to vector.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#)  
*A list-to-vector instrumentation line in the object table.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_insert](#) (const void \*, size\_t, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_invalid\\_operator](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_iterate](#) (const void \*, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_report](#) (FILE \*\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_resize](#) (const void \*, size\_t, size\_t)

### 6.240.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler\\_list\\_to\\_vector.h](#).

## 6.241 profiler\_map\_to\_unordered\_map.h File Reference

Diagnostics for map to unordered\_map.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#)  
*A map-to-unordered\_map instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#)  
*A map-to-unordered\_map instrumentation line in the stack table.*
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#)  
*Map-to-unordered\_map instrumentation producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- int [\\_\\_gnu\\_profile::\\_\\_log2](#) (size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_erase\\_cost](#) (size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_find\\_cost](#) (size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_insert\\_cost](#) (size\_t \_\_size)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_erase](#) (const void \*, size\_t, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_find](#) (const void \*, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_insert](#) (const void \*, size\_t, size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_invalidate](#) (const void \*)

- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void \*, size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_report` (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)

### 6.241.1 Detailed Description

Diagnostics for map to unordered\_map.

Definition in file [profiler\\_map\\_to\\_unordered\\_map.h](#).

## 6.242 profiler\_node.h File Reference

Data structures to represent a single profiling event.

### Classes

- class `__gnu_profile::__object_info_base`  
*Base class for a line in the object table.*
- class `__gnu_profile::__stack_hash`  
*Hash function for summary trace using call stack as index.*
- class `__gnu_profile::__stack_info_base< __object_info >`  
*Base class for a line in the stack table.*

### Namespaces

- namespace `__gnu_profile`

### Typedefs

- typedef void \* `__gnu_profile::__instruction_address_t`
- typedef const void \* `__gnu_profile::__object_t`
- typedef std::vector< \_\_instruction\_address\_t > `__gnu_profile::__stack_npt`
- typedef \_\_stack\_npt \* `__gnu_profile::__stack_t`

## Functions

- `__stack_t __gnu_profile::__get_stack ()`
- `__gnu_profile::__size (const __stack_t &__stack)`
- `size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__write (FILE *__f, const __stack_t __stack)`

### 6.242.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler\\_node.h](#).

## 6.243 profiler\_state.h File Reference

Global profiler state.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

### Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `bool __gnu_profile::__turn (__state_type __s)`
- `bool __gnu_profile::__turn_off ()`
- `bool __gnu_profile::__turn_on ()`
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`

### 6.243.1 Detailed Description

Global profiler state.

Definition in file [profiler\\_state.h](#).

## 6.244 profiler\_trace.h File Reference

Diagnostics for container sizes.

### Classes

- class `__gnu_profile::__trace_base< __object_info, __stack_info >`  
*Base class for all trace producers.*
- struct `__gnu_profile::__warning_data`  
*Representation of a warning.*

### Namespaces

- namespace `__gnu_profile`

### Defines

- `#define _GLIBCXX_IMPL_UNORDERED_MAP`

### Typedefs

- typedef `std::vector< __cost_factor * > __gnu_profile::__cost_factor_vector`
- typedef `std::unordered_map< std::string, std::string > __gnu_profile::__env_t`
- typedef `std::vector< __warning_data > __gnu_profile::__warning_vector_t`

### Functions

- `size_t __gnu_profile::__env_to_size_t (const char *__env_var, size_t __default_value)`
- `__cost_factor_vector * & __gnu_profile::__get__cost_factors ()`
- `__env_t & __gnu_profile::__get__env ()`
- `__gnu_cxx::__mutex & __gnu_profile::__get__global_lock ()`
- `__cost_factor & __gnu_profile::__get__list_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_shift_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_erase_cost_factor ()`

- `__cost_factor & __gnu_profile::__get__map_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_insert_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_erase_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_insert_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __gnu_profile::__get__S_hash_func ()`
- `__trace_hashtable_size *& __gnu_profile::__get__S_hashtable_size ()`
- `__trace_list_to_slist *& __gnu_profile::__get__S_list_to_slist ()`
- `__trace_list_to_vector *& __gnu_profile::__get__S_list_to_vector ()`
- `__trace_map2umap *& __gnu_profile::__get__S_map2umap ()`
- `size_t & __gnu_profile::__get__S_max_mem ()`
- `size_t & __gnu_profile::__get__S_max_stack_depth ()`
- `size_t & __gnu_profile::__get__S_max_warn_count ()`
- `const char *& __gnu_profile::__get__S_trace_file_name ()`
- `__trace_vector_size *& __gnu_profile::__get__S_vector_size ()`
- `__trace_vector_to_list *& __gnu_profile::__get__S_vector_to_list ()`
- `int __gnu_profile::__log_magnitude (float __f)`
- `size_t __gnu_profile::__max_mem ()`
- `FILE * __gnu_profile::__open_output_file (const char *__extension)`
- `bool __gnu_profile::__profctx_init (void)`
- `void __gnu_profile::__profctx_init_unconditional ()`
- `void __gnu_profile::__read_cost_factors ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__set_cost_factors ()`
- `void __gnu_profile::__set_max_mem ()`
- `void __gnu_profile::__set_max_stack_trace_depth ()`
- `void __gnu_profile::__set_max_warn_count ()`
- `void __gnu_profile::__set_trace_path ()`
- `size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__trace_hash_func_init ()`
- `void __gnu_profile::__trace_hash_func_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __gnu_profile::__trace_hashtable_size_init ()`
- `void __gnu_profile::__trace_hashtable_size_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __gnu_profile::__trace_list_to_slist_init ()`
- `void __gnu_profile::__trace_list_to_slist_report (FILE *__f, __warning_vector_t &__warnings)`

- void `__gnu_profile::__trace_list_to_vector_init ()`
- void `__gnu_profile::__trace_list_to_vector_report (FILE *_f, __warning_vector_t &__warnings)`
- void `__gnu_profile::__trace_map_to_unordered_map_init ()`
- void `__gnu_profile::__trace_map_to_unordered_map_report (FILE *_f, __warning_vector_t &__warnings)`
- void `__gnu_profile::__trace_vector_size_init ()`
- void `__gnu_profile::__trace_vector_size_report (FILE *, __warning_vector_t &)`
- void `__gnu_profile::__trace_vector_to_list_init ()`
- void `__gnu_profile::__trace_vector_to_list_report (FILE *, __warning_vector_t &)`
- void `__gnu_profile::__write_cost_factors ()`

### 6.244.1 Detailed Description

Diagnostics for container sizes. Data structures to represent profiling traces.

Definition in file [profiler\\_trace.h](#).

## 6.245 profiler\_vector\_size.h File Reference

Collection of vector size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size](#)  
*Hashtable size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void `__gnu_profile::__trace_vector_size_construct (const void *, size_t)`
- void `__gnu_profile::__trace_vector_size_destruct (const void *, size_t, size_t)`
- void `__gnu_profile::__trace_vector_size_init ()`

- void `__gnu_profile::__trace_vector_size_report` (FILE \*, `__warning_vector_t` &)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, `size_t`, `size_t`)

### 6.245.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler\\_vector\\_size.h](#).

## 6.246 profiler\_vector\_to\_list.h File Reference

diagnostics for vector to list.

### Classes

- class `__gnu_profile::__trace_vector_to_list`  
*Vector-to-list instrumentation producer.*
- class `__gnu_profile::__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__gnu_profile::__vector2list_stack_info`  
*A vector-to-list instrumentation line in the stack table.*

### Namespaces

- namespace `__gnu_profile`

### Functions

- void `__gnu_profile::__trace_vector_to_list_construct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_find` (const void \*, `size_t`)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_insert` (const void \*, `size_t`, `size_t`)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void \*, `size_t`)

- void `__gnu_profile::__trace_vector_to_list_report` (FILE \*, `__warning_vector_t` &)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void \*, `size_t`, `size_t`)

### 6.246.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler\\_vector\\_to\\_list.h](#).

## 6.247 queue File Reference

### 6.247.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

## 6.248 queue.h File Reference

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- `#define` [\\_GLIBCXX\\_VOLATILE](#)

### 6.248.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

### 6.248.2 Define Documentation

#### 6.248.2.1 #define \_GLIBCXX\_VOLATILE

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file [queue.h](#).

## 6.249 quicksort.h File Reference

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator\_traits< \_RAIter >::difference\_type \\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qs\\_divide \(\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, typename std::iterator\\_traits< \\_RAIter >::difference\\_type \_\_pivot\_rank, typename std::iterator\\_traits< \\_RAIter >::difference\\_type \_\_num\_samples, \_ThreadIndex \_\_num\_threads\)`

### 6.249.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [quicksort.h](#).

## 6.250 random File Reference

### 6.250.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

## 6.251 random.h File Reference

### Classes

- class [std::bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [std::bernoulli\\_distribution::param\\_type](#)
- class [std::binomial\\_distribution<\\_IntType>](#)  
*A discrete binomial random number distribution.*
- struct [std::binomial\\_distribution<\\_IntType>::param\\_type](#)
- class [std::cauchy\\_distribution<\\_RealType>](#)  
*A *cauchy\_distribution* random number distribution.*
- struct [std::cauchy\\_distribution<\\_RealType>::param\\_type](#)
- class [std::chi\\_squared\\_distribution<\\_RealType>](#)  
*A *chi\_squared\_distribution* random number distribution.*
- struct [std::chi\\_squared\\_distribution<\\_RealType>::param\\_type](#)
- class [std::discard\\_block\\_engine<\\_RandomNumberEngine, \\_\\_p, \\_\\_r>](#)
- class [std::discrete\\_distribution<\\_IntType>](#)  
*A *discrete\_distribution* random number distribution.*
- struct [std::discrete\\_distribution<\\_IntType>::param\\_type](#)
- class [std::exponential\\_distribution<\\_RealType>](#)

*An exponential continuous distribution for random numbers.*

- struct `std::exponential_distribution<_RealType>::param_type`
- class `std::extreme_value_distribution<_RealType>`  
*A extreme\_value\_distribution random number distribution.*
- struct `std::extreme_value_distribution<_RealType>::param_type`
- class `std::fisher_f_distribution<_RealType>`  
*A fisher\_f\_distribution random number distribution.*
- struct `std::fisher_f_distribution<_RealType>::param_type`
- class `std::gamma_distribution<_RealType>`  
*A gamma continuous distribution for random numbers.*
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::geometric_distribution<_IntType>`  
*A discrete geometric random number distribution.*
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`  
*>*
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`  
*A model of a linear congruential random number generator.*
- class `std::lognormal_distribution<_RealType>`  
*A lognormal\_distribution random number distribution.*
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::negative_binomial_distribution<_IntType>`  
*A negative\_binomial\_distribution random number distribution.*
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`  
*A normal continuous distribution for random numbers.*
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`  
*A piecewise\_constant\_distribution random number distribution.*
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`  
*A piecewise\_linear\_distribution random number distribution.*

- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`  
*A discrete Poisson random number distribution.*
- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`  
*The `seed_seq` class generates sequences of seeds for random number generators.*
- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.*
- class `std::student_t_distribution<_RealType>`  
*A `student_t_distribution` random number distribution.*
- struct `std::student_t_distribution<_RealType>::param_type`
- class `std::uniform_int_distribution<_IntType>`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- struct `std::uniform_int_distribution<_IntType>::param_type`
- class `std::uniform_real_distribution<_RealType>`  
*Uniform continuous distribution for random numbers.*
- struct `std::uniform_real_distribution<_RealType>::param_type`
- class `std::weibull_distribution<_RealType>`  
*A `weibull_distribution` random number distribution.*
- struct `std::weibull_distribution<_RealType>::param_type`

## Namespaces

- namespace `std`
- namespace `std::__detail`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` **`std::minstd_rand`**

- typedef linear\_congruential\_engine< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [std::minstd\\_rand0](#)
- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [std::mt19937\\_64](#)
- typedef discard\_block\_engine< ranlux24\_base, 223, 23 > [std::ranlux24](#)
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > [std::ranlux24\\_base](#)
- typedef discard\_block\_engine< ranlux48\_base, 389, 11 > [std::ranlux48](#)
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > [std::ranlux48\\_base](#)

## Functions

- template<typename \_RealType, size\_t \_\_bits, typename \_UniformRandomNumberGenerator > [\\_RealType std::generate\\_canonical](#) (\_UniformRandomNumberGenerator &\_\_g)
- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m> bool [std::operator!=](#) (const [std::linear\\_congruential\\_engine](#)< \_UIntType, \_\_a, \_\_c, \_\_m > &\_\_lhs, const [std::linear\\_congruential\\_engine](#)< \_UIntType, \_\_a, \_\_c, \_\_m > &\_\_rhs)
- template<typename \_UIntType, size\_t \_\_w, size\_t \_\_s, size\_t \_\_r> bool [std::operator!=](#) (const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType, \_\_w, \_\_s, \_\_r > &\_\_lhs, const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType, \_\_w, \_\_s, \_\_r > &\_\_rhs)
- template<typename \_RandomNumberEngine, size\_t \_\_k> bool [std::operator!=](#) (const [std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine, \_\_k > &\_\_lhs, const [std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine, \_\_k > &\_\_rhs)
- template<typename \_IntType > bool [std::operator!=](#) (const [std::geometric\\_distribution](#)< \_IntType > &\_\_d1, const [std::geometric\\_distribution](#)< \_IntType > &\_\_d2)
- template<typename \_RealType > bool [std::operator!=](#) (const [std::normal\\_distribution](#)< \_RealType > &\_\_d1, const [std::normal\\_distribution](#)< \_RealType > &\_\_d2)
- template<typename \_RealType > bool [std::operator!=](#) (const [std::lognormal\\_distribution](#)< \_RealType > &\_\_d1, const [std::lognormal\\_distribution](#)< \_RealType > &\_\_d2)
- template<typename \_IntType > bool [std::operator!=](#) (const [std::negative\\_binomial\\_distribution](#)< \_IntType > &\_\_d1, const [std::negative\\_binomial\\_distribution](#)< \_IntType > &\_\_d2)

- `template<typename _IntType >`  
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const`  
`std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1,`  
`const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1,`  
`const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine<`  
`_RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1,`  
`const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1,`  
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__-`  
`__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, -`  
`_UIntType __f>`  
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, -`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`  
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_-`  
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const`  
`std::discrete_distribution< _IntType > &__d2)`

- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`  
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType >`  
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType >`  
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1,`  
`const std::uniform_real_distribution< _IntType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const`  
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1,`  
`const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`  
`&)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`  
`&)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType >`  
`&)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType`  
`> &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType`  
`> &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _-`  
`RealType > &)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`  
`, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__x)`

- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `bool std::operator== (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`

- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::weibull_distribution<_RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::extreme_value_distribution<_RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::geometric_distribution<_IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::uniform_real_distribution<_RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::cauchy_distribution<_RealType > &)`
- `template<typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > & __is, std::bernoulli_distribution & __x)`

### 6.251.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.h](#).

## 6.252 random.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >  
_OutputIterator std::__detail::__transform (_InputIterator __first, _-  
InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`

- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const lognormal_distribution< _RealType > & __x)`
- `template<typename _RealType >`  
`bool std::operator== (const std::normal_distribution< _RealType > & __d1, const std::normal_distribution< _RealType > & __d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, piecewise_constant_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, discard_block_engine< _RandomNumberEngine, __p, __r > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, student_t_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, gamma_distribution< _RealType > & __x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, shuffle_order_engine< _`  
`RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`  
`, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType,`  
`__a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &`  
`__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType >`  
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType`  
`> &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`  
`istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_`

[istream](#)< \_CharT, \_Traits > &\_\_is, [discrete\\_distribution](#)< \_IntType > &\_\_x)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [subtract\\_with\\_carry\\_engine](#)< \_UIntType, \_\_w, \_\_s, \_\_r > &\_\_x)
- `template<typename _IntType, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &, [std::uniform\\_int\\_distribution](#)< \_IntType > &)
- `template<typename _IntType, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &, [std::geometric\\_distribution](#)< \_IntType > &)
- `template<typename _RealType, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &, [std::weibull\\_distribution](#)< \_RealType > &)
- `template<typename _RealType, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [piecewise\\_linear\\_distribution](#)< \_RealType > &\_\_x)
- `template<typename _RealType, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **std::operator**>> ([std::basic\\_istream](#)< \_CharT, \_Traits > &, [std::cauchy\\_distribution](#)< \_RealType > &)

### 6.252.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.tcc](#).

## 6.253 random\_number.h File Reference

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.253.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random\\_number.h](#).

## 6.254 random\_shuffle.h File Reference

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter >](#)  
*Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator >](#)  
*Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\(\)](#).*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

## Functions

- `template<typename _RAIter, typename _RandomNumberGenerator > void \_\_gnu\_parallel::\_parallel\_random\_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())`

- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter __begin, _RAIter`  
`__end, typename std::iterator_traits< _RAIter >::difference_type __n, _-`  
`ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU< _-`  
`RAIter, _RandomNumberGenerator > *__pus)`
- `template<typename _RandomNumberGenerator >`  
`int __gnu_parallel::__random_number_pow2 (int __logp, _-`  
`RandomNumberGenerator &__rng)`
- `template<typename _Tp >`  
`_Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter`  
`__end, _RandomNumberGenerator &__rng)`

### 6.254.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random\\_shuffle.h](#).

## 6.255 ratio File Reference

### Classes

- struct `std::ratio< _Num, _Den >`  
*Provides compile-time rational arithmetic.*
- struct `std::ratio_add< _R1, _R2 >`  
*ratio\_add*
- struct `std::ratio_divide< _R1, _R2 >`  
*ratio\_divide*
- struct `std::ratio_equal< _R1, _R2 >`  
*ratio\_equal*
- struct `std::ratio_greater< _R1, _R2 >`  
*ratio\_greater*

- struct `std::ratio_greater_equal< _R1, _R2 >`  
*ratio\_greater\_equal*
- struct `std::ratio_less< _R1, _R2 >`  
*ratio\_less*
- struct `std::ratio_less_equal< _R1, _R2 >`  
*ratio\_less\_equal*
- struct `std::ratio_multiply< _R1, _R2 >`  
*ratio\_multiply*
- struct `std::ratio_not_equal< _R1, _R2 >`  
*ratio\_not\_equal*
- struct `std::ratio_subtract< _R1, _R2 >`  
*ratio\_subtract*

## Namespaces

- namespace `std`

## Typedefs

- typedef `ratio< 1, 1000000000000000000 >` **`std::atto`**
- typedef `ratio< 1, 100 >` **`std::centi`**
- typedef `ratio< 10, 1 >` **`std::deca`**
- typedef `ratio< 1, 10 >` **`std::deci`**
- typedef `ratio< 1000000000000000000, 1 >` **`std::exa`**
- typedef `ratio< 1, 1000000000000000000 >` **`std::femto`**
- typedef `ratio< 1000000000, 1 >` **`std::giga`**
- typedef `ratio< 100, 1 >` **`std::hecto`**
- typedef `ratio< 1000, 1 >` **`std::kilo`**
- typedef `ratio< 1000000, 1 >` **`std::mega`**
- typedef `ratio< 1, 1000000 >` **`std::micro`**
- typedef `ratio< 1, 1000 >` **`std::milli`**
- typedef `ratio< 1, 1000000000 >` **`std::nano`**
- typedef `ratio< 1000000000000000000, 1 >` **`std::peta`**
- typedef `ratio< 1, 1000000000000000000 >` **`std::pico`**
- typedef `ratio< 1000000000000000000, 1 >` **`std::tera`**

### 6.255.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

## 6.256 `rb_tree` File Reference

### Classes

- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`

### Namespaces

- namespace `__gnu_cxx`

### 6.256.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rb\\_tree](#).

## 6.257 `rc_string_base.h` File Reference

### Classes

- class `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`

### Namespaces

- namespace `__gnu_cxx`

### 6.257.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [rc\\_string\\_base.h](#).

## 6.258 regex File Reference

### 6.258.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

## 6.259 rope File Reference

### Classes

- class [\\_\\_gnu\\_cxx::rope< \\_CharT, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- #define [\\_\\_GC\\_CONST](#)
- #define [\\_\\_ROPE\\_DEFINE\\_ALLOC\(\\_Tp, \\_\\_name\)](#)
- #define [\\_\\_ROPE\\_DEFINE\\_ALLOC\(\\_Tp, \\_\\_name\)](#)
- #define [\\_\\_ROPE\\_DEFINE\\_ALLOCS\(\\_\\_a\)](#)
- #define [\\_\\_STATIC\\_IF\\_SGI\\_ALLOC](#)
- #define [\\_\\_STL\\_FREE\\_STRING\(\\_\\_s, \\_\\_l, \\_\\_a\)](#)
- #define [\\_\\_STL\\_ROPE\\_FROM\\_UNOWNED\\_CHAR\\_PTR\(\\_\\_s, \\_\\_size, \\_\\_a\)](#)

### Typedefs

- typedef [rope< char > \\_\\_gnu\\_cxx::crope](#)
- typedef [rope< wchar\\_t > \\_\\_gnu\\_cxx::wrope](#)

### Enumerations

- enum [{ \\_S\\_max\\_rope\\_depth }](#)
- enum [\\_Tag { \\_S\\_leaf, \\_S\\_concat, \\_S\\_substringfn, \\_S\\_function }](#)

## Functions

- `crope::reference __gnu_cxx::__mutable_reference_at` (`crope &__c`, `size_t __i`)
- `template<typename _ForwardIterator, typename _Tp >`  
`void __gnu_cxx::__Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `allocator< _Tp >`)
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __gnu_cxx::__Destroy_const` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator __alloc`)
- `template<class _CharT >`  
`void __gnu_cxx::__S_cond_store_eos` (`_CharT &`)
- `void __gnu_cxx::__S_cond_store_eos` (`char &__c`)
- `void __gnu_cxx::__S_cond_store_eos` (`wchar_t &__c`)
- `template<class _CharT >`  
`_CharT __gnu_cxx::__S_eos` (`_CharT *`)
- `bool __gnu_cxx::__S_is_basic_char_type` (`wchar_t *`)
- `template<class _CharT >`  
`bool __gnu_cxx::__S_is_basic_char_type` (`_CharT *`)
- `bool __gnu_cxx::__S_is_basic_char_type` (`char *`)
- `template<class _CharT >`  
`bool __gnu_cxx::__S_is_one_byte_char_type` (`_CharT *`)
- `bool __gnu_cxx::__S_is_one_byte_char_type` (`char *`)
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!=` (`const _Rope_iterator< _CharT, _Alloc > &__x`, `const _Rope_iterator< _CharT, _Alloc > &__y`)
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!=` (`const rope< _CharT, _Alloc > &__x`, `const rope< _CharT, _Alloc > &__y`)
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!=` (`const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x`, `const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y`)
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!=` (`const _Rope_const_iterator< _CharT, _Alloc > &__x`, `const _Rope_const_iterator< _CharT, _Alloc > &__y`)
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+` (`const _Rope_iterator< _CharT, _Alloc > &__x`, `ptrdiff_t __n`)
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+` (`ptrdiff_t __n`, `const _Rope_iterator< _CharT, _Alloc > &__x`)
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+` (`const rope< _CharT, _Alloc > &__left`, `const rope< _CharT, _Alloc > &__right`)

- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`  
`> &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`  
`> &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _-`  
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __-`  
`n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc`  
`> &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_-`  
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &_-`  
`__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _-`  
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const`  
`rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`

([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_o, const rope< \_CharT, \_Alloc > &\_\_r)

- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const`  
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc`  
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const`  
`rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const`  
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const`  
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _`  
`Rope_char_ref_proxy< _CharT, _Alloc > __b)`

- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

## Variables

- `rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

### 6.259.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rope](#).

## 6.260 ropeimpl.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<class _CharT, class _Traits >`  
`void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`  
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `template<class _Rope_iterator >`  
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic\_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> __first, _Rope_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> __middle, _Rope_iterator< char, \_\_STL\_DEFAULT\_ALLOCATOR(char)> __last)`

### 6.260.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ropeimpl.h](#).

## 6.261 `safe_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base](#)  
*Base class that supports tracking of iterators that reference a sequence.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.261.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_base.h](#).

## 6.262 `safe_iterator.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#)  
*Safe iterator wrapper.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

## Functions

- `bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base * __x)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::__operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__operator!= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
_Safe_iterator< _Iterator, _Sequence > __gnu_debug::__operator+ (typename _Safe_iterator< _Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
_Safe_iterator< _IteratorL, _Sequence >::difference_type __gnu_debug::__operator- (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
_Safe_iterator< _Iterator, _Sequence >::difference_type __gnu_debug::__operator- (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::__operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::__operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::__operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::__operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`

### 6.262.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_iterator.h](#).

## 6.263 `safe_iterator.tcc` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.263.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_iterator.tcc](#).

## 6.264 `safe_sequence.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_After\\_nth\\_from< \\_Iterator >](#)
- class [\\_\\_gnu\\_debug::\\_Not\\_equal\\_to< \\_Type >](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >](#)

*Base class for constructing a safe sequence type that tracks iterators that reference it.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.264.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_sequence.h](#).

## 6.265 search.h File Reference

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _DifferenceTp >  
void \_\_gnu\_parallel::\_\_calc\_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >  
\_\_RAIter1 \_\_gnu\_parallel::\_\_search\_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

### 6.265.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [search.h](#).

## 6.266 set File Reference

### 6.266.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

## 6.267 set File Reference

### 6.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

## 6.268 set File Reference

### 6.268.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

## 6.269 set.h File Reference

### Classes

- class [std::\\_\\_debug::set<\\_Key, \\_Compare, \\_Allocator>](#)  
*Class [std::set](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const set<_Key, _Compare, _Allocator > &_  
_lhs, const set<_Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const set<_Key, _Compare, _Allocator > &_  
_lhs, const set<_Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const set<_Key, _Compare, _Allocator >  
&_lhs, const set<_Key, _Compare, _Allocator > &_rhs)`

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator== (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &_`  
`__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _`  
`Key, _Compare, _Allocator > &__y)`

### 6.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

## 6.270 set.h File Reference

### Classes

- class [std::\\_\\_profile::set< \\_Key, \\_Compare, \\_Allocator >](#)  
*Class [std::set](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &_`  
`__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__-`  
`__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator >`  
`&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _-`  
`Key, _Compare, _Allocator > &__y)`

### 6.270.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

## 6.271 set\_operations.h File Reference

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator __gnu_parallel::copy_tail (std::pair< _Iter, _Iter > __b,  
std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::parallel_set_difference (_Iter __begin1,`  
`_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-`  
`Compare __comp)`

- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1,`  
`_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename Operation >`  
`_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1, _-`  
`Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, Operation`  
`__op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter`  
`__begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator _-`  
`result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter`  
`__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare _-`  
`comp)`

### 6.271.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [set\\_operations.h](#).

## 6.272 settings.h File Reference

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Settings](#)  
*class [\\_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

### 6.272.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

### 6.272.2 parallelization\_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and \_\_off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

- force\_sequential, meaning the sequential algorithm is executed.
- force\_parallel, meaning the parallel algorithm is executed.
- heuristic

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file [settings.h](#).

### 6.272.3 Define Documentation

#### 6.272.3.1 #define \_GLIBCXX\_PARALLEL\_CONDITION( \_\_c )

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

#### Parameters

`__c` A condition that is convertible to bool that is overruled by `__gnu_parallel::_Settings::algorithm_strategy`. Usually a decision based on the input size.

Definition at line 95 of file `settings.h`.

## 6.273 `shared_ptr.h` File Reference

### Classes

- class [std::enable\\_shared\\_from\\_this< \\_Tp >](#)  
*Base class allowing use of member function `shared_from_this`.*
- struct [std::owner\\_less< shared\\_ptr< \\_Tp > >](#)  
*Partial specialization of `owner_less` for `shared_ptr`.*
- struct [std::owner\\_less< weak\\_ptr< \\_Tp > >](#)  
*Partial specialization of `owner_less` for `weak_ptr`.*
- class [std::shared\\_ptr< \\_Tp >](#)  
*A smart pointer with reference-counted copy semantics.*
- class [std::weak\\_ptr< \\_Tp >](#)  
*A smart pointer with weak semantics.*

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp > std::allocate\_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::const\_pointer\_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::dynamic\_pointer\_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * std::get\_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _Tp > std::make\_shared (_Args &&...__args)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic\_ostream< _Ch, _Tr > & std::operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::static\_pointer\_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`  
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Tp >`  
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

### 6.273.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared\\_ptr.h](#).

## 6.274 `shared_ptr_base.h` File Reference

### 6.274.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared\\_ptr\\_base.h](#).

## 6.275 `slice_array.h` File Reference

### Classes

- class [std::slice](#)  
*Class defining one-dimensional subset of an array.*
- class [std::slice\\_array< \\_Tp >](#)  
*Reference to one-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.275.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [slice\\_array.h](#).

## 6.276 `slist` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::slist< \\_Tp, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link` (`_Slist_node_base * __prev_node`, `_Slist_node_base * __new_node`)
- `_Slist_node_base * __gnu_cxx::__slist_previous` (`_Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `const _Slist_node_base * __gnu_cxx::__slist_previous` (`const _Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `_Slist_node_base * __gnu_cxx::__slist_reverse` (`_Slist_node_base * __node`)
- `size_t __gnu_cxx::__slist_size` (`_Slist_node_base * __node`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __before_first`, `_Slist_node_base * __before_last`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __head`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator!=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator<` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator<=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator==` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator>` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`bool __gnu_cxx::operator>=` (`const slist< _Tp, _Alloc > &_SL1`, `const slist< _Tp, _Alloc > &_SL2`)
- `template<class _Tp, class _Alloc >`  
`void __gnu_cxx::swap` (`slist< _Tp, _Alloc > &__x`, `slist< _Tp, _Alloc > &__y`)

### 6.276.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [slist](#).

## 6.277 `sort.h` File Reference

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare > void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`

### 6.277.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

## 6.278 sso\_string\_base.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.278.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sso\\_string\\_base.h](#).

## 6.279 sstream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc >](#)  
*Controlling input for std::string.*  
*This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ostream<\\_CharT, \\_Traits, \\_Alloc >](#)  
*Controlling output for std::string.*  
*This class supports writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >](#)  
*The actual work of input and output (for std::string).*  
*This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic\\_string](#). (Paraphrased from [27.7.1]/1.).*
- class [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc >](#)

*Controlling input and output for `std::string`.*

*This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*

## Namespaces

- namespace [std](#)

### 6.279.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [sstream](#).

## 6.280 sstream.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.280.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sstream.tcc](#).

## 6.281 stack File Reference

### 6.281.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stack](#).

## 6.282 `standard_policies.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Enumerations

- enum { `default_store_hash` }

#### 6.282.1 Detailed Description

Contains standard policies for containers.

Definition in file [standard\\_policies.hpp](#).

## 6.283 `stdatomic.h` File Reference

#### 6.283.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdatomic.h](#).

## 6.284 `stdexcept` File Reference

### Classes

- class [std::domain\\_error](#)
- class [std::invalid\\_argument](#)
- class [std::length\\_error](#)
- class [std::logic\\_error](#)

*One of two subclasses of exception.*

- class [std::out\\_of\\_range](#)
- class [std::overflow\\_error](#)
- class [std::range\\_error](#)
- class [std::runtime\\_error](#)

*One of two subclasses of exception.*

- class [std::underflow\\_error](#)

## Namespaces

- namespace [std](#)

### 6.284.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdexcept](#).

## 6.285 `stdio_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits >](#)

*Provides a layer of compatibility for C/POSIX.*

*This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.285.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio\\_filebuf.h](#).

## 6.286 `stdio_sync_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits >](#)

*Provides a layer of compatibility for C.*

*This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

## Namespaces

- namespace `__gnu_cxx`

### 6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `stdio_sync_filebuf.h`.

## 6.287 `stl_algo.h` File Reference

### Namespaces

- namespace `std`

### Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }

### Functions

- `template<typename _RandomAccessIterator, typename _Distance >`  
`void std::__chunk_insertion_sort` (`_RandomAccessIterator __first`, `_-`  
`RandomAccessIterator __last`, `_Distance __chunk_size`)
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void std::__chunk_insertion_sort` (`_RandomAccessIterator __first`, `_-`  
`RandomAccessIterator __last`, `_Distance __chunk_size`, `_Compare __comp`)
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::__copy_n` (`_InputIterator __first`, `_Size __n`, `_-`  
`OutputIterator __result`, `input_iterator_tag`)
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::__copy_n` (`_RandomAccessIterator __first`, `_Size __n`, `_-`  
`OutputIterator __result`, `random_access_iterator_tag`)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__final_insertion_sort` (`_RandomAccessIterator __first`, `_-`  
`RandomAccessIterator __last`, `_Compare __comp`)
- `template<typename _RandomAccessIterator >`  
`void std::__final_insertion_sort` (`_RandomAccessIterator __first`, `_-`  
`RandomAccessIterator __last`)

- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator std::__find (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::__find (_InputIterator __first, _InputIterator __last, const _Tp`  
`&__val, input_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BinaryPredicate >`  
`_BidirectionalIterator1 std::__find_end (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`  
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional _-`  
`iterator_tag, BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::__find_end (_ForwardIterator1 __first1, _-`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`  
`__last2, forward_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::__find_end (_ForwardIterator1 __first1, _-`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`  
`__last2, forward_iterator_tag, forward_iterator_tag, BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 std::__find_end (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`  
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional _-`  
`iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::__find_if (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator std::__find_if (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator std::__find_if_not (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::__find_if_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred, input_iterator_tag)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement std::__gcd (_EuclideanRingElement __m, _-`  
`EuclideanRingElement __n)`
- `template<typename _RandomAccessIterator >`  
`void std::__heap_select (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __middle, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_heap\_select (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare`  
`__comp)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`  
`_ForwardIterator std::\_\_inplace\_stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_inplace\_stable\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_inplace\_stable\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void std::\_\_introslect (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __-`  
`depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::\_\_introslect (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __-`  
`depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void std::\_\_introsort\_loop (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::\_\_introsort\_loop (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`  
`void std::\_\_merge\_adaptive (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`  
`__len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename`  
`_Compare >`  
`void std::\_\_merge\_adaptive (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1,`  
`_Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare`  
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3 >`

- ```

_BidirectionalIterator3 std::\_\_merge\_backward (_BidirectionalIterator1 __-
first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-
BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)

```
- ```

template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _-
BidirectionalIterator3 , typename _Compare >
_BidirectionalIterator3 std::\_\_merge\_backward (_BidirectionalIterator1 __-
first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-
BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __-
comp)

```
  - ```

template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename
_Distance >
void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-
RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance
__step_size)

```
  - ```

template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename
_Distance , typename _Compare >
void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-
RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance
__step_size, _Compare __comp)

```
  - ```

template<typename _RandomAccessIterator , typename _Pointer >
void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Pointer __buffer)

```
  - ```

template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >
void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-
RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)

```
  - ```

template<typename _BidirectionalIterator , typename _Distance >
void std::\_\_merge\_without\_buffer (_BidirectionalIterator __first, _-
BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance
__len1, _Distance __len2)

```
  - ```

template<typename _BidirectionalIterator , typename _Distance , typename _Compare >
void std::\_\_merge\_without\_buffer (_BidirectionalIterator __first, _-
BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance
__len1, _Distance __len2, _Compare __comp)

```
  - ```

template<typename _Iterator , typename _Compare >
void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _-
Compare __comp)

```
  - ```

template<typename _Iterator >
void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)

```
  - ```

template<typename _ForwardIterator , typename _Predicate >
_ForumdIterator std::\_\_partition (_ForwardIterator __first, _ForwardIterator __-
__last, _Predicate __pred, forward_iterator_tag)

```
  - ```

template<typename _BidirectionalIterator , typename _Predicate >
_BidirectionalIterator std::\_\_partition (_BidirectionalIterator __first, _-
BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)

```

- `template<typename _BidirectionalIterator >`  
`void std::\_\_reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void std::\_\_rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void std::\_\_rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance >`  
`_BidirectionalIterator1 std::\_\_rotate\_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp >`  
`_ForwardIterator std::\_\_search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp >`  
`_RandomAccessIter std::\_\_search\_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >`  
`_ForwardIterator std::\_\_search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _BinaryPredicate >`  
`_RandomAccessIter std::\_\_search\_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance >`  
`_ForwardIterator std::\_\_stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >`  
`void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _`

RandomAccessIterator \_\_last, \_Pointer \_\_buffer, \_Distance \_\_buffer\_size)

- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`  
`void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare`  
`__comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator _-`  
`__first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator _-`  
`__first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_-`  
`RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_-`  
`RandomAccessIterator __first, _RandomAccessIterator __last, _Compare`  
`__comp)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator _-`  
`__last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward _-`  
`iterator_tag, output_iterator_tag)`

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`  
`output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __`  
`__last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_`  
`iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp & __val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _`  
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator`  
`__result)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __`  
`first, _InputIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator`  
`__first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`  
`& __val)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _-`  
`BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function`  
`__f)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`  
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator`  
`__gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`  
`__middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`  
`__middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred)`

- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`

- `template<typename _Tp >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`

- `template<typename _BidirectionalIterator >`  
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`  
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __-`  
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`  
`__predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _-`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _`  
`BinaryPredicate >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator _-`  
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator _-`  
`last, _Integer __count, const _Tp &__val)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 _-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`  
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 _-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator _-`  
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator _-`  
`result, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _UniformRandomNumberGenerator & __g)`
- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_-`  
`op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`  
`_BinaryPredicate __binary_pred)`

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator , typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator , typename _Tp >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator , typename _Tp , typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

### 6.287.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_algo.h](#).

## 6.288 `stl_algobase.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define \_GLIBCXX\_MOVE3(_Tp, _Up, _Vp)`
- `#define \_GLIBCXX\_MOVE\_BACKWARD3(_Tp, _Up, _Vp)`

### Functions

- `template<bool _IsMove, typename _II , typename _OI >`  
`_OI std::\_\_copy\_move\_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type std::\_\_copy\_move\_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`  
`iterator< _CharT, char_traits< _CharT > > >::__type std::copy_move_a2`  
`(const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits<`  
`_CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type`  
`std::copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT >`  
`>, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI std::copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __-`  
`result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __-`  
`result)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, void >::__type std::-`  
`fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::-`  
`fill_a (_Tp * __first, _Tp * __last, const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if<!__is_scalar< _Tp >::__value, _OutputIterator >::__-`  
`type std::fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::-`  
`fill_n_a (_Tp * __first, _Size __n, const _Tp &__c)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __-`  
`__first2, _II2 __last2)`
- `long long std::lg (long long __n)`
- `long std::lg (long __n)`
- `template<typename _Size >`  
`_Size std::lg (_Size __n)`
- `int std::lg (int __n)`
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type std::miter_base (_Iterator __it)`
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type std::niter_base (_Iterator __it)`

- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`  
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill\_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter\_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2, _Compare __comp)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`  
`__last2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp &__val)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`  
`pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 6.288.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_algobase.h](#).

## 6.289 `std_bvector.h` File Reference

### Classes

- struct [std::hash<::vector< bool, \\_Alloc > >](#)  
*std::hash specialization for vector<bool>.*
- class [std::vector< bool, \\_Alloc >](#)  
*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*

### Namespaces

- namespace [std](#)

### Typedefs

- typedef unsigned long [std::\\_Bit\\_type](#)

### Enumerations

- enum { [\\_S\\_word\\_bit](#) }

### Functions

- void [std::\\_fill\\_bvector](#) (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, bool \_\_x)
- void [std::fill](#) (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, const bool &\_\_x)
- \_Bit\_iterator [std::operator+](#) (ptrdiff\_t \_\_n, const \_Bit\_iterator &\_\_x)

- `_Bit_const_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `ptrdiff_t std::operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)

### 6.289.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_bvector.h](#).

## 6.290 stl\_construct.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _T1 , typename _T2 >`  
void [std::\\_Construct](#) (`_T1 *__p`, `_T2 &&__value`)
- `template<typename _ForwardIterator , typename _Tp >`  
void [std::\\_Destroy](#) (`_ForwardIterator __first`, `_ForwardIterator __last`, `allocator<_Tp > &`)
- `template<typename _ForwardIterator , typename _Allocator >`  
void [std::\\_Destroy](#) (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Allocator &__alloc`)
- `template<typename _ForwardIterator >`  
void [std::\\_Destroy](#) (`_ForwardIterator __first`, `_ForwardIterator __last`)
- `template<typename _Tp >`  
void [std::\\_Destroy](#) (`_Tp *__pointer`)

### 6.290.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_construct.h](#).

## 6.291 `std_deque.h` File Reference

### Classes

- class `std::_Deque_base<_Tp, _Alloc>`
- struct `std::_Deque_iterator<_Tp, _Ref, _Ptr>`  
*A deque::iterator.*
- class `std::deque<_Tp, _Alloc>`  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*

### Namespaces

- namespace `std`

### Defines

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

### Functions

- `size_t std::_deque_buf_size (size_t __size)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy (_Deque_iterator<_Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator<_Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy (_Deque_iterator<_Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator<_Tp, _Tp &, _Tp * > __last, _Deque_`  
`iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator<_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator<_Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator<_Tp, _Tp &, _Tp * > __first, _Deque_iterator<_Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator<_Tp, _Tp &, _Tp * > &__first, const _`  
`Deque_iterator<_Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`  
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _`  
`Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator- (const _`  
`Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref,`  
`_Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const`  
`_Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`  
`RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

### 6.291.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_deque.h](#).

### 6.291.2 Define Documentation

#### 6.291.2.1 #define \_GLIBCXX\_DEQUE\_BUF\_SIZE

This function controls the size of memory nodes.

#### Parameters

*size* The size of an element.

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 82 of file `stl_deque.h`.

## 6.292 `stl_function.h` File Reference

### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::binary_negate<_Predicate >`  
*One of the negation functors.*
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`  
*One of the adaptors for member /// pointers.*
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`  
*One of the adaptors for member /// pointers.*
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`

One of the *adaptors for member /// pointers*.

- class `std::const_mem_fun_t<_Ret, _Tp >`  
One of the *adaptors for member /// pointers*.
- struct `std::divides<_Tp >`  
One of the *math functors*.
- struct `std::equal_to<_Tp >`  
One of the *comparison functors*.
- struct `std::greater<_Tp >`  
One of the *comparison functors*.
- struct `std::greater_equal<_Tp >`  
One of the *comparison functors*.
- struct `std::less<_Tp >`  
One of the *comparison functors*.
- struct `std::less_equal<_Tp >`  
One of the *comparison functors*.
- struct `std::logical_and<_Tp >`  
One of the *Boolean operations functors*.
- struct `std::logical_not<_Tp >`  
One of the *Boolean operations functors*.
- struct `std::logical_or<_Tp >`  
One of the *Boolean operations functors*.
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`  
One of the *adaptors for member /// pointers*.
- class `std::mem_fun1_t<_Ret, _Tp, _Arg >`  
One of the *adaptors for member /// pointers*.
- class `std::mem_fun_ref_t<_Ret, _Tp >`  
One of the *adaptors for member /// pointers*.
- class `std::mem_fun_t<_Ret, _Tp >`

*One of the [adaptors for member /// pointers](#).*

- struct [std::minus<\\_Tp >](#)  
*One of the [math functors](#).*
- struct [std::modulus<\\_Tp >](#)  
*One of the [math functors](#).*
- struct [std::multiplies<\\_Tp >](#)  
*One of the [math functors](#).*
- struct [std::negate<\\_Tp >](#)  
*One of the [math functors](#).*
- struct [std::not\\_equal\\_to<\\_Tp >](#)  
*One of the [comparison functors](#).*
- struct [std::plus<\\_Tp >](#)  
*One of the [math functors](#).*
- class [std::pointer\\_to\\_binary\\_function<\\_Arg1, \\_Arg2, \\_Result >](#)  
*One of the [adaptors for function pointers](#).*
- class [std::pointer\\_to\\_unary\\_function<\\_Arg, \\_Result >](#)  
*One of the [adaptors for function pointers](#).*
- struct [std::unary\\_function<\\_Arg, \\_Result >](#)
- class [std::unary\\_negate<\\_Predicate >](#)  
*One of the [negation functors](#).*

## Namespaces

- namespace [std](#)

## Functions

- [template<typename \\_Ret, typename \\_Tp > mem\\_fun\\_t<\\_Ret, \\_Tp > \*\*std::mem\\_fun\*\* \(\\_Ret\(\\_Tp::\\*\\_f\)\(\)\)](#)
- [template<typename \\_Ret, typename \\_Tp, typename \\_Arg > mem\\_fun1\\_t<\\_Ret, \\_Tp, \\_Arg > \*\*std::mem\\_fun\*\* \(\\_Ret\(\\_Tp::\\*\\_f\)\(\\_Arg\)\)](#)

- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::*_f)(-_Arg))`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::*_f)())`
- `template<typename _Predicate >`  
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*_x)(-_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*_x)(_Arg1, _Arg2))`

### 6.292.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_function.h](#).

## 6.293 `stl_heap.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`

- `template<typename _RandomAccessIterator >`  
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`  
`n, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`  
`n)`
- `template<typename _RandomAccessIterator >`  
`void std::__pop_heap (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__pop_heap (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _RandomAccessIterator __result, _Compare`  
`__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _-`  
`Compare >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`  
`holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`  
`holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

### 6.293.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_heap.h](#).

## 6.294 `stl_iterator.h` File Reference

### Classes

- class [std::back\\_insert\\_iterator<\\_Container >](#)  
*Turns assignment into insertion.*
- class [std::front\\_insert\\_iterator<\\_Container >](#)  
*Turns assignment into insertion.*
- class [std::insert\\_iterator<\\_Container >](#)  
*Turns assignment into insertion.*
- class [std::move\\_iterator<\\_Iterator >](#)
- class [std::reverse\\_iterator<\\_Iterator >](#)

## Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Defines

- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<typename _Container >`  
`back_insert_iterator< _Container > std::back\_inserter (_Container &__x)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front\_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::make\_move\_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool \_\_gnu\_cxx::operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool \_\_gnu\_cxx::operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > \_\_gnu\_cxx::operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`

- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container >::difference_type` **gnu\_cxx::operator-** (const `__normal_iterator< _Iterator, _Container >` &\_lhs, const `__normal_iterator< _Iterator, _Container >` &\_rhs)
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator-` (const `move_iterator< _IteratorL >` &\_x, const `move_iterator< _IteratorR >` &\_y)-> `decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::difference_type` **std::operator-** (const `reverse_iterator< _Iterator >` &\_x, const `reverse_iterator< _Iterator >` &\_y)
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator-` (const `reverse_iterator< _IteratorL >` &\_x, const `reverse_iterator< _IteratorR >` &\_y)-> `decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`auto gnu_cxx::operator-` (const `__normal_iterator< _IteratorL, _Container >` &\_lhs, const `__normal_iterator< _IteratorR, _Container >` &\_rhs)-> `decltype(__lhs.base()-__rhs.base())`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<` (const `move_iterator< _IteratorL >` &\_x, const `move_iterator< _IteratorR >` &\_y)
- `template<typename _Iterator >`  
`bool std::operator<` (const `reverse_iterator< _Iterator >` &\_x, const `reverse_iterator< _Iterator >` &\_y)
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<` (const `reverse_iterator< _IteratorL >` &\_x, const `reverse_iterator< _IteratorR >` &\_y)
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator<` (const `__normal_iterator< _IteratorL, _Container >` &\_lhs, const `__normal_iterator< _IteratorR, _Container >` &\_rhs)
- `template<typename _Iterator, typename _Container >`  
`bool gnu_cxx::operator<` (const `__normal_iterator< _Iterator, _Container >` &\_lhs, const `__normal_iterator< _Iterator, _Container >` &\_rhs)
- `template<typename _Iterator >`  
`bool std::operator<=` (const `reverse_iterator< _Iterator >` &\_x, const `reverse_iterator< _Iterator >` &\_y)
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<=` (const `move_iterator< _IteratorL >` &\_x, const `move_iterator< _IteratorR >` &\_y)
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<=` (const `reverse_iterator< _IteratorL >` &\_x, const `reverse_iterator< _IteratorR >` &\_y)
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator<=` (const `__normal_iterator< _IteratorL, _Container >` &\_lhs, const `__normal_iterator< _IteratorR, _Container >` &\_rhs)

- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`

### 6.294.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

Definition in file [stl\\_iterator.h](#).

## 6.295 `stl_iterator_base_funcs.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Distance >`  
`void std::__advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void std::__advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void std::__advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`iterator_traits< _RandomAccessIterator >::difference_type std::__distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type std::__distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`

- `template<typename _InputIterator, typename _Distance >`  
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator >`  
`iterator_traits<_InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _ForwardIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _ForwardIterator >::__value, _ForwardIterator >::__type std::next (_ForwardIterator __x, typename iterator_traits<_ForwardIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`  
`__gnu_cxx::__enable_if< __is_iterator< _BidirectionalIterator >::__value, _BidirectionalIterator >::__type std::prev (_BidirectionalIterator __x, typename iterator_traits<_BidirectionalIterator >::difference_type __n=1)`

### 6.295.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility functions, such as [distance\(\)](#) and [advance\(\)](#).

Definition in file [stl\\_iterator\\_base\\_funcs.h](#).

## 6.296 `stl_iterator_base_types.h` File Reference

### Classes

- struct [std::bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct [std::forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- struct [std::input\\_iterator\\_tag](#)  
*Marking input iterators.*
- struct [std::iterator<\\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)  
*Common iterator class.*
- struct [std::iterator\\_traits<\\_Iterator >](#)  
*Traits class for iterators.*

- struct `std::iterator_traits< _Tp * >`  
*Partial specialization for pointer types.*
- struct `std::iterator_traits< const _Tp * >`  
*Partial specialization for const pointer types.*
- struct `std::output_iterator_tag`  
*Marking output iterators.*
- struct `std::random_access_iterator_tag`  
*Random-access iterators support a superset of bidirectional /// iterator operations.*

## Namespaces

- namespace `std`

## Functions

- template<typename `_Iter` >  
`iterator_traits< _Iter >::iterator_category` `std::__iterator_category` (`const _Iter &`)

### 6.296.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file `stl_iterator_base_types.h`.

## 6.297 `stl_list.h` File Reference

### Classes

- class `std::_List_base< _Tp, _Alloc >`  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- struct `std::_List_const_iterator< _Tp >`

*A list::const\_iterator.*

- struct [std::\\_List\\_iterator< \\_Tp >](#)  
*A list::iterator.*
- struct [std::\\_List\\_node< \\_Tp >](#)  
*An actual node in the list.*
- struct [std::\\_List\\_node\\_base](#)  
*Common part of a node in the list.*
- class [std::list< \\_Tp, \\_Alloc >](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Val >`  
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`  
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

### 6.297.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_list.h](#).

## 6.298 `std_map.h` File Reference

### Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`

### 6.298.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_map.h](#).

## 6.299 [stl\\_multimap.h](#) File Reference

### Classes

- class [std::multimap< \\_Key, \\_Tp, \\_Compare, \\_Alloc >](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator==` (`const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>` (`const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>=` (`const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap` (`multimap< _Key, _Tp, _Compare, _Alloc > &__x,` `multimap<`  
`_Key, _Tp, _Compare, _Alloc > &__y`)

### 6.299.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std::multimap.h](#).

## 6.300 `std::multiset.h` File Reference

### Classes

- class `std::multiset< _Key, _Compare, _Alloc >`  
*A standard container made up of elements, which can be retrieved in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!=` (`const multiset< _Key, _Compare, _Alloc > &__x,` `const`  
`multiset< _Key, _Compare, _Alloc > &__y`)
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<` (`const multiset< _Key, _Compare, _Alloc > &__x,` `const`  
`multiset< _Key, _Compare, _Alloc > &__y`)

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

### 6.300.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_multiset.h](#).

## 6.301 `stl_numeric.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Tp >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp std::inner_product` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_Tp __init`, `_BinaryOperation1 __binary_op1`, `_BinaryOperation2 __binary_op2`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp std::inner_product` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_Tp __init`)
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::iota` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Tp __value`)
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::partial_sum` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`, `_BinaryOperation __binary_op`)
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::partial_sum` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`)

### 6.301.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_numeric.h](#).

## 6.302 `stl_pair.h` File Reference

### Classes

- struct `std::pair<_T1, _T2 >`  
*pair holds two objects of arbitrary type.*

### Namespaces

- namespace `std`

### Functions

- `template<class _T1, class _T2 >`  
`pair<typename __decay_and_strip<_T1 >::__type, typename __decay_and_strip<_T2 >::__type >` `std::make_pair` (`_T1 &&__x`, `_T2 &&__y`)

- `template<class _T1, class _T2 >`  
`bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`  
`void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)`

### 6.302.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_pair.h](#).

## 6.303 stl\_queue.h File Reference

### Classes

- class [std::priority\\_queue< \\_Tp, \\_Sequence, \\_Compare >](#)  
*A standard container automatically sorting its contents.*
- class [std::queue< \\_Tp, \\_Sequence >](#)  
*A standard container giving FIFO behavior.*

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!=(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<=(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator==(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>=(const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`void std::swap(priority_queue<_Tp, _Sequence, _Compare > &__x, priority_queue<_Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void std::swap(queue<_Tp, _Seq > &__x, queue<_Tp, _Seq > &__y)`

### 6.303.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std::queue.h](#).

## 6.304 `std::raw_storage_iter.h` File Reference

### Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp >`

### Namespaces

- namespace `std`

### 6.304.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_raw\\_storage\\_iter.h](#).

## 6.305 stl\_relops.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::rel\\_ops](#)

### Functions

- `template<class _Tp >`  
`bool std::rel\_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator>= (const _Tp &__x, const _Tp &__y)`

### 6.305.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file [stl\\_relops.h](#).

## 6.306 `stl_set.h` File Reference

### Classes

- class `std::set<_Key, _Compare, _Alloc >`  
*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)`

### 6.306.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `stl_set.h`.

## 6.307 `std_stack.h` File Reference

### Classes

- class [std::stack<\\_Tp, \\_Sequence >](#)  
*A standard container giving FILO behavior.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`

### 6.307.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_stack.h](#).

## 6.308 `stl_tempbuf.h` File Reference

### Classes

- class `std::_Temporary_buffer<_ForwardIterator, _Tp >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp > pair<_Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _Tp > void std::return_temporary_buffer (_Tp *__p)`

### 6.308.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_tempbuf.h](#).

## 6.309 `stl_tree.h` File Reference

### Namespaces

- namespace `std`

### Enumerations

- enum `_Rb_tree_color { _S_red, _S_black }`

### Functions

- `std::_attribute __ ((__pure__)) _Rb_tree_node_base *_Rb_tree_increment(_Rb_tree_node_base *__x) throw ()`
- `void std::_Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`

- `_Rb_tree_node_base * std::Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Val >  
bool std::operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Val >  
bool std::operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
void std::swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `const _Rb_tree_node_base *__root std::throw ()`

### 6.309.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_tree.h](#).

## 6.310 `std_uninitialized.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`void std::__uninitialized_construct_range (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _-`  
`InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _-`  
`InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`  
`name _Allocator >`  
`_ForwardIterator std::__uninitialized_copy_move (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::__uninitialized_copy_n (_InputIterator __first, _Size _-`  
`_n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::__uninitialized_copy_n (_RandomAccessIterator __-`  
`first, _Size _n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _-`  
`Allocator >`

- ```

_FForwardIterator std:: uninitialized_fill_move (_ForwardIterator __result, _-
ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator
__last, _Allocator &__alloc)

```
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator > void std:: uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
  - `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 > void std:: uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)`
  - `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator > _ForwardIterator std:: uninitialized_move_a (_InputIterator __first, _-InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
  - `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type- name _Allocator > _ForwardIterator std:: uninitialized_move_copy (_InputIterator1 __first1, _-InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-ForwardIterator __result, _Allocator &__alloc)`
  - `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _-Allocator > void std:: uninitialized_move_fill (_InputIterator __first1, _InputIterator _-last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _-Allocator &__alloc)`
  - `template<typename _InputIterator, typename _ForwardIterator > _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
  - `template<typename _InputIterator, typename _Size, typename _ForwardIterator > _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _-ForwardIterator __result)`
  - `template<typename _ForwardIterator, typename _Tp > void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
  - `template<typename _ForwardIterator, typename _Size, typename _Tp > void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &_-__x)`

### 6.310.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_uninitialized.h](#).

## 6.311 `std_vector.h` File Reference

### Classes

- struct `std::_Vector_base<_Tp, _Alloc>`  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- class `std::vector<_Tp, _Alloc>`  
*A standard container which offers fixed time access to individual elements in any order.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!=(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<=(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator==(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>=(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap(vector<_Tp, _Alloc> &__x, vector<_Tp, _Alloc> &__y)`

#### 6.311.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_vector.h](#).

## 6.312 [stream\\_iterator.h](#) File Reference

### Classes

- class [std::istream\\_iterator](#)< [\\_Tp](#), [\\_CharT](#), [\\_Traits](#), [\\_Dist](#) >  
*Provides input iterator semantics for streams.*
- class [std::ostream\\_iterator](#)< [\\_Tp](#), [\\_CharT](#), [\\_Traits](#) >  
*Provides output iterator semantics for streams.*

### Namespaces

- namespace [std](#)

### Functions

- `template<class \_Tp, class \_CharT, class \_Traits, class \_Dist >`  
`bool std::operator!=(const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_x, const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_y)`
- `template<typename \_Tp, typename \_CharT, typename \_Traits, typename \_Dist >`  
`bool std::operator==(const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_x, const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_y)`

#### 6.312.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stream\\_iterator.h](#).

## 6.313 [streambuf](#) File Reference

### Classes

- class [std::basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) >

*The actual work of input and output (interface).*

*This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< wchar_t > * __sbin, basic_streambuf< wchar_t > * __sbout, bool & __ineof)`
- `template<>`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< char > * __sbin, basic_streambuf< char > * __sbout, bool & __ineof)`

### 6.313.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

## 6.314 streambuf.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs (basic_streambuf< _CharT, _Traits > * __sbin, basic_streambuf< _CharT, _Traits > * __sbout)`
- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`

### 6.314.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf.tcc](#).

## 6.315 streambuf\_iterator.h File Reference

### Classes

- class [std::istreambuf\\_iterator<\\_CharT, \\_Traits >](#)  
*Provides input iterator semantics for streambufs.*
- class [std::ostreambuf\\_iterator<\\_CharT, \\_Traits >](#)  
*Provides output iterator semantics for streambufs.*

### Namespaces

- namespace [std](#)

### Functions

- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy_move_a2(_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy_move_a2(const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::copy_move_a2(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`

- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_`  
`iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT >`  
`__first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`

### 6.315.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf\\_iterator.h](#).

## 6.316 string File Reference

### 6.316.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

## 6.317 string File Reference

### Classes

- class [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#)  
*Class [std::basic\\_string](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Typedefs

- typedef `basic_string< char >` [\\_\\_gnu\\_debug::string](#)

- typedef basic\_string< wchar\_t > **\_\_gnu\_debug::wstring**

## Functions

- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **\_\_gnu\_debug::getline** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_str, \_CharT \_\_delim)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **\_\_gnu\_debug::getline** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_str)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 bool **\_\_gnu\_debug::operator!=** (const \_CharT \*\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 bool **\_\_gnu\_debug::operator!=** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 bool **\_\_gnu\_debug::operator!=** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 basic\_string< \_CharT, \_Traits, \_Allocator > **\_\_gnu\_debug::operator+** (const \_CharT \*\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 basic\_string< \_CharT, \_Traits, \_Allocator > **\_\_gnu\_debug::operator+** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 basic\_string< \_CharT, \_Traits, \_Allocator > **\_\_gnu\_debug::operator+** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, \_CharT \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 basic\_string< \_CharT, \_Traits, \_Allocator > **\_\_gnu\_debug::operator+** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 basic\_string< \_CharT, \_Traits, \_Allocator > **\_\_gnu\_debug::operator+** (\_CharT \_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 bool **\_\_gnu\_debug::operator<** (const \_CharT \*\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Allocator >  
 bool **\_\_gnu\_debug::operator<** (const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_lhs, const basic\_string< \_CharT, \_Traits, \_Allocator > &\_\_rhs)

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_debug::operator<<`  
`(std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT,`  
`_Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const _CharT *__lhs, const basic_string< _-`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _-`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _-`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _-`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & __gnu_debug::operator>>`  
`(std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,`  
`_Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__-`  
`lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

### 6.317.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

## 6.318 stringfwd.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef `basic_string< char >` [std::string](#)
- typedef `basic_string< char16_t >` [std::u16string](#)
- typedef `basic_string< char32_t >` [std::u32string](#)
- typedef `basic_string< wchar_t >` [std::wstring](#)

### 6.318.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stringfwd.h](#).

## 6.319 system\_error File Reference

### Classes

- class `std::error_category`  
*error\_category*
- struct `std::error_code`  
*error\_code*
- struct `std::error_condition`  
*error\_condition*
- struct `std::hash< error_code >`  
*std::hash specialization for error\_code.*
- struct `std::is_error_code_enum< _Tp >`  
*is\_error\_code\_enum*
- struct `std::is_error_condition_enum< _Tp >`  
*is\_error\_condition\_enum*
- class `std::system_error`  
*Thrown to indicate error code of underlying system.*

### Namespaces

- namespace `std`

### Functions

- `_GLIBCXX_CONST const error_category & std::generic_category () throw ()`
- `error_code std::make_error_code (errc __e)`
- `error_condition std::make_error_condition (errc __e)`
- `bool std::operator!= (const error_code &__lhs, const error_condition &__rhs)`
- `bool std::operator!= (const error_condition &__lhs, const error_condition &__rhs)`
- `bool std::operator!= (const error_code &__lhs, const error_code &__rhs)`
- `bool std::operator!= (const error_condition &__lhs, const error_code &__rhs)`
- `bool std::operator< (const error_code &__lhs, const error_code &__rhs)`

- `bool std::operator<` (const error\_condition &\_\_lhs, const error\_condition &\_\_rhs)
- `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::operator<<` (basic\_ostream< \_CharT, \_Traits > &\_\_os, const error\_code &\_\_e)
- `bool std::operator==` (const error\_condition &\_\_lhs, const error\_condition &\_\_rhs)
- `bool std::operator==` (const error\_code &\_\_lhs, const error\_condition &\_\_rhs)
- `bool std::operator==` (const error\_code &\_\_lhs, const error\_code &\_\_rhs)
- `bool std::operator==` (const error\_condition &\_\_lhs, const error\_code &\_\_rhs)
- `_GLIBCXX_CONST const error_category & std::system_category () throw ()`

## Variables

- error\_code `std::make_error_code` (errc)
- error\_condition `std::make_error_condition` (errc)

### 6.319.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system\\_error](#).

## 6.320 tag\_and\_trait.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_container\\_tag](#)  
*Basic associative-container.*
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)  
*Basic hash.*
- struct [\\_\\_gnu\\_pbds::basic\\_tree\\_tag](#)  
*Basic tree.*
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)  
*Binary-heap (array-based).*

- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)  
*Binomial-heap.*
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)  
*Collision-chaining hash.*
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)  
*Base data structure tag.*
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#)  
*container\_traits*
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)  
*General-probing hash.*
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)  
*List-update.*
- struct [\\_\\_gnu\\_pbds::null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)  
*Basic sequence.*

- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)  
*Basic string container, inclusive of strings, ropes, etc.*
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)  
*Thin heap.*
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)  
*tree.*
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)  
*trie.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

### 6.320.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag\\_and\\_trait.hpp](#).

## 6.321 tags.h File Reference

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- struct [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#)  
*Forces parallel sorting using balanced quicksort at compile time.*

- struct [\\_\\_gnu\\_parallel::balanced\\_tag](#)  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#)  
*Selects the constant block size variant for `std::find()`.*
- struct [\\_\\_gnu\\_parallel::default\\_parallel\\_tag](#)  
*Recommends parallel execution using the default parallel algorithm.*
- struct [\\_\\_gnu\\_parallel::equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [\\_\\_gnu\\_parallel::exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [\\_\\_gnu\\_parallel::find\\_tag](#)  
*Base class for for `std::find()` variants.*
- struct [\\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [\\_\\_gnu\\_parallel::quicksort\\_tag](#)

*Forces parallel sorting using unbalanced quicksort at compile time.*

- struct [\\_\\_gnu\\_parallel::sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [\\_\\_gnu\\_parallel::sequential\\_tag](#)  
*Forces sequential execution at compile time.*
- struct [\\_\\_gnu\\_parallel::unbalanced\\_tag](#)  
*Recommends parallel execution using static load-balancing at compile time.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.321.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

## 6.322 tgmth.h File Reference

### 6.322.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgmth.h](#).

## 6.323 thread File Reference

### Classes

- struct [std::hash< thread::id >](#)  
*std::hash specialization for thread::id.*
- class [std::thread](#)  
*thread*

- class `std::thread::id`  
*`thread::id`*

## Namespaces

- namespace `std`
- namespace `std::this_thread`

## Functions

- `thread::id` `std::this_thread::get_id ()`
- `bool` **`std::operator!=`** (`thread::id` `__x`, `thread::id` `__y`)
- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > &` **`std::operator<<`** (`basic_ostream< _-`  
`CharT, _Traits > &``__out`, `thread::id` `__id`)
- `bool` **`std::operator<=`** (`thread::id` `__x`, `thread::id` `__y`)
- `bool` **`std::operator>`** (`thread::id` `__x`, `thread::id` `__y`)
- `bool` **`std::operator>=`** (`thread::id` `__x`, `thread::id` `__y`)
- `template<typename _Rep, typename _Period >`  
`void` `std::this_thread::sleep_for` (`const chrono::duration< _Rep, _Period > &``__`  
`__rtime`)
- `template<typename _Clock, typename _Duration >`  
`void` `std::this_thread::sleep_until` (`const chrono::time_point< _Clock, _Duration`  
`> &``__atime`)
- `void` **`std::swap`** (`thread &``__x`, `thread &``__y`)
- `void` `std::this_thread::yield ()`

### 6.323.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

## 6.324 `throw_allocator.h` File Reference

### Classes

- struct `__gnu_cxx::annotate_base`  
*Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*

- struct `__gnu_cxx::condition_base`  
*Base struct for condition policy.*
- struct `__gnu_cxx::forced_error`  
*Thrown by exception safety machinery.*
- struct `__gnu_cxx::limit_condition`  
*Base class for incremental control and throw.*
- struct `__gnu_cxx::limit_condition::always_adjustor`  
*Always enter the condition.*
- struct `__gnu_cxx::limit_condition::limit_adjustor`  
*Enter the nth condition.*
- struct `__gnu_cxx::limit_condition::never_adjustor`  
*Never enter the condition.*
- struct `__gnu_cxx::random_condition`  
*Base class for random probability control and throw.*
- struct `__gnu_cxx::random_condition::always_adjustor`  
*Always enter the condition.*
- struct `__gnu_cxx::random_condition::group_adjustor`  
*Group condition.*
- struct `__gnu_cxx::random_condition::never_adjustor`  
*Never enter the condition.*
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`  
*Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.*  
*Note: Deallocate not allowed to throw.*
- struct `__gnu_cxx::throw_allocator_limit< _Tp >`  
*Allocator throwing via limit condition.*
- struct `__gnu_cxx::throw_allocator_random< _Tp >`  
*Allocator throwing via random condition.*
- struct `__gnu_cxx::throw_value_base< _Cond >`

*Class with exception generation control. Intended to be used as a `value_type` in templated code.*

- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)  
*Type throwing via limit condition.*
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)  
*Type throwing via random condition.*
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)  
*Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.*
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)  
*Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_throw\\_forced\\_error](#) ()
- template<typename `_Tp`, typename `_Cond` >  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const `throw_allocator_base< _Tp, _Cond >` &, const `throw_allocator_base< _Tp, _Cond >` &)
- template<typename `_Cond` >  
`throw_value_base< _Cond >` [\\_\\_gnu\\_cxx::operator\\*](#) (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)
- template<typename `_Cond` >  
`throw_value_base< _Cond >` [\\_\\_gnu\\_cxx::operator+](#) (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)
- template<typename `_Cond` >  
`throw_value_base< _Cond >` [\\_\\_gnu\\_cxx::operator-](#) (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)
- template<typename `_Cond` >  
bool [\\_\\_gnu\\_cxx::operator<](#) (const `throw_value_base< _Cond >` &\_\_a, const `throw_value_base< _Cond >` &\_\_b)
- [std::ostream](#) & [\\_\\_gnu\\_cxx::operator<<](#) ([std::ostream](#) &os, const `annotate_base` &\_\_b)

- `template<typename _Tp, typename _Cond >`  
`bool __gnu_cxx::operator==(const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond >`  
`bool __gnu_cxx::operator==(const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond >`  
`void __gnu_cxx::swap(throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`

### 6.324.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

Definition in file [throw\\_allocator.h](#).

## 6.325 `time_members.h` File Reference

### Namespaces

- namespace [std](#)

### 6.325.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [time\\_members.h](#).

## 6.326 `tree_policy.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Defines

- #define `PB_DS_BASE_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`

### 6.326.1 Detailed Description

Contains tree-related policies.

Definition in file [tree\\_policy.hpp](#).

## 6.327 tree\_trace\_base.hpp File Reference

### 6.327.1 Detailed Description

Contains tree-related policies.

Definition in file [tree\\_trace\\_base.hpp](#).

## 6.328 trie\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Defines

- #define `PB_DS_BASE_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`
- #define `PB_DS_CLASS_T_DEC`

### 6.328.1 Detailed Description

Contains trie-related policies.

Definition in file [trie\\_policy.hpp](#).

## 6.329 tuple File Reference

### Classes

- struct [std::\\_Tuple\\_impl<\\_Idx>](#)
- struct [std::\\_Tuple\\_impl<\\_Idx, \\_Head, \\_Tail...>](#)
- class [std::tuple<\\_Elements>](#)  
*tuple*
- class [std::tuple<\\_T1, \\_T2>](#)  
*tuple (2-element), with construction and assignment from a pair.*
- struct [std::tuple\\_element<0, tuple<\\_Head, \\_Tail...>>](#)
- struct [std::tuple\\_element<\\_\\_i, tuple<\\_Head, \\_Tail...>>](#)
- struct [std::tuple\\_size<tuple<\\_Elements...>>](#)  
*class tuple\_size*

### Namespaces

- namespace [std](#)

### Functions

- template<std::size\_t \_\_i, typename \_Head, typename... \_Tail>  
[\\_\\_add\\_ref<\\_Head>::type](#) **std::\_\_get\_helper** (\_Tuple\_impl< \_\_i, \_Head, \_Tail...> &\_\_t)
- template<std::size\_t \_\_i, typename \_Head, typename... \_Tail>  
[\\_\\_add\\_c\\_ref<\\_Head>::type](#) **std::\_\_get\_helper** (const \_Tuple\_impl< \_\_i, \_Head, \_Tail...> &\_\_t)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
tuple< \_TElements..., \_UElements...> **std::tuple\_cat\_helper** (const tuple< \_TElements...> &\_\_t, const \_\_index\_holder< \_TIdx...> &, const tuple< \_UElements...> &\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
tuple< \_TElements..., \_UElements...> **std::tuple\_cat\_helper** (tuple< \_TElements...> &&\_\_t, const \_\_index\_holder< \_TIdx...> &, const tuple< \_UElements...> &\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
tuple< \_TElements..., \_UElements...> **std::tuple\_cat\_helper** (const tuple<

- ```

    _TElements...> &__t, const __index_holder< _TIdx...> &, tuple< _-
    UElements...> &&__u, const __index_holder< _UIIdx...> &)

```
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIIdx>`  
`tuple< _TElements..., _UElements...> std::tuple_cat_helper (tuple< _-`  
`TElements...> &&__t, const __index_holder< _TIdx...> &, tuple< _-`  
`UElements...> &&__u, const __index_holder< _UIIdx...> &)`
  - `template<std::size_t __i, typename... _Elements>`  
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type`  
`>::type std::get (const tuple< _Elements...> &__t)`
  - `template<std::size_t __i, typename... _Elements>`  
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type`  
`>::type std::get (tuple< _Elements...> &__t)`
  - `template<typename... _Elements>`  
`tuple< typename __decay_and_strip< _Elements >::__type...> std::make_`  
`tuple (_Elements &&...__args)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator< (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator== (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator> (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
  - `template<typename... _Elements>`  
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
  - `template<typename... _Elements>`  
`tuple< _Elements &...> std::tie (_Elements &...__args)`
  - `template<typename... _TElements, typename... _UElements>`  
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`  
`TElements...> &__t, const tuple< _UElements...> &__u)`
  - `template<typename... _TElements, typename... _UElements>`  
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`  
`TElements...> &__t, tuple< _UElements...> &&__u)`

- `template<typename... _TElements, typename... _UElements>`  
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`  
`&&_t, const tuple< _UElements...> &&_u)`
- `template<typename... _TElements, typename... _UElements>`  
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`  
`&&_t, tuple< _UElements...> &&_u)`

### 6.329.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

## 6.330 `type_traits` File Reference

### Classes

- struct [std::\\_\\_declval\\_protector< \\_Tp >](#)  
*declval*
- struct [std::add\\_lvalue\\_reference< \\_Tp >](#)  
*add\_lvalue\_reference*
- struct [std::add\\_rvalue\\_reference< \\_Tp >](#)  
*add\_rvalue\_reference*
- struct [std::aligned\\_storage< \\_Len, \\_Align >](#)  
*Alignment type.*
- struct [std::conditional< \\_Cond, \\_Iftrue, \\_Iffalse >](#)  
*conditional*
- class [std::decay< \\_Tp >](#)  
*decay*
- struct [std::enable\\_if< bool, \\_Tp >](#)  
*enable\_if*
- struct [std::has\\_nothrow\\_assign< \\_Tp >](#)  
*has\_nothrow\_assign*
- struct [std::has\\_nothrow\\_copy\\_constructor< \\_Tp >](#)

*has\_nothrow\_copy\_constructor*

- struct `std::has_nothrow_default_constructor< _Tp >`  
*has\_nothrow\_default\_constructor*
- struct `std::has_trivial_assign< _Tp >`  
*has\_trivial\_assign*
- struct `std::has_trivial_copy_constructor< _Tp >`  
*has\_trivial\_copy\_constructor*
- struct `std::has_trivial_default_constructor< _Tp >`  
*has\_trivial\_default\_constructor*
- struct `std::has_trivial_destructor< _Tp >`  
*has\_trivial\_destructor*
- struct `std::is_base_of< _Base, _Derived >`  
*is\_base\_of*
- struct `std::is_constructible< _Tp, _Args >`  
*is\_constructible*
- struct `std::is_convertible< _From, _To >`  
*is\_convertible*
- struct `std::is_explicitly_convertible< _From, _To >`  
*is\_explicitly\_convertible*
- struct `std::is_lvalue_reference< typename >`  
*is\_lvalue\_reference*
- struct `std::is_pod< _Tp >`  
*is\_pod*
- struct `std::is_reference< _Tp >`  
*is\_reference*
- struct `std::is_rvalue_reference< typename >`  
*is\_rvalue\_reference*
- struct `std::is_signed< _Tp >`

*is\_signed*

- struct `std::is_standard_layout< _Tp >`  
*is\_standard\_layout*
- struct `std::is_trivial< _Tp >`  
*is\_trivial*
- struct `std::is_unsigned< _Tp >`  
*is\_unsigned*
- struct `std::make_signed< _Tp >`  
*make\_signed*
- struct `std::make_unsigned< _Tp >`  
*make\_unsigned*
- struct `std::remove_reference< _Tp >`  
*remove\_reference*

## Namespaces

- namespace `std`

## Functions

- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type` **std::declval** ()

### 6.330.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type\\_traits](#).

## 6.331 `type_traits` File Reference

### Classes

- struct `std::tr1::__is_member_pointer_helper< _Tp >`

*is\_member\_pointer*

- struct `std::tr1::add_const< _Tp >`  
*add\_const*
- struct `std::tr1::add_cv< _Tp >`  
*add\_cv*
- struct `std::tr1::add_pointer< _Tp >`  
*add\_pointer*
- struct `std::tr1::add_volatile< _Tp >`  
*add\_volatile*
- struct `std::tr1::alignment_of< _Tp >`  
*alignment\_of*
- struct `std::tr1::extent< typename, _UInt >`  
*extent*
- struct `std::tr1::has_virtual_destructor< _Tp >`  
*has\_virtual\_destructor*
- struct `std::tr1::integral_constant< _Tp, __v >`  
*integral\_constant*
- struct `std::tr1::is_abstract< _Tp >`  
*is\_abstract*
- struct `std::tr1::is_arithmetic< _Tp >`  
*is\_arithmetic*
- struct `std::tr1::is_array< typename >`  
*is\_array*
- struct `std::tr1::is_class< _Tp >`  
*is\_class*
- struct `std::tr1::is_compound< _Tp >`  
*is\_compound*
- struct `std::tr1::is_const< typename >`

*is\_const*

- struct `std::tr1::is_empty< _Tp >`  
*is\_empty*
- struct `std::tr1::is_enum< _Tp >`  
*is\_enum*
- struct `std::tr1::is_floating_point< _Tp >`  
*is\_floating\_point*
- struct `std::tr1::is_function< typename >`  
*is\_function*
- struct `std::tr1::is_fundamental< _Tp >`  
*is\_fundamental*
- struct `std::tr1::is_integral< _Tp >`  
*is\_integral*
- struct `std::tr1::is_member_function_pointer< _Tp >`  
*is\_member\_function\_pointer*
- struct `std::tr1::is_member_object_pointer< _Tp >`  
*is\_member\_object\_pointer*
- struct `std::tr1::is_object< _Tp >`  
*is\_object*
- struct `std::tr1::is_pointer< _Tp >`  
*is\_pointer*
- struct `std::tr1::is_polymorphic< _Tp >`  
*is\_polymorphic*
- struct `std::tr1::is_same< typename, typename >`  
*is\_same*
- struct `std::tr1::is_scalar< _Tp >`  
*is\_scalar*
- struct `std::tr1::is_union< _Tp >`

*is\_union*

- struct `std::tr1::is_void< _Tp >`  
*is\_void*
- struct `std::tr1::is_volatile< typename >`  
*is\_volatile*
- struct `std::tr1::rank< typename >`  
*rank*
- struct `std::tr1::remove_all_extents< _Tp >`  
*remove\_all\_extents*
- struct `std::tr1::remove_const< _Tp >`  
*remove\_const*
- struct `std::tr1::remove_cv< _Tp >`  
*remove\_cv*
- struct `std::tr1::remove_extent< _Tp >`  
*remove\_extent*
- struct `std::tr1::remove_pointer< _Tp >`  
*remove\_pointer*
- struct `std::tr1::remove_volatile< _Tp >`  
*remove\_volatile*

## Namespaces

- namespace `std`
- namespace `std::tr1`

## Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`

## Typedefs

- typedef integral\_constant< bool, false > [std::tr1::false\\_type](#)
- typedef integral\_constant< bool, true > [std::tr1::true\\_type](#)

### 6.331.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/type\\_traits](#).

## 6.332 type\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Type >  
bool [\\_\\_gnu\\_cxx::\\_\\_is\\_null\\_pointer](#) (\_Type \*\_\_ptr)
- template<typename \_Type >  
bool [\\_\\_gnu\\_cxx::\\_\\_is\\_null\\_pointer](#) (\_Type)

### 6.332.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type\\_traits.h](#).

## 6.333 type\_utils.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- #define [PB\\_DS\\_STATIC\\_ASSERT](#)(UNIQUE, E)

## Typedefs

- typedef [std::tr1::integral\\_constant](#)< int, 0 > [\\_\\_gnu\\_pbds::detail::false\\_type](#)
- typedef [std::tr1::integral\\_constant](#)< int, 1 > [\\_\\_gnu\\_pbds::detail::true\\_type](#)

### 6.333.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

Definition in file [type\\_utils.hpp](#).

## 6.334 typeinfo File Reference

### Classes

- class [std::bad\\_cast](#)  
*Thrown during incorrect typecasting.  
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [std::bad\\_typeid](#)  
*Thrown when a NULL pointer in a `typeid` expression is used.*
- class [std::type\\_info](#)  
*Part of RTTI.*

### Namespaces

- namespace [std](#)

### Defines

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`

### 6.334.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeinfo](#).

## 6.335 `typelist.h` File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::typelist`

### Defines

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

### Functions

- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`

- `template<typename Gn , typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`

### 6.335.1 Detailed Description

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

## 6.336 types.h File Reference

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef `int64_t __gnu_parallel::_CASable`
- typedef `uint64_t __gnu_parallel::_SequenceIndex`
- typedef `uint16_t __gnu_parallel::_ThreadIndex`

### Enumerations

- enum `__gnu_parallel::_AlgorithmStrategy` { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum `__gnu_parallel::_FindAlgorithm` { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum `__gnu_parallel::_MultiwayMergeAlgorithm` { **LOSER\_TREE** }
- enum `__gnu_parallel::_Parallelism` {  
`__gnu_parallel::sequential`, `__gnu_parallel::parallel_unbalanced`, `__gnu_parallel::parallel_balanced`, `__gnu_parallel::parallel_omp_loop`,  
`__gnu_parallel::parallel_omp_loop_static`, `__gnu_parallel::parallel_taskqueue` }
- enum `__gnu_parallel::_PartialSumAlgorithm` { **RECURSIVE**, **LINEAR** }
- enum `__gnu_parallel::_SortAlgorithm` { **MWMS**, **QS**, **QS\_BALANCED** }
- enum `__gnu_parallel::_SplittingAlgorithm` { **SAMPLING**, **EXACT** }

## Variables

- static const int `__gnu_parallel::_CASable_bits`
- static const `_CASable` `__gnu_parallel::_CASable_mask`

### 6.336.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

## 6.337 `types_traits.hpp` File Reference

### Namespaces

- namespace `__gnu_pbds`

### 6.337.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types\\_traits.hpp](#).

## 6.338 `unique_copy.h` File Reference

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator` `__gnu_parallel::_parallel_unique_copy` (`_Iter` `__first`, `_Iter` `__last`, `_OutputIterator` `__result`, `_BinaryPredicate` `__binary_pred`)
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator` `__gnu_parallel::_parallel_unique_copy` (`_Iter` `__first`, `_Iter` `__last`, `_OutputIterator` `__result`)

### 6.338.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [unique\\_copy.h](#).

## 6.339 `unique_ptr.h` File Reference

### Classes

- struct [std::default\\_delete<\\_Tp>](#)  
*Primary template, default\_delete.*
- struct [std::default\\_delete<\\_Tp\[\]>](#)  
*Specialization, default\_delete.*
- class [std::unique\\_ptr<\\_Tp, \\_Tp\\_Deleter>](#)  
*20.7.12.2 unique\_ptr for single objects.*
- class [std::unique\\_ptr<\\_Tp\[\], \\_Tp\\_Deleter>](#)  
*20.7.12.3 unique\_ptr for array objects with a runtime length*

### Namespaces

- namespace [std](#)

### Functions

- [template<typename \\_Tp, typename \\_Tp\\_Deleter, typename \\_Up, typename \\_Up\\_Deleter> bool std::operator!=\(const unique\\_ptr<\\_Tp, \\_Tp\\_Deleter> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Up\\_Deleter> &\\_\\_y\)](#)
- [template<typename \\_Tp, typename \\_Tp\\_Deleter, typename \\_Up, typename \\_Up\\_Deleter> bool std::operator<\(const unique\\_ptr<\\_Tp, \\_Tp\\_Deleter> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Up\\_Deleter> &\\_\\_y\)](#)
- [template<typename \\_Tp, typename \\_Tp\\_Deleter, typename \\_Up, typename \\_Up\\_Deleter> bool std::operator<=\(const unique\\_ptr<\\_Tp, \\_Tp\\_Deleter> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Up\\_Deleter> &\\_\\_y\)](#)
- [template<typename \\_Tp, typename \\_Tp\\_Deleter, typename \\_Up, typename \\_Up\\_Deleter> bool std::operator==\(const unique\\_ptr<\\_Tp, \\_Tp\\_Deleter> &\\_\\_x, const unique\\_ptr<\\_Up, \\_Up\\_Deleter> &\\_\\_y\)](#)

- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter > bool std::operator> (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter, typename _Up, typename _Up_Deleter > bool std::operator>= (const unique_ptr< _Tp, _Tp_Deleter > &__x, const unique_ptr< _Up, _Up_Deleter > &__y)`
- `template<typename _Tp, typename _Tp_Deleter > void std::swap (unique_ptr< _Tp, _Tp_Deleter > &__x, unique_ptr< _Tp, _Tp_Deleter > &__y)`

### 6.339.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unique\\_ptr.h](#).

## 6.340 unordered\_map File Reference

### 6.340.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered\\_map](#).

## 6.341 unordered\_map File Reference

### Classes

- class `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
*Class [std::unordered\\_map](#) with safety/checking/debug instrumentation.*
- class `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`  
*Class [std::unordered\\_multimap](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`bool std::__debug::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`bool std::__debug::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`bool std::__debug::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`bool std::__debug::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`void std::__debug::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`void std::__debug::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

### 6.341.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered\\_map](#).

## 6.342 unordered\_map File Reference

### Classes

- class [std::\\_\\_profile::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)

Class `std::unordered_map` wrapper with performance instrumentation.

- class `std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`

Class `std::unordered_multimap` wrapper with performance instrumentation.

## Namespaces

- namespace `std`
- namespace `std::__profile`

## Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

## Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!=(const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!=(const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator==(const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator==(const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`void std::__profile::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`  
`void std::__profile::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

### 6.342.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered\\_map](#).

## 6.343 unordered\_map.h File Reference

### Classes

- class [std::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*
- class [std::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*

### Namespaces

- namespace [std](#)

### Functions

- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc , bool __cache_hash_code >`  
`bool std::operator!=(const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`bool std::operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _`  
`Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred,`  
`_Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`bool std::operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _`  
`Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _`  
`Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`bool std::operator== (const __unordered_multimap< _Key, _Tp, _Hash, _`  
`Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _`  
`Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred,`  
`_Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`  
`> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`void std::swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __`  
`cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`void std::swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __`  
`cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _`  
`Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_`  
`__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

### 6.343.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered\\_map.h](#).

## 6.344 unordered\_set File Reference

### 6.344.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered\\_set](#).

## 6.345 unordered\_set File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_multiset<\\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_multiset](#) with safety/checking/debug instrumentation.*
- class [std::\\_\\_debug::unordered\\_set<\\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_set](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >  
bool std::__debug::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >  
bool std::__debug::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__debug::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.345.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered\\_set](#).

## 6.346 unordered\_set File Reference

### Classes

- class [std::\\_\\_profile::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Unordered\_multiset wrapper with performance instrumentation.*
- class [std::\\_\\_profile::unordered\\_set< \\_Key, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Unordered\_set wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

## Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__profile::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__profile::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.346.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered\\_set](#).

## 6.347 unordered\_set.h File Reference

### Classes

- class [std::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*
- class [std::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

## Namespaces

- namespace [std](#)

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc,`  
`__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc`  
`> &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc,`  
`__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator== (const __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__-`  
`__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void std::swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __-`  
`cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void std::swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_-`  
`hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`  
`cache_hash_code > &__y)`

- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`void std::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.347.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered\\_set.h](#).

## 6.348 utility File Reference

### 6.348.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

## 6.349 utility File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`  
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`  
`(std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`  
`const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`  
`(const std::pair< _Tp1, _Tp2 > &__in)`

### 6.349.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/utility](#).

## 6.350 valarray File Reference

### Classes

- class [std::valarray<\\_Tp>](#)  
*Smart array designed to support numeric processing.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

### Functions

- `template<typename _Tp >`  
`_Expr<_BinClos<__not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun<__not_equal_to, _Tp >::result_type > std::operator!= (const`  
`valarray<_Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun<__not_equal_to, _Tp >::result_type > std::operator!= (const`  
`valarray<_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun<__not_equal_to, _Tp >::result_type > std::operator!= (const _`  
`Tp &__t, const valarray<_Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _-`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _-`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`  
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _-`  
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _-`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`  
`const valarray< _Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`_fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`  
`valarray< _Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _-`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`_fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`  
`_fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`  
`_fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`_fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`  
`_fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`  
`_fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`_fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _-`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`  
`_fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _-`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`  
`_fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`  
`const valarray< _Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _-`  
`_fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`  
`&__t, const valarray< _Tp > &__v)`

- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun<__bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray<_Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun<__bitwise_or, _Tp >::result_type > std::operator| (const valarray<_`  
`Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun<__bitwise_or, _Tp >::result_type > std::operator| (const valarray<_`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun<__bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`  
`const valarray<_Tp > &__v)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun<__logical_or, _Tp >::result_type > std::operator|| (const valarray<_`  
`Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun<__logical_or, _Tp >::result_type > std::operator|| (const valarray<_`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun<__logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`  
`const valarray<_Tp > &__v)`

### 6.350.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

## 6.351 valarray\_after.h File Reference

### Namespaces

- namespace [std](#)

## Defines

- #define **\_DEFINE\_EXPR\_BINARY\_FUNCTION**(\_Fun, \_UFun)
- #define **\_DEFINE\_EXPR\_BINARY\_OPERATOR**(\_Op, \_Name)
- #define **\_DEFINE\_EXPR\_UNARY\_FUNCTION**(\_Name, \_UName)
- #define **\_DEFINE\_EXPR\_UNARY\_OPERATOR**(\_Op, \_Name)

## Functions

- template<class \_Dom >  
\_Expr< \_UnClos< \_Abs, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::abs** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Abs, \_ValArray, \_Tp >, \_Tp > **std::abs** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Acos, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::acos** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Acos, \_ValArray, \_Tp >, \_Tp > **std::acos** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Asin, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::asin** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Asin, \_ValArray, \_Tp >, \_Tp > **std::asin** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Atan, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::atan** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Atan, \_ValArray, \_Tp >, \_Tp > **std::atan** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_Expr, \_Dom1, \_Dom2 >, type-  
name \_Dom1::value\_type > **std::atan2** (const \_Expr< \_Dom1, typename \_-  
Dom1::value\_type > &\_\_e1, const \_Expr< \_Dom2, typename \_Dom2::value\_-  
type > &\_\_e2)
- template<class \_Dom >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_ValArray, \_Dom, typename \_-  
Dom::value\_type >, typename \_Dom::value\_type > **std::atan2** (const \_Expr<  
\_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_-  
Dom::value\_type > &\_\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _-`  
`Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename`  
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value _-`  
`type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename _Dom::value_type > std::atan2 (const typename _-`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp >`  
`std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp >`  
`std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp >`  
`std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type >`  
`std::cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type >`  
`std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray<`  
`_Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray<`  
`_Tp > &__v)`

- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type >`  
`std::log (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`  
`std::log10 (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp >` `std::log10 (const`  
`valarray< _Tp > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type >` `std::operator!= (const typename _-`  
`Dom::value_type &_t, const _Expr< _Dom, typename _Dom::value_type >`  
`&_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`Dom::value_type >::result_type >` `std::operator!= (const _Expr< _Dom, type-`  
`name _Dom::value_type > &_e, const valarray< typename _Dom::value_type`  
`> &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type >` `std::operator!= (const valarray<`  
`typename _Dom::value_type > &_v, const _Expr< _Dom, typename`  
`_Dom::value_type > &_e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`  
`std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &_v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`Dom::value_type >::result_type >` `std::operator!= (const _Expr< _Dom, type-`  
`name _Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_`  
`type >::result_type >` `std::operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __modulus, typename _Dom1::value_type >::result_type >`

- std::operator%** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_modulus, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_modulus, typename \_Dom::value\_type >::result\_type > **std::operator%** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_modulus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_modulus, typename \_Dom::value\_type >::result\_type > **std::operator%** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_modulus, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_modulus, typename \_Dom::value\_type >::result\_type > **std::operator%** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **std::operator&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **std::operator&** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **std::operator&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom::value\_type >::result\_type > **std::operator&** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
  - template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_bitwise\_and, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_and, typename \_Dom1::value\_type >::result\_type > **std::operator&** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)

- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __logical_and, typename _Dom1::value_type >::result_type >`  
`std::operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &_`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > std::operator&& (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > std::operator&& (const _Expr< _Dom, typename _`  
`Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > std::operator&& (const valarray< typename _Dom::value_`  
`type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > std::operator&& (const _Expr< _Dom, typename _`  
`Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __multiplies, typename _Dom1::value_type >::result_type >`  
`std::operator* (const _Expr< _Dom1, typename _Dom1::value_type > &_v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`  
`type >::result_type > std::operator* (const _Expr< _Dom, typename _`  
`Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`  
`>::result_type > std::operator* (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`  
`type >::result_type > std::operator* (const _Expr< _Dom, typename _`

- Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_multiplies, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_plus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_plus, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_plus, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
  - template<class \_Dom1, class \_Dom2 >  
 \_Expr< \_BinClos< \_\_plus, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_plus, typename \_Dom1::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_plus, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_minus, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **std::operator-** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_minus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **std::operator-** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
  - template<class \_Dom >  
 \_Expr< \_BinClos< \_\_minus, \_ValArray, \_Expr, typename \_Dom::value\_type,

```

_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_
type > std::operator- (const valarray< typename _Dom::value_type > &__v,
const _Expr< _Dom, typename _Dom::value_type > &__e)

```

- ```

template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __
fun< __minus, typename _Dom1::value_type >::result_type > std::operator-
(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
_Dom2, typename _Dom2::value_type > &__w)

```
- ```

template<class _Dom >
_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _
Dom::value_type >, typename __fun< __minus, typename _Dom::value_
type >::result_type > std::operator- (const _Expr< _Dom, typename _
Dom::value_type > &__v, const typename _Dom::value_type &__t)

```
- ```

template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __
fun< __divides, typename _Dom1::value_type >::result_type > std::operator/
(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
_Dom2, typename _Dom2::value_type > &__w)

```
- ```

template<class _Dom >
_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _
Dom::value_type >, typename __fun< __divides, typename _Dom::value_
type >::result_type > std::operator/ (const _Expr< _Dom, typename _
Dom::value_type > &__v, const typename _Dom::value_type &__t)

```
- ```

template<class _Dom >
_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_
type > std::operator/ (const typename _Dom::value_type &__t, const _Expr<
_Dom, typename _Dom::value_type > &__v)

```
- ```

template<class _Dom >
_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _
Dom::value_type >, typename __fun< __divides, typename _Dom::value_
type >::result_type > std::operator/ (const _Expr< _Dom, typename _
Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__
_v)

```
- ```

template<class _Dom >
_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_
type > std::operator/ (const valarray< typename _Dom::value_type > &__v,
const _Expr< _Dom, typename _Dom::value_type > &__e)

```
- ```

template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun<
__less, typename _Dom1::value_type >::result_type > std::operator< (const
_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,
typename _Dom2::value_type > &__w)

```

- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v,`  
`const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _`  
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const typename _Dom::value_type &__t, const _Expr< _`  
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e,`  
`const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _`  
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const valarray< typename _Dom::value_type > &__v, const`  
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`  
`std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`  
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > std::operator<< (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`  
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > std::operator<< (const valarray< typename _Dom::value_`  
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`  
`std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &_`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`  
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _-`  
`Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`  
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _-`  
`Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`  
`>::result_type > std::operator<= (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`  
`>::result_type > std::operator<= (const valarray< typename _Dom::value_`  
`type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_`  
`type >::result_type > std::operator== (const _Expr< _Dom, typename _-`  
`Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`  
`>::result_type > std::operator== (const valarray< typename _Dom::value_`  
`type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`  
`>::result_type > std::operator== (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __equal_to, typename _Dom1::value_type >::result_type >`

**std::operator==** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_w)

- template<class \_Dom >  
\_Expr< \_BinClos< \_\_equal\_to, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_equal\_to, typename \_Dom::value\_type >::result\_type > **std::operator==** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v, const typename \_Dom::value\_type &\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_type > **std::operator>** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v, const typename \_Dom::value\_type &\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_type > **std::operator>** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_e, const valarray< typename \_Dom::value\_type > &\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_type > **std::operator>** (const valarray< typename \_Dom::value\_type > &\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_e)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_greater, typename \_Dom1::value\_type >::result\_type > **std::operator>** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_type > **std::operator>** (const typename \_Dom::value\_type &\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater\_equal, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_greater\_equal, typename \_Dom::value\_type >::result\_type > **std::operator>=** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_e, const valarray< typename \_Dom::value\_type > &\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater\_equal, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_greater\_equal, typename \_Dom::value\_type >::result\_type > **std::operator>=** (const typename \_Dom::value\_type &\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _-`  
`Dom::value_type >::result_type > std::operator>= (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`  
`> &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`  
`std::operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &_-`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __greater_equal, typename _-`  
`Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &_-`  
`_t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`  
`std::operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &_-`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_-`  
`type >::result_type > std::operator>> (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`  
`>::result_type > std::operator>> (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_-`  
`type >::result_type > std::operator>> (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`  
`>::result_type > std::operator>> (const valarray< typename _Dom::value_-`  
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`

>::result\_type > **std::operator**<sup>^</sup> (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)

- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom1::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom1::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`  
`>::result_type > std::operator| (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > std::operator|| (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > std::operator|| (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`  
`std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`  
`Dom >, typename _Dom::value_type > std::pow (const valarray< typename`  
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename _Dom::value_type > std::pow (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow`  
`(const valarray< _Tp > &__v, const _Tp &__t)`

- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`  
`name _Dom::value_type > &_e, const typename _Dom::value_type &_t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow`  
`(const valarray< _Tp > &_v, const valarray< _Tp > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`  
`name _Dom::value_type > &_e, const valarray< typename _Dom::value_type`  
`> &_v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name _Dom1::value_type > std::pow (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &_e1, const _Expr< _Dom2, typename _Dom2::value_`  
`type > &_e2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow`  
`(const _Tp &_t, const valarray< _Tp > &_v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sin (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray<`  
`_Tp > &_v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sinh (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray<`  
`_Tp > &_v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray<`  
`_Tp > &_v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray<`  
`_Tp > &_v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type >`  
`std::tan (const _Expr< _Dom, typename _Dom::value_type > &_e)`

- `template<class _Dom >`  
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray<`  
`_Tp > &__v)`

### 6.351.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_after.h](#).

## 6.352 `valarray_array.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`  
`restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t`  
`__s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__-`  
`restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _-`  
`Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _-`  
`Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`  
`__b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t`  
`> __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`  
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const`  
`size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
  
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s,`  
`_Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __-`  
`b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`  
`__b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t`  
`__n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const`  
`size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n,`  
`size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp`  
`*__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`

- `template<typename _Tp >`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `size_t __-`  
`n`, `_Array< _Tp > __b`, `_Array< size_t > __i`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `size_t __-`  
`n`, `const _Tp &__t`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `size_t __-`  
`n`, `_Array< _Tp > __b`)
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `const _-`  
`Expr< _Dom, _Tp > &__e`, `size_t __n`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `size_t __-`  
`n`, `size_t __s`, `_Array< _Tp > __b`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `_Array<`  
`_Tp > __b`, `size_t __n`, `size_t __s`)
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `size_t __-`  
`s`, `const Expr< _Dom, _Tp > &__e`, `size_t __n`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_and` (`_Array< _Tp > __a`, `_Array<`  
`size_t > __i`, `_Array< _Tp > __b`, `size_t __n`)
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `_Array<`  
`bool > __m`, `const Expr< _Dom, _Tp > &__e`, `size_t __n`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `size_t __n`,  
`const _Tp &__t`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `size_t __n`,  
`_Array< _Tp > __b`)
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `const _-`  
`Expr< _Dom, _Tp > &__e`, `size_t __n`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `size_t __n`,  
`size_t __s`, `_Array< _Tp > __b`)
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `_Array< _-`  
`Tp > __b`, `size_t __n`, `size_t __s`)
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_or` (`_Array< _Tp > __a`, `size_t __s`,  
`const Expr< _Dom, _Tp > &__e`, `size_t __n`)

- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n, _`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n, _`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, size_t __n, _`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___divides (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___minus (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _`  
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _-`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s,`  
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, const Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

### 6.352.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_array.h](#).

## 6.353 valarray\_array.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _-`  
`Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t`  
`__n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t`  
`__n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`  
`Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`  
`Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`  
`Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`  
`Array< _Tp > __a, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`  
`__b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e,`  
`size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool >`  
`__m, const _Tp &__t)`

### 6.353.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_array.tcc](#).

## 6.354 `valarray_before.h` File Reference

### Namespaces

- namespace [std](#)

### 6.354.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_before.h](#).

## 6.355 `vector` File Reference

### 6.355.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

## 6.356 `vector` File Reference

### Classes

- class [std::\\_\\_debug::vector<\\_Tp, \\_Allocator>](#)  
*Class [std::vector](#) with safety/checking/debug instrumentation.*
- struct [std::hash<\\_\\_debug::vector<bool, \\_Alloc>>](#)  
*[std::hash](#) specialization for [vector<bool>](#).*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::\_\_debug::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::\_\_debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

### 6.356.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

## 6.357 vector File Reference

### Classes

- struct [std::hash< \\_\\_profile::vector< bool, \\_Alloc > >](#)  
*[std::hash](#) specialization for `vector<bool>`.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

### 6.357.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).

## 6.358 vector.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.358.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vector.tcc](#).

## 6.359 vstring.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Alloc](#), [\\_Base](#) >

*Template class [\\_\\_versa\\_string](#).*

*Data structure managing sequences of characters and character-like objects.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>  
basic_istream< \_CharT, \_Traits > & std::getline (basic_istream< \_CharT, \_Traits > &\_is, \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_str, \_CharT \_\_delim)`
- `template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>  
basic_istream< \_CharT, \_Traits > & std::getline (basic_istream< \_CharT, \_Traits > &\_is, \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_str)`
- `template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>`

- ```
bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _-
CharT, _Traits, _Alloc, _Base > &__rhs)
```
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-
Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-
Base > &__lhs, const _CharT *__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-
rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa _-
string< _CharT, _Traits, _Alloc, _Base > &__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_-
CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-
Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  

```
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-
Base > &__lhs, const _CharT *__rhs)
```
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- ```
bool \_\_gnu\_cxx::operator< (const _CharT *__lhs, const __versa_string< _-
CharT, _Traits, _Alloc, _Base > &__rhs)
```
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__os, const \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits,`  
`_Alloc, _Base > &__str)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`  
`__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > &__lhs, const _CharT *__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator<= (const _CharT *__lhs, const __versa_string< _-`  
`CharT, _Traits, _Alloc, _Base > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator== (const _CharT *__lhs, const __versa_string< _-`  
`CharT, _Traits, _Alloc, _Base > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const _CharT *__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`  
`__rhs)`
  - `template<typename _CharT, template< typename, typename, typename > class _Base>`  
`\_\_enable\_if< std::is\_char< \_CharT >::\_\_value, bool >::\_\_type \\_\\_gnu\\_-`  
`cxx::operator== (const __versa_string< _CharT, std::char\_traits< _CharT >,`  
`std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT,`  
`std::char\_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool \_\_gnu\_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _-`  
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`  
`__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool [\\_\\_gnu\\_cxx::operator](#)> (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator](#)> (const \_CharT \*\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - bool [\\_\\_gnu\\_cxx::operator>=](#) (const \_CharT \*\_\_lhs, const \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - basic\_istream< \_CharT, \_Traits > & [std::operator>>](#) (basic\_istream< \_CharT, \_Traits > &\_\_is, [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_str)
- double [\\_\\_gnu\\_cxx::stod](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- float [\\_\\_gnu\\_cxx::stof](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- int [\\_\\_gnu\\_cxx::stoi](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- long [\\_\\_gnu\\_cxx::stol](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- long double [\\_\\_gnu\\_cxx::stold](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0)
- long long [\\_\\_gnu\\_cxx::stoll](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long [\\_\\_gnu\\_cxx::stoul](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long long [\\_\\_gnu\\_cxx::stoull](#) (const \_\_vstring &\_\_str, std::size\_t \*\_\_idx, int \_\_base=10)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>
  - void [\\_\\_gnu\\_cxx::swap](#) (\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, \_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (long long \_\_val)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (unsigned \_\_val)
- \_\_vstring [\\_\\_gnu\\_cxx::to\\_string](#) (int \_\_val)

- `__vstring __gnu_cxx::to_string` (long `__val`)
- `__vstring __gnu_cxx::to_string` (unsigned long `__val`)
- `__vstring __gnu_cxx::to_string` (float `__val`)
- `__vstring __gnu_cxx::to_string` (unsigned long long `__val`)
- `__vstring __gnu_cxx::to_string` (long double `__val`)
- `__vstring __gnu_cxx::to_string` (double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (float `__val`)
- `__wvstring __gnu_cxx::to_wstring` (int `__val`)

### 6.359.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

## 6.360 `vstring.tcc` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (_CharT __lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const  
_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,  
typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const  
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,  
typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const  
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,  
typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const  
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_  
string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,  
typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,  
_Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
> &__str)`

### 6.360.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring.tcc](#).

## 6.361 vstring\_fwd.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Typedefs

- typedef `__versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > __gnu_cxx::__rc_string`
- typedef `__vstring __gnu_cxx::__sso_string`

- typedef `__versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > __gnu_cxx::__u16rc_string`
- typedef `__u16vstring __gnu_cxx::__u16sso_string`
- typedef `__versa_string< char16_t > __gnu_cxx::__u16vstring`
- typedef `__versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > __gnu_cxx::__u32rc_string`
- typedef `__u32vstring __gnu_cxx::__u32sso_string`
- typedef `__versa_string< char32_t > __gnu_cxx::__u32vstring`
- typedef `__versa_string< char > __gnu_cxx::__vstring`
- typedef `__versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > __gnu_cxx::__wrc_string`
- typedef `__wvstring __gnu_cxx::__wsso_string`
- typedef `__versa_string< wchar_t > __gnu_cxx::__wvstring`

### 6.361.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring\\_fwd.h](#).

## 6.362 `vstring_util.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.362.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring\\_util.h](#).

## 6.363 `workstealing.h` File Reference

Parallelization of embarrassingly parallel execution by means of work-stealing.

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job<\\_DifferenceTp >](#)  
*One \_\_job for a certain thread.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- #define `_GLIBCXX_JOB_VOLATILE`

## Functions

- template<typename [\\_RAIter](#) , typename [\\_Op](#) , typename [\\_Fu](#) , typename [\\_Red](#) , typename [\\_Result](#)  
>  
[\\_Op](#) [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing](#) ([\\_RAIter](#) [\\_\\_begin](#), [\\_RAIter](#) [\\_\\_end](#), [\\_Op](#) [\\_\\_op](#), [\\_Fu](#) & [\\_\\_f](#), [\\_Red](#) [\\_\\_r](#), [\\_Result](#) [\\_\\_base](#), [\\_Result](#) & [\\_\\_output](#), typename [std::iterator\\_traits<\\_RAIter >::difference\\_type](#) [\\_\\_bound](#))

### 6.363.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [workstealing.h](#).

# Index

- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase, 1111
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue, 1145
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 980
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string, 740
- ~auto\_ptr
  - std::auto\_ptr, 1438
- ~basic\_filebuf
  - std::basic\_filebuf, 1457
- ~basic\_fstream
  - std::basic\_fstream, 1498
- ~basic\_ifstream
  - std::basic\_ifstream, 1561
- ~basic\_ios
  - std::basic\_ios, 1612
- ~basic\_iostream
  - std::basic\_iostream, 1648
- ~basic\_istream
  - std::basic\_istream, 1709
- ~basic\_istreamream
  - std::basic\_istreamream, 1763
- ~basic\_ofstream
  - std::basic\_ofstream, 1816
- ~basic\_ostream
  - std::basic\_ostream, 1859
- ~basic\_ostringstream
  - std::basic\_ostringstream, 1904
- ~basic\_streambuf
  - std::basic\_streambuf, 1942
- ~basic\_string
  - std::basic\_string, 1970
- ~basic\_stringstream
  - std::basic\_stringstream, 2060
- ~collate
  - std::collate, 2191
- ~ctype
  - std::ctype< char >, 2231
  - std::ctype< wchar\_t >, 2246
- ~deque
  - std::deque, 2308
- ~facet
  - std::locale::facet, 2566
- ~forward\_list
  - std::forward\_list, 2370
- ~gslice
  - numeric\_arrays, 86
- ~ios\_base
  - std::ios\_base, 2483
- ~locale
  - std::locale, 2558
- ~messages
  - std::messages, 2616
- ~money\_get
  - std::money\_get, 2629
- ~money\_put
  - std::money\_put, 2634
- ~moneypunct
  - std::moneypunct, 2642
- ~num\_get
  - std::num\_get, 2721
- ~num\_put
  - std::num\_put, 2741
- ~numpunct
  - std::numpunct, 2787
- ~sentry
  - std::basic\_ostream::sentry, 1891
- ~stdio\_filebuf

- \_\_gnu\_cxx::stdio\_filebuf, 891
- ~temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer, 945
- ~time\_get
  - std::time\_get, 2933
- ~time\_put
  - std::time\_put, 2954
- ~type\_info
  - std::type\_info, 3005
- ~vector
  - std::vector, 3044
- /mnt/share/src/ Directory Reference, 296
- /mnt/share/src/gcc.svn-4\_5-branch/
  - Directory Reference, 282
- /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/ Directory Reference, 288
- /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/
  - Directory Reference, 278
- /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/doc/doxygen/ Directory Reference, 279
- /mnt/share/src/gcc.svn-4\_5-branch/libstdc++-v3/libsupc++/
  - Directory Reference, 289
- \_\_gnu\_parallel
  - parallel\_balanced, 365
  - parallel\_omp\_loop, 365
  - parallel\_omp\_loop\_static, 365
  - parallel\_taskqueue, 365
  - parallel\_unbalanced, 365
  - sequential, 365
- \_AlgorithmStrategy
  - \_\_gnu\_parallel, 365
- \_BALLOC\_ALIGN\_BYTES
  - bitmap\_allocator.h, 3134
- \_BinIndex
  - \_\_gnu\_parallel, 364
- \_Bit\_scan\_forward
  - \_\_gnu\_cxx, 324
- \_CASable
  - \_\_gnu\_parallel, 364
- \_CASable\_bits
  - \_\_gnu\_parallel, 412
- \_CASable\_mask
  - \_\_gnu\_parallel, 412
- \_Construct
  - std, 568
- \_DRandomShufflingGlobalData
  - \_\_gnu\_parallel::-
    - DRandomShufflingGlobalData, 1080
- \_Destroy
  - std, 569
- \_FindAlgorithm
  - \_\_gnu\_parallel, 365
- \_Find\_first
  - SGIextensions, 13
- \_Find\_next
  - SGIextensions, 13
- \_GLIBCXX\_ASSERTIONS
  - compiletime\_settings.h, 3167
- \_GLIBCXX\_ATOMIC\_PROPERTY
  - atomics, 202
- \_GLIBCXX\_BAL\_QUICKSORT
  - features.h, 3215
- \_GLIBCXX\_CALL
  - compiletime\_settings.h, 3167
- \_GLIBCXX\_DEBUG\_VERIFY
  - macros.h, 3275
- \_GLIBCXX\_DEPRECATED\_ATTR
  - binders, 120
- \_GLIBCXX\_DEQUE\_BUF\_SIZE
  - stl\_deque.h, 3402
- \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS
  - features.h, 3215
- \_GLIBCXX\_FIND\_EQUAL\_SPLIT
  - features.h, 3215
- \_GLIBCXX\_FIND\_GROWING\_BLOCKS
  - features.h, 3215
- \_GLIBCXX\_MERGESORT
  - features.h, 3216
- \_GLIBCXX\_PARALLEL\_CONDITION
  - settings.h, 3372
- \_GLIBCXX\_PARALLEL\_LENGTH
  - multiway\_merge.h, 3296
- \_GLIBCXX\_QUICKSORT
  - features.h, 3216

- 
- `_GLIBCXX_RANDOM_SHUFFLE_-`
    - `CONSIDER_L1`
    - `compiletime_settings.h`, 3168
  - `_GLIBCXX_RANDOM_SHUFFLE_-`
    - `CONSIDER_TLB`
    - `compiletime_settings.h`, 3168
  - `_GLIBCXX_SCALE_DOWN_FPU`
    - `compiletime_settings.h`, 3168
  - `_GLIBCXX_TREE_DYNAMIC_-`
    - `BALANCING`
    - `features.h`, 3216
  - `_GLIBCXX_TREE_FULL_COPY`
    - `features.h`, 3216
  - `_GLIBCXX_TREE_INITIAL_-`
    - `SPLITTING`
    - `features.h`, 3217
  - `_GLIBCXX_VERBOSE_LEVEL`
    - `compiletime_settings.h`, 3168
  - `_GLIBCXX_VOLATILE`
    - `partition.h`, 3320
    - `queue.h`, 3340
  - `_GuardedIterator`
    - `__gnu_parallel::_GuardedIterator`, 1089
  - `_LoserTreeBase`
    - `__gnu_parallel::_LoserTreeBase`, 1111
  - `_M_allocate_and_copy`
    - `std::vector`, 3045
  - `_M_allocate_single_object`
    - `__gnu_cxx::bitmap_allocator`, 809
  - `_M_attach`
    - `__gnu_debug::_Safe_iterator`, 963
    - `__gnu_debug::_Safe_iterator_base`, 973
  - `_M_attach_single`
    - `__gnu_debug::_Safe_iterator`, 963
    - `__gnu_debug::_Safe_iterator_base`, 973
  - `_M_attached_to`
    - `__gnu_debug::_Safe_iterator`, 963
    - `__gnu_debug::_Safe_iterator_base`, 973
  - `_M_begin`
    - `__gnu_parallel::_Piece`, 1132
  - `_M_bin_proc`
    - `__gnu_parallel::_-`
      - `DRandomShufflingGlobalData`, 1080
  - `_M_bins_begin`
    - `__gnu_parallel::_DRSSorterPU`, 1083
  - `_M_buf`
    - `__gnu_cxx::enc_filebuf`, 839
    - `__gnu_cxx::stdio_filebuf`, 912
    - `std::basic_filebuf`, 1477
  - `_M_buf_locale`
    - `__gnu_cxx::enc_filebuf`, 839
    - `__gnu_cxx::stdio_filebuf`, 912
    - `__gnu_cxx::stdio_sync_filebuf`, 939
    - `std::basic_filebuf`, 1477
    - `std::basic_streambuf`, 1958
    - `std::basic_stringbuf`, 2042
  - `_M_buf_size`
    - `__gnu_cxx::enc_filebuf`, 839
    - `__gnu_cxx::stdio_filebuf`, 912
    - `std::basic_filebuf`, 1477
  - `_M_can_compare`
    - `__gnu_debug::_Safe_iterator`, 963
    - `__gnu_debug::_Safe_iterator_base`, 973
  - `_M_clear`
    - `__gnu_cxx::free_list`, 849
  - `_M_comp`
    - `__gnu_parallel::_LoserTree`, 1105
    - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1109
    - `__gnu_parallel::_LoserTreeBase`, 1112
  - `_M_const_iterators`
    - `__gnu_debug::_Safe_sequence`, 978
    - `__gnu_debug::_Safe_sequence_base`, 982
    - `__gnu_debug::basic_string`, 1033
    - `std::_debug::bitset`, 1270
    - `std::_debug::deque`, 1275
    - `std::_debug::list`, 1281
    - `std::_debug::map`, 1287
    - `std::_debug::multimap`, 1292
    - `std::_debug::multiset`, 1297
    - `std::_debug::set`, 1303
    - `std::_debug::unordered_map`, 1308
-

- std::\_\_debug::unordered\_multimap, 1312
- std::\_\_debug::unordered\_multiset, 1317
- std::\_\_debug::unordered\_set, 1322
- std::\_\_debug::vector, 1327
- \_M\_create\_node
  - std::list, 2538
- \_M\_create\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 822
  - \_\_gnu\_cxx::stdio\_filebuf, 892
  - std::basic\_filebuf, 1457
- \_M\_data
  - std::\_List\_node, 1389
- \_M\_deallocate\_single\_object
  - \_\_gnu\_cxx::bitmap\_allocator, 809
- \_M\_dereferenceable
  - \_\_gnu\_debug::\_Safe\_iterator, 964
- \_M\_destroy\_pback
  - \_\_gnu\_cxx::enc\_filebuf, 822
  - \_\_gnu\_cxx::stdio\_filebuf, 892
  - std::basic\_filebuf, 1457
- \_M\_detach
  - \_\_gnu\_debug::\_Safe\_iterator, 964
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 973
- \_M\_detach\_all
  - \_\_gnu\_debug::\_Safe\_sequence, 977
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 981
  - \_\_gnu\_debug::basic\_string, 991
  - std::\_\_debug::bitset, 1269
  - std::\_\_debug::deque, 1274
  - std::\_\_debug::list, 1280
  - std::\_\_debug::map, 1285
  - std::\_\_debug::multimap, 1291
  - std::\_\_debug::multiset, 1296
  - std::\_\_debug::set, 1301
  - std::\_\_debug::unordered\_map, 1306
  - std::\_\_debug::unordered\_multimap, 1311
  - std::\_\_debug::unordered\_multiset, 1315
  - std::\_\_debug::unordered\_set, 1320
  - std::\_\_debug::vector, 1326
- \_M\_detach\_single
  - \_\_gnu\_debug::\_Safe\_iterator, 964
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 973
- \_M\_detach\_singular
  - \_\_gnu\_debug::\_Safe\_sequence, 977
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 981
  - \_\_gnu\_debug::basic\_string, 991
  - std::\_\_debug::bitset, 1269
  - std::\_\_debug::deque, 1274
  - std::\_\_debug::list, 1280
  - std::\_\_debug::map, 1285
  - std::\_\_debug::multimap, 1291
  - std::\_\_debug::multiset, 1296
  - std::\_\_debug::set, 1301
  - std::\_\_debug::unordered\_map, 1306
  - std::\_\_debug::unordered\_multimap, 1311
  - std::\_\_debug::unordered\_multiset, 1315
  - std::\_\_debug::unordered\_set, 1320
  - std::\_\_debug::vector, 1326
- \_M\_dist
  - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 1080
- \_M\_elements\_leftover
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1141
- \_M\_end
  - \_\_gnu\_parallel::\_Piece, 1132
- \_M\_ext\_buf
  - \_\_gnu\_cxx::enc\_filebuf, 839
  - \_\_gnu\_cxx::stdio\_filebuf, 913
  - std::basic\_filebuf, 1477
- \_M\_ext\_buf\_size
  - \_\_gnu\_cxx::enc\_filebuf, 839
  - \_\_gnu\_cxx::stdio\_filebuf, 913
  - std::basic\_filebuf, 1478
- \_M\_ext\_next
  - \_\_gnu\_cxx::enc\_filebuf, 840
  - \_\_gnu\_cxx::stdio\_filebuf, 913
  - std::basic\_filebuf, 1478
- \_M\_fill\_initialize
  - std::deque, 2309
- \_M\_finish\_iterator

- 
- \_\_gnu\_parallel::\_\_accumulate\_-  
selector, [1036](#)
  - \_\_gnu\_parallel::\_\_adjacent\_-  
difference\_selector, [1037](#)
  - \_\_gnu\_parallel::\_\_count\_if\_selector,  
[1044](#)
  - \_\_gnu\_parallel::\_\_count\_selector,  
[1046](#)
  - \_\_gnu\_parallel::\_\_fill\_selector, [1048](#)
  - \_\_gnu\_parallel::\_\_for\_each\_-  
selector, [1054](#)
  - \_\_gnu\_parallel::\_\_generate\_-  
selector, [1055](#)
  - \_\_gnu\_parallel::\_\_generic\_for\_-  
each\_selector, [1058](#)
  - \_\_gnu\_parallel::\_\_identity\_selector,  
[1059](#)
  - \_\_gnu\_parallel::\_\_inner\_product\_-  
selector, [1062](#)
  - \_\_gnu\_parallel::\_\_replace\_if\_-  
selector, [1071](#)
  - \_\_gnu\_parallel::\_\_replace\_selector,  
[1074](#)
  - \_\_gnu\_parallel::\_\_transform1\_-  
selector, [1075](#)
  - \_\_gnu\_parallel::\_\_transform2\_-  
selector, [1077](#)
  - \_M\_first
    - \_\_gnu\_parallel::\_\_Job, [1096](#)
  - \_M\_first\_insert
    - \_\_gnu\_parallel::\_\_LoserTree, [1105](#)
    - \_\_gnu\_parallel::\_\_LoserTree< false,  
\_Tp, \_Compare >, [1109](#)
    - \_\_gnu\_parallel::\_\_LoserTreeBase,  
[1112](#)
  - \_M\_gcount
    - std::basic\_fstream, [1543](#)
    - std::basic\_ifstream, [1596](#)
    - std::basic\_iostream, [1692](#)
    - std::basic\_istream, [1743](#)
    - std::basic\_istreamstream, [1798](#)
    - std::basic\_stringstream, [2104](#)
  - \_M\_get
    - \_\_gnu\_cxx::free\_list, [849](#)
  - \_M\_get\_distance
    - \_\_gnu\_debug::\_\_Safe\_iterator, [964](#)
  - \_M\_get\_mutex
    - \_\_gnu\_debug::\_\_Safe\_iterator, [964](#)
    - \_\_gnu\_debug::\_\_Safe\_iterator\_base,  
[973](#)
    - \_\_gnu\_debug::\_\_Safe\_sequence, [977](#)
    - \_\_gnu\_debug::\_\_Safe\_sequence\_-  
base, [981](#)
    - \_\_gnu\_debug::\_\_basic\_string, [991](#)
  - std::\_\_debug::bitset, [1269](#)
  - std::\_\_debug::deque, [1274](#)
  - std::\_\_debug::list, [1280](#)
  - std::\_\_debug::map, [1286](#)
  - std::\_\_debug::multimap, [1291](#)
  - std::\_\_debug::multiset, [1296](#)
  - std::\_\_debug::set, [1302](#)
  - std::\_\_debug::unordered\_map, [1306](#)
  - std::\_\_debug::unordered\_multimap,  
[1311](#)
  - std::\_\_debug::unordered\_multiset,  
[1316](#)
  - std::\_\_debug::unordered\_set, [1320](#)
  - std::\_\_debug::vector, [1326](#)
  - \_M\_get\_result
    - std::\_\_basic\_future, [1246](#)
    - std::future, [2402](#)
    - std::future< \_Res & >, [2405](#)
    - std::future< void >, [2407](#)
    - std::shared\_future, [2895](#)
    - std::shared\_future< \_Res & >, [2898](#)
    - std::shared\_future< void >, [2901](#)
  - \_M\_getloc
    - std::basic\_fstream, [1499](#)
    - std::basic\_ifstream, [1562](#)
    - std::basic\_ios, [1613](#)
    - std::basic\_iostream, [1648](#)
    - std::basic\_istream, [1710](#)
    - std::basic\_istreamstream, [1764](#)
    - std::basic\_ofstream, [1816](#)
    - std::basic\_ostream, [1859](#)
    - std::basic\_ostreamstream, [1904](#)
    - std::basic\_stringstream, [2060](#)
    - std::ios\_base, [2483](#)
  - \_M\_global
    - \_\_gnu\_parallel::\_\_QSBThreadLocal,  
[1141](#)
  - \_M\_in\_beg
-

- \_\_gnu\_cxx::enc\_filebuf, 840
- \_\_gnu\_cxx::stdio\_filebuf, 913
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 939
- std::basic\_filebuf, 1478
- std::basic\_streambuf, 1958
- std::basic\_stringbuf, 2042
- \_M\_in\_cur
  - \_\_gnu\_cxx::enc\_filebuf, 840
  - \_\_gnu\_cxx::stdio\_filebuf, 914
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 939
  - std::basic\_filebuf, 1478
  - std::basic\_streambuf, 1959
  - std::basic\_stringbuf, 2042
- \_M\_in\_end
  - \_\_gnu\_cxx::enc\_filebuf, 840
  - \_\_gnu\_cxx::stdio\_filebuf, 914
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 940
  - std::basic\_filebuf, 1479
  - std::basic\_streambuf, 1959
  - std::basic\_stringbuf, 2043
- \_M\_incrementable
  - \_\_gnu\_debug::\_Safe\_iterator, 965
- \_M\_initial
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1141
- \_M\_initialize\_map
  - std::\_Deque\_base, 1371
  - std::deque, 2309
- \_M\_insert
  - \_\_gnu\_cxx::free\_list, 849
- \_M\_invalidate
  - \_\_gnu\_debug::\_Safe\_iterator, 965
- \_M\_invalidate\_all
  - \_\_gnu\_debug::\_Safe\_sequence, 977
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 981
  - \_\_gnu\_debug::basic\_string, 991
  - std::\_debug::bitset, 1269
  - std::\_debug::deque, 1274
  - std::\_debug::list, 1280
  - std::\_debug::map, 1286
  - std::\_debug::multimap, 1291
  - std::\_debug::multiset, 1296
  - std::\_debug::set, 1302
  - std::\_debug::unordered\_map, 1307
- std::\_debug::unordered\_multimap, 1311
- std::\_debug::unordered\_multiset, 1316
- std::\_debug::unordered\_set, 1321
- std::\_debug::vector, 1326
- \_M\_invalidate\_if
  - \_\_gnu\_debug::\_Safe\_sequence, 977
  - \_\_gnu\_debug::basic\_string, 991
  - std::\_debug::deque, 1275
  - std::\_debug::list, 1281
  - std::\_debug::map, 1286
  - std::\_debug::multimap, 1291
  - std::\_debug::multiset, 1296
  - std::\_debug::set, 1302
  - std::\_debug::unordered\_map, 1307
  - std::\_debug::unordered\_multimap, 1311
  - std::\_debug::unordered\_multiset, 1316
  - std::\_debug::unordered\_set, 1321
  - std::\_debug::vector, 1327
- \_M\_invalidate\_single
  - \_\_gnu\_debug::\_Safe\_iterator, 965
- \_M\_is\_begin
  - \_\_gnu\_debug::\_Safe\_iterator, 966
- \_M\_is\_end
  - \_\_gnu\_debug::\_Safe\_iterator, 966
- \_M\_iterators
  - \_\_gnu\_debug::\_Safe\_sequence, 978
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 982
  - \_\_gnu\_debug::basic\_string, 1033
  - std::\_debug::bitset, 1270
  - std::\_debug::deque, 1276
  - std::\_debug::list, 1282
  - std::\_debug::map, 1287
  - std::\_debug::multimap, 1292
  - std::\_debug::multiset, 1297
  - std::\_debug::set, 1303
  - std::\_debug::unordered\_map, 1308
  - std::\_debug::unordered\_multimap, 1312
  - std::\_debug::unordered\_multiset, 1317
  - std::\_debug::unordered\_set, 1322

- 
- std::\_\_debug::vector, 1328
  - \_M\_key
    - \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, 1114
  - \_M\_last
    - \_\_gnu\_parallel::\_Job, 1096
  - \_M\_leftover\_parts
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1141
  - \_M\_load
    - \_\_gnu\_parallel::\_Job, 1096
  - \_M\_log\_k
    - \_\_gnu\_parallel::\_LoserTree, 1105
    - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1109
    - \_\_gnu\_parallel::\_LoserTreeBase, 1113
  - \_M\_losers
    - \_\_gnu\_parallel::\_LoserTree, 1105
    - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1109
    - \_\_gnu\_parallel::\_LoserTreeBase, 1113
  - \_M\_mode
    - \_\_gnu\_cxx::enc\_filebuf, 841
    - \_\_gnu\_cxx::stdio\_filebuf, 914
    - std::basic\_filebuf, 1479
    - std::basic\_stringbuf, 2043
  - \_M\_new\_elements\_at\_back
    - std::deque, 2310
  - \_M\_new\_elements\_at\_front
    - std::deque, 2310
  - \_M\_next
    - \_\_gnu\_debug::\_Safe\_iterator, 969
    - \_\_gnu\_debug::\_Safe\_iterator\_base, 974
  - \_M\_num\_bins
    - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 1081
  - \_M\_num\_bits
    - \_\_gnu\_parallel::\_DRandomShufflingGlobalData, 1081
  - \_M\_num\_threads
    - \_\_gnu\_parallel::\_DRSSorterPU, 1083
    - \_\_gnu\_parallel::\_PMWMSortingData, 1135
    - \_\_gnu\_parallel::\_QSBThreadLocal, 1141
  - \_M\_offsets
    - \_\_gnu\_parallel::\_PMWMSortingData, 1135
  - \_M\_out\_beg
    - \_\_gnu\_cxx::enc\_filebuf, 841
    - \_\_gnu\_cxx::stdio\_filebuf, 915
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 940
    - std::basic\_filebuf, 1479
    - std::basic\_streambuf, 1959
    - std::basic\_stringbuf, 2043
  - \_M\_out\_cur
    - \_\_gnu\_cxx::enc\_filebuf, 841
    - \_\_gnu\_cxx::stdio\_filebuf, 915
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 941
    - std::basic\_filebuf, 1480
    - std::basic\_streambuf, 1960
    - std::basic\_stringbuf, 2044
  - \_M\_out\_end
    - \_\_gnu\_cxx::enc\_filebuf, 842
    - \_\_gnu\_cxx::stdio\_filebuf, 915
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 941
    - std::basic\_filebuf, 1480
    - std::basic\_streambuf, 1960
    - std::basic\_stringbuf, 2044
  - \_M\_pback
    - \_\_gnu\_cxx::enc\_filebuf, 842
    - \_\_gnu\_cxx::stdio\_filebuf, 916
    - std::basic\_filebuf, 1481
  - \_M\_pback\_cur\_save
    - \_\_gnu\_cxx::enc\_filebuf, 842
    - \_\_gnu\_cxx::stdio\_filebuf, 916
    - std::basic\_filebuf, 1481
  - \_M\_pback\_end\_save
    - \_\_gnu\_cxx::enc\_filebuf, 842
    - \_\_gnu\_cxx::stdio\_filebuf, 916
    - std::basic\_filebuf, 1481
  - \_M\_pback\_init
    - \_\_gnu\_cxx::enc\_filebuf, 843
    - \_\_gnu\_cxx::stdio\_filebuf, 916
    - std::basic\_filebuf, 1481
-

- `_M_pieces`
  - `__gnu_parallel::_-`  
PMWMSortingData, 1135
- `_M_pop_back_aux`
  - `std::deque`, 2310
- `_M_pop_front_aux`
  - `std::deque`, 2310
- `_M_prior`
  - `__gnu_debug::_Safe_iterator`, 969
  - `__gnu_debug::_Safe_iterator_base`, 974
- `_M_push_back_aux`
  - `std::deque`, 2311
- `_M_push_front_aux`
  - `std::deque`, 2311
- `_M_range_check`
  - `std::deque`, 2311
  - `std::vector`, 3045
- `_M_range_initialize`
  - `std::deque`, 2311, 2312
- `_M_reading`
  - `__gnu_cxx::enc_filebuf`, 843
  - `__gnu_cxx::stdio_filebuf`, 917
  - `std::basic_filebuf`, 1482
- `_M_reallocate_map`
  - `std::deque`, 2312
- `_M_reserve_elements_at_back`
  - `std::deque`, 2313
- `_M_reserve_elements_at_front`
  - `std::deque`, 2313
- `_M_reserve_map_at_back`
  - `std::deque`, 2313
- `_M_reserve_map_at_front`
  - `std::deque`, 2313
- `_M_revalidate_singular`
  - `__gnu_debug::_Safe_sequence`, 977
  - `__gnu_debug::_Safe_sequence_-`  
base, 981
  - `__gnu_debug::basic_string`, 991
  - `std::__debug::bitset`, 1269
  - `std::__debug::deque`, 1275
  - `std::__debug::list`, 1281
  - `std::__debug::map`, 1286
  - `std::__debug::multimap`, 1291
  - `std::__debug::multiset`, 1297
  - `std::__debug::set`, 1302
  - `std::__debug::unordered_map`, 1307
  - `std::__debug::unordered_multimap`, 1312
  - `std::__debug::unordered_multiset`, 1316
  - `std::__debug::unordered_set`, 1321
  - `std::__debug::vector`, 1327
- `_M_samples`
  - `__gnu_parallel::_-`  
PMWMSortingData, 1136
- `_M_sd`
  - `__gnu_parallel::_DRSSorterPU`, 1083
- `_M_seed`
  - `__gnu_parallel::_DRSSorterPU`, 1084
- `_M_sequence`
  - `__gnu_debug::_Safe_iterator`, 969
  - `__gnu_debug::_Safe_iterator_base`, 974
- `_M_sequential_algorithm`
  - `__gnu_parallel::_adjacent_find_-`  
selector, 1039
  - `__gnu_parallel::_find_first_of_-`  
selector, 1050
  - `__gnu_parallel::_find_if_selector`, 1051
  - `__gnu_parallel::_mismatch_-`  
selector, 1065
- `_M_set_buffer`
  - `__gnu_cxx::enc_filebuf`, 822
  - `__gnu_cxx::stdio_filebuf`, 892
  - `std::basic_filebuf`, 1458
- `_M_set_node`
  - `std::_Deque_iterator`, 1373
- `_M_singular`
  - `__gnu_debug::_Safe_iterator`, 966
  - `__gnu_debug::_Safe_iterator_base`, 974
- `_M_source`
  - `__gnu_parallel::_-`  
DRandomShufflingGlobalData, 1081
  - `__gnu_parallel::_LoserTreeBase::_-`  
Loser, 1114

- \_\_gnu\_parallel::\_-
  - PMWMSSortingData, 1136
- \_M\_starts
  - \_\_gnu\_parallel::\_-
    - DRandomShufflingGlobalData, 1081
  - \_\_gnu\_parallel::\_-
    - PMWMSSortingData, 1136
- \_M\_sup
  - \_\_gnu\_parallel::\_LoserTreeBase::\_-
    - Loser, 1114
- \_M\_swap
  - \_\_gnu\_debug::\_Safe\_sequence, 978
  - \_\_gnu\_debug::\_Safe\_sequence\_-
    - base, 981
  - \_\_gnu\_debug::basic\_string, 992
  - std::\_\_debug::bitset, 1270
  - std::\_\_debug::deque, 1275
  - std::\_\_debug::list, 1281
  - std::\_\_debug::map, 1286
  - std::\_\_debug::multimap, 1292
  - std::\_\_debug::multiset, 1297
  - std::\_\_debug::set, 1302
  - std::\_\_debug::unordered\_map, 1307
  - std::\_\_debug::unordered\_multimap, 1312
  - std::\_\_debug::unordered\_multiset, 1316
  - std::\_\_debug::unordered\_set, 1321
  - std::\_\_debug::vector, 1327
- \_M\_temporaries
  - \_\_gnu\_parallel::\_-
    - DRandomShufflingGlobalData, 1082
- \_M\_temporary
  - \_\_gnu\_parallel::\_-
    - PMWMSSortingData, 1136
- \_M\_transfer\_iter
  - \_\_gnu\_debug::\_Safe\_sequence, 978
  - \_\_gnu\_debug::basic\_string, 992
  - std::\_\_debug::deque, 1275
  - std::\_\_debug::list, 1281
  - std::\_\_debug::map, 1287
  - std::\_\_debug::multimap, 1292
  - std::\_\_debug::multiset, 1297
  - std::\_\_debug::set, 1303
- std::\_\_debug::unordered\_map, 1307
- std::\_\_debug::unordered\_multimap, 1312
- std::\_\_debug::unordered\_multiset, 1316
- std::\_\_debug::unordered\_set, 1321
- std::\_\_debug::vector, 1327
- \_M\_use\_pointer
  - \_\_gnu\_parallel::\_LoserTreeTraits, 1124
- \_M\_version
  - \_\_gnu\_debug::\_Safe\_iterator, 970
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 975
  - \_\_gnu\_debug::\_Safe\_sequence, 979
  - \_\_gnu\_debug::\_Safe\_sequence\_-
    - base, 982
  - \_\_gnu\_debug::basic\_string, 1033
  - std::\_\_debug::bitset, 1270
  - std::\_\_debug::deque, 1276
  - std::\_\_debug::list, 1282
  - std::\_\_debug::map, 1287
  - std::\_\_debug::multimap, 1292
  - std::\_\_debug::multiset, 1298
  - std::\_\_debug::set, 1303
  - std::\_\_debug::unordered\_map, 1308
  - std::\_\_debug::unordered\_multimap, 1313
  - std::\_\_debug::unordered\_multiset, 1317
  - std::\_\_debug::unordered\_set, 1322
  - std::\_\_debug::vector, 1328
- \_M\_w
  - std::\_Base\_bitset, 1366
- \_M\_write
  - std::basic\_fstream, 1499
  - std::basic\_iostream, 1648
  - std::basic\_ofstream, 1817
  - std::basic\_ostream, 1860
  - std::basic\_ostringstream, 1905
  - std::basic\_stringstream, 2061
- \_MultiwayMergeAlgorithm
  - \_\_gnu\_parallel, 365
- \_Parallelism
  - \_\_gnu\_parallel, 365
- \_PartialSumAlgorithm

- \_\_gnu\_parallel, 365
- \_Piece
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1140
- \_PseudoSequence
  - \_\_gnu\_parallel::\_PseudoSequence, 1137
- \_QSBThreadLocal
  - \_\_gnu\_parallel::\_QSBThreadLocal, 1140
- \_RandomNumber
  - \_\_gnu\_parallel::\_RandomNumber, 1142
- \_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_-RestrictedBoundedConcurrentQueue, 1145
- \_Safe\_iterator
  - \_\_gnu\_debug::\_Safe\_iterator, 961, 962
- \_Safe\_iterator\_base
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 972
- \_SequenceIndex
  - \_\_gnu\_parallel, 364
- \_SortAlgorithm
  - \_\_gnu\_parallel, 365
- \_SplittingAlgorithm
  - \_\_gnu\_parallel, 366
- \_Temporary\_buffer
  - std::\_Temporary\_buffer, 1408
- \_ThreadIndex
  - \_\_gnu\_parallel, 364
- \_Unchecked\_flip
  - SGIextensions, 14
- \_Unchecked\_reset
  - SGIextensions, 14
- \_Unchecked\_set
  - SGIextensions, 14
- \_Unchecked\_test
  - SGIextensions, 14
- \_\_atomic0::atomic\_address, 707
- \_\_atomic0::atomic\_bool, 708
- \_\_atomic0::atomic\_flag, 709
- \_\_atomic2::atomic\_address, 709
- \_\_atomic2::atomic\_bool, 710
- \_\_atomic2::atomic\_flag, 711
- \_\_begin1\_iterator
  - \_\_gnu\_parallel::\_inner\_product\_selector, 1061
- \_\_begin2\_iterator
  - \_\_gnu\_parallel::\_inner\_product\_selector, 1062
- \_\_bins\_end
  - \_\_gnu\_parallel::\_DRSSorterPU, 1083
- \_\_bit\_allocate
  - \_\_gnu\_cxx::\_detail, 336
- \_\_bit\_free
  - \_\_gnu\_cxx::\_detail, 336
- \_\_calc\_borders
  - \_\_gnu\_parallel, 366
- \_\_check\_dereferenceable
  - \_\_gnu\_debug, 344
- \_\_check\_singular
  - \_\_gnu\_debug, 345
- \_\_check\_singular\_aux
  - \_\_gnu\_debug, 345
- \_\_check\_string
  - \_\_gnu\_debug, 345
- \_\_compare\_and\_swap
  - \_\_gnu\_parallel, 366
- \_\_compare\_and\_swap\_32
  - \_\_gnu\_parallel, 367
- \_\_compare\_and\_swap\_64
  - \_\_gnu\_parallel, 367
- \_\_ctype\_type
  - std::basic\_fstream, 1491
  - std::basic\_ifstream, 1556
  - std::basic\_ios, 1607
  - std::basic\_iostream, 1641
  - std::basic\_istream, 1704
  - std::basic\_istream, 1758
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1854
  - std::basic\_ostringstream, 1899
  - std::basic\_stringstream, 2053
- \_\_cxxabiv1::\_\_forced\_unwind, 711
- \_\_decode2
  - \_\_gnu\_parallel, 367
- \_\_delete\_min\_insert
  - \_\_gnu\_parallel::\_LoserTree, 1104

- 
- \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1107
  - \_\_determine\_samples
    - \_\_gnu\_parallel, 368
  - \_\_encode2
    - \_\_gnu\_parallel, 368
  - \_\_env\_t
    - \_\_gnu\_profile, 420
  - \_\_fetch\_and\_add
    - \_\_gnu\_parallel, 369
  - \_\_fetch\_and\_add\_32
    - \_\_gnu\_parallel, 369
  - \_\_fetch\_and\_add\_64
    - \_\_gnu\_parallel, 370
  - \_\_final\_insertion\_sort
    - std, 555
  - \_\_find
    - std, 556
  - \_\_find\_if
    - std, 556
  - \_\_find\_if\_not
    - std, 556, 557
  - \_\_find\_template
    - \_\_gnu\_parallel, 370–372
  - \_\_for\_each\_template\_random\_access
    - \_\_gnu\_parallel, 373
  - \_\_for\_each\_template\_random\_access\_ed
    - \_\_gnu\_parallel, 373
  - \_\_for\_each\_template\_random\_access\_omp\_loop
    - \_\_gnu\_parallel, 374
  - \_\_for\_each\_template\_random\_access\_omp\_loop\_static
    - \_\_gnu\_parallel, 375
  - \_\_for\_each\_template\_random\_access\_workstealing
    - \_\_gnu\_parallel, 375
  - \_\_gcd
    - std, 557
  - \_\_genrand\_bits
    - \_\_gnu\_parallel::\_RandomNumber, 1143
  - \_\_get\_\_global\_lock
    - \_\_gnu\_profile, 420
  - \_\_get\_min\_source
    - \_\_gnu\_parallel::\_LoserTree, 1104
  - \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1107
  - \_\_gnu\_parallel::\_LoserTreeBase, 1112
  - \_\_get\_num\_threads
    - \_\_gnu\_parallel::balanced\_quick\_sort\_tag, 1159
    - \_\_gnu\_parallel::balanced\_tag, 1161
    - \_\_gnu\_parallel::default\_parallel\_tag, 1163
    - \_\_gnu\_parallel::exact\_tag, 1166
    - \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, 1169
    - \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, 1171
    - \_\_gnu\_parallel::multiway\_mergesort\_tag, 1172
    - \_\_gnu\_parallel::omp\_loop\_static\_tag, 1174
    - \_\_gnu\_parallel::omp\_loop\_tag, 1175
    - \_\_gnu\_parallel::parallel\_tag, 1178
    - \_\_gnu\_parallel::quicksort\_tag, 1180
    - \_\_gnu\_parallel::sampling\_tag, 1181
    - \_\_gnu\_parallel::unbalanced\_tag, 1183
  - \_\_glibcxx\_check\_erase
    - macros.h, 3274
  - \_\_glibcxx\_check\_erase\_range
    - macros.h, 3274
  - \_\_glibcxx\_check\_heap\_pred
    - macros.h, 3274
  - \_\_glibcxx\_check\_insert
    - macros.h, 3274
  - \_\_glibcxx\_check\_insert\_range
    - macros.h, 3274
  - \_\_glibcxx\_check\_partitioned\_lower
    - macros.h, 3275
  - \_\_glibcxx\_check\_partitioned\_lower\_pred
    - macros.h, 3275
  - \_\_glibcxx\_check\_partitioned\_upper\_pred
    - macros.h, 3275
  - \_\_glibcxx\_check\_sorted\_pred
    - macros.h, 3275
  - \_\_gnu\_cxx, 301
-

- `_Bit_scan_forward`, 324
- `__static_pointer_cast`, 323
- `operator<`, 327, 328
- `operator<=`, 329, 330
- `operator>`, 332
- `operator>=`, 333, 334
- `operator+`, 325–327
- `operator==`, 330, 331
- `swap`, 334
- `__gnu_cxx::_Caster`, 793
- `__gnu_cxx::_Char_types`, 793
- `__gnu_cxx::_ExtPtr_allocator`, 794
- `__gnu_cxx::_Invalid_type`, 796
- `__gnu_cxx::_Pointer_adapter`, 796
- `__gnu_cxx::_Relative_pointer_impl`, 799
- `__gnu_cxx::_Relative_pointer_impl< const_Tp >`, 800
- `__gnu_cxx::_Std_pointer_impl`, 800
- `__gnu_cxx::_Unqualified_type`, 801
- `__gnu_cxx::__common_pool_policy`, 712
- `__gnu_cxx::__detail`, 335
  - `__bit_allocate`, 336
  - `__bit_free`, 336
  - `__num_bitmaps`, 336
  - `__num_blocks`, 337
- `__gnu_cxx::__detail::_Bitmap_counter`, 713
- `__gnu_cxx::__detail::_Ffit_finder`, 715
  - `argument_type`, 716
  - `result_type`, 716
- `__gnu_cxx::__detail::_mini_vector`, 712
- `__gnu_cxx::_mt_alloc`, 716
- `__gnu_cxx::_mt_alloc_base`, 718
- `__gnu_cxx::_per_type_pool_policy`, 719
- `__gnu_cxx::_pool< false >`, 720
- `__gnu_cxx::_pool< true >`, 721
- `__gnu_cxx::_pool_alloc`, 723
- `__gnu_cxx::_pool_alloc_base`, 725
- `__gnu_cxx::_pool_base`, 726
- `__gnu_cxx::_rc_string_base`, 728
- `__gnu_cxx::_scoped_lock`, 731
- `__gnu_cxx::_versa_string`, 731
  - `~__versa_string`, 740
  - `__versa_string`, 736–739
- `append`, 740–742
- `assign`, 743–746
  - `at`, 747
  - `back`, 748
  - `begin`, 748
  - `c_str`, 748
  - `capacity`, 749
  - `cbegin`, 749
  - `cend`, 749
  - `clear`, 749
  - `compare`, 750–753
  - `copy`, 754
  - `crbegin`, 754
  - `crend`, 755
  - `data`, 755
  - `empty`, 755
  - `end`, 755, 756
  - `erase`, 756, 757
  - `find`, 757–759
  - `find_first_not_of`, 759–761
  - `find_first_of`, 762, 763
  - `find_last_not_of`, 764, 765
  - `find_last_of`, 766, 767
  - `front`, 768
  - `get_allocator`, 768
  - `insert`, 769–773
  - `length`, 774
  - `max_size`, 774
  - `npos`, 792
  - `operator+=`, 774–776
  - `operator=`, 776, 777
  - `push_back`, 778
  - `rbegin`, 779
  - `rend`, 779
  - `replace`, 780–786
  - `reserve`, 787
  - `resize`, 788
  - `rfind`, 788–790
  - `shrink_to_fit`, 791
  - `size`, 791
  - `substr`, 791
  - `swap`, 792
- `__gnu_cxx::annotate_base`, 802
- `__gnu_cxx::array_allocator`, 803
- `__gnu_cxx::array_allocator_base`, 804
- `__gnu_cxx::binary_compose`, 805
  - `argument_type`, 807
  - `result_type`, 807

- \_\_gnu\_cxx::bitmap\_allocator, 807
  - \_M\_allocate\_single\_object, 809
  - \_M\_deallocate\_single\_object, 809
- \_\_gnu\_cxx::char\_traits, 810
- \_\_gnu\_cxx::character, 811
- \_\_gnu\_cxx::condition\_base, 812
- \_\_gnu\_cxx::constant\_binary\_fun, 813
- \_\_gnu\_cxx::constant\_unary\_fun, 814
- \_\_gnu\_cxx::constant\_void\_fun, 815
- \_\_gnu\_cxx::debug\_allocator, 816
- \_\_gnu\_cxx::enc\_filebuf, 817
  - \_M\_buf, 839
  - \_M\_buf\_locale, 839
  - \_M\_buf\_size, 839
  - \_M\_create\_pback, 822
  - \_M\_destroy\_pback, 822
  - \_M\_ext\_buf, 839
  - \_M\_ext\_buf\_size, 839
  - \_M\_ext\_next, 840
  - \_M\_in\_beg, 840
  - \_M\_in\_cur, 840
  - \_M\_in\_end, 840
  - \_M\_mode, 841
  - \_M\_out\_beg, 841
  - \_M\_out\_cur, 841
  - \_M\_out\_end, 842
  - \_M\_pback, 842
  - \_M\_pback\_cur\_save, 842
  - \_M\_pback\_end\_save, 842
  - \_M\_pback\_init, 843
  - \_M\_reading, 843
  - \_M\_set\_buffer, 822
  - \_\_streambuf\_type, 820
- char\_type, 821
- close, 823
- eback, 823
- egptr, 823
- epptr, 824
- gbump, 824
- getloc, 824
- gptr, 825
- imbue, 825
- in\_avail, 825
- int\_type, 821
- is\_open, 826
- off\_type, 821
- open, 826
- overflow, 827
- pbackfail, 827
- pbase, 828
- pbump, 828
- pos\_type, 821
- pptr, 829
- pubimbue, 829
- pubseekoff, 829
- pubseekpos, 830
- pubsetbuf, 830
- pubsync, 830
- sbumpc, 831
- seekoff, 831
- seekpos, 831
- setbuf, 832
- setg, 832
- setp, 833
- sgetc, 833
- sgetn, 833
- showmanyc, 834
- snextc, 834
- sputbackc, 835
- sputc, 835
- sputn, 835
- stossc, 836
- sungetc, 836
- sync, 836
- traits\_type, 822
- uflow, 837
- underflow, 837
- xsgetn, 838
- xsputn, 838
- \_\_gnu\_cxx::encoding\_char\_traits, 843
- \_\_gnu\_cxx::encoding\_state, 845
- \_\_gnu\_cxx::forced\_error, 846
  - what, 847
- \_\_gnu\_cxx::free\_list, 848
  - \_M\_clear, 849
  - \_M\_get, 849
  - \_M\_insert, 849
- \_\_gnu\_cxx::hash\_map, 849
- \_\_gnu\_cxx::hash\_multimap, 852
- \_\_gnu\_cxx::hash\_multiset, 854
- \_\_gnu\_cxx::hash\_set, 856
- \_\_gnu\_cxx::limit\_condition, 858

- \_\_gnu\_cxx::limit\_condition::always\_ -  
adjustor, 859
- \_\_gnu\_cxx::limit\_condition::limit\_ -  
adjustor, 859
- \_\_gnu\_cxx::limit\_condition::never\_ -  
adjustor, 860
- \_\_gnu\_cxx::malloc\_allocator, 860
- \_\_gnu\_cxx::new\_allocator, 861
- \_\_gnu\_cxx::project1st, 863
  - first\_argument\_type, 864
  - result\_type, 864
  - second\_argument\_type, 864
- \_\_gnu\_cxx::project2nd, 864
  - first\_argument\_type, 865
  - result\_type, 865
  - second\_argument\_type, 865
- \_\_gnu\_cxx::random\_condition, 866
- \_\_gnu\_cxx::random\_condition::always\_ -  
adjustor, 867
- \_\_gnu\_cxx::random\_condition::group\_ -  
adjustor, 867
- \_\_gnu\_cxx::random\_condition::never\_ -  
adjustor, 868
- \_\_gnu\_cxx::rb\_tree, 868
- \_\_gnu\_cxx::recursive\_init\_error, 871
  - what, 872
- \_\_gnu\_cxx::rope, 873
- \_\_gnu\_cxx::select1st, 879
  - argument\_type, 880
  - result\_type, 880
- \_\_gnu\_cxx::select2nd, 880
  - argument\_type, 881
  - result\_type, 881
- \_\_gnu\_cxx::slist, 881
- \_\_gnu\_cxx::stdio\_filebuf, 884
  - ~stdio\_filebuf, 891
  - \_M\_buf, 912
  - \_M\_buf\_locale, 912
  - \_M\_buf\_size, 912
  - \_M\_create\_pback, 892
  - \_M\_destroy\_pback, 892
  - \_M\_ext\_buf, 913
  - \_M\_ext\_buf\_size, 913
  - \_M\_ext\_next, 913
  - \_M\_in\_beg, 913
  - \_M\_in\_cur, 914
  - \_M\_in\_end, 914
  - \_M\_mode, 914
  - \_M\_out\_beg, 915
  - \_M\_out\_cur, 915
  - \_M\_out\_end, 915
  - \_M\_pback, 916
  - \_M\_pback\_cur\_save, 916
  - \_M\_pback\_end\_save, 916
  - \_M\_pback\_init, 916
  - \_M\_reading, 917
  - \_M\_set\_buffer, 892
  - \_\_streambuf\_type, 889
- char\_type, 889
- close, 892
- eback, 893
- egptr, 893
- epptr, 894
- fd, 894
- file, 894
- gbump, 895
- getloc, 895
- gpptr, 895
- imbue, 896
- in\_avail, 896
- int\_type, 889
- is\_open, 897
- off\_type, 890
- open, 897, 898
- overflow, 898
- pbackfail, 899
- pbase, 899
- pbump, 900
- pos\_type, 890
- pptr, 900
- pubimbue, 901
- pubseekoff, 901
- pubseekpos, 901
- pubsetbuf, 902
- pubsync, 902
- sbumpc, 902
- seekoff, 903
- seekpos, 903
- setbuf, 904
- setg, 904
- setp, 905
- sgetc, 905

- sgetc, 906
- showmanyc, 906
- snextc, 907
- sputbackc, 907
- sputc, 907
- sputn, 908
- stdio\_filebuf, 890, 891
- stossc, 908
- sungetc, 909
- sync, 909
- traits\_type, 890
- uflow, 909
- underflow, 910
- xsgetc, 911
- xsputn, 911
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 917
  - \_M\_buf\_locale, 939
  - \_M\_in\_beg, 939
  - \_M\_in\_cur, 939
  - \_M\_in\_end, 940
  - \_M\_out\_beg, 940
  - \_M\_out\_cur, 941
  - \_M\_out\_end, 941
  - \_\_streambuf\_type, 921
- char\_type, 921
- eback, 923
- egptr, 923
- epptr, 924
- file, 924
- gbump, 924
- getloc, 925
- gptr, 925
- imbue, 925
- in\_avail, 926
- int\_type, 922
- off\_type, 922
- overflow, 926
- pbackfail, 927
- pbase, 927
- pbump, 928
- pos\_type, 922
- pptr, 928
- pubimbue, 929
- pubseekoff, 929
- pubseekpos, 929
- pubsetbuf, 930
- pubsync, 930
- sbumpc, 930
- seekoff, 931
- seekpos, 931
- setbuf, 932
- setg, 932
- setp, 932
- sgetc, 933
- sgetc, 933
- showmanyc, 933
- snextc, 934
- sputbackc, 934
- sputc, 935
- sputn, 935
- stossc, 936
- sungetc, 936
- sync, 936
- traits\_type, 922
- uflow, 937
- underflow, 937
- xsgetc, 938
- xsputn, 938
- \_\_gnu\_cxx::subtractive\_rng, 942
  - argument\_type, 943
  - operator(), 943
  - result\_type, 943
  - subtractive\_rng, 943
- \_\_gnu\_cxx::temporary\_buffer, 944
  - ~temporary\_buffer, 945
  - begin, 946
  - end, 946
  - requested\_size, 946
  - size, 946
  - temporary\_buffer, 945
- \_\_gnu\_cxx::throw\_allocator\_base, 947
- \_\_gnu\_cxx::throw\_allocator\_limit, 948
- \_\_gnu\_cxx::throw\_allocator\_random, 950
- \_\_gnu\_cxx::throw\_value\_base, 951
- \_\_gnu\_cxx::throw\_value\_limit, 952
- \_\_gnu\_cxx::throw\_value\_random, 954
- \_\_gnu\_cxx::typelist, 337
  - apply\_generator, 337
- \_\_gnu\_cxx::unary\_compose, 955
  - argument\_type, 957
  - result\_type, 957

- 
- \_\_gnu\_debug, 338
    - \_\_check\_dereferenceable, 344
    - \_\_check\_singular, 345
    - \_\_check\_singular\_aux, 345
    - \_\_check\_string, 345
    - \_\_valid\_range, 346
    - \_\_valid\_range\_aux, 346
    - \_\_valid\_range\_aux2, 347
  - \_\_gnu\_debug::\_After\_nth\_from, 958
  - \_\_gnu\_debug::\_Not\_equal\_to, 958
  - \_\_gnu\_debug::\_Safe\_iterator, 959
    - \_M\_attach, 963
    - \_M\_attach\_single, 963
    - \_M\_attached\_to, 963
    - \_M\_can\_compare, 963
    - \_M\_dereferenceable, 964
    - \_M\_detach, 964
    - \_M\_detach\_single, 964
    - \_M\_get\_distance, 964
    - \_M\_get\_mutex, 964
    - \_M\_incrementable, 965
    - \_M\_invalidate, 965
    - \_M\_invalidate\_single, 965
    - \_M\_is\_begin, 966
    - \_M\_is\_end, 966
    - \_M\_next, 969
    - \_M\_prior, 969
    - \_M\_sequence, 969
    - \_M\_singular, 966
    - \_M\_version, 970
    - \_Safe\_iterator, 961, 962
    - base, 966
    - operator\_iterator, 967
    - operator\*, 967
    - operator++, 967
    - operator->, 968
    - operator--, 968
    - operator=, 969
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 970
    - \_M\_attach, 973
    - \_M\_attach\_single, 973
    - \_M\_attached\_to, 973
    - \_M\_can\_compare, 973
    - \_M\_detach, 973
    - \_M\_detach\_single, 973
    - \_M\_get\_mutex, 973
    - \_M\_next, 974
    - \_M\_prior, 974
    - \_M\_sequence, 974
    - \_M\_singular, 974
    - \_M\_version, 975
    - \_Safe\_iterator\_base, 972
  - \_\_gnu\_debug::\_Safe\_sequence, 975
    - \_M\_const\_iterators, 978
    - \_M\_detach\_all, 977
    - \_M\_detach\_singular, 977
    - \_M\_get\_mutex, 977
    - \_M\_invalidate\_all, 977
    - \_M\_invalidate\_if, 977
    - \_M\_iterators, 978
    - \_M\_revalidate\_singular, 977
    - \_M\_swap, 978
    - \_M\_transfer\_iter, 978
    - \_M\_version, 979
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 979
    - ~\_Safe\_sequence\_base, 980
    - \_M\_const\_iterators, 982
    - \_M\_detach\_all, 981
    - \_M\_detach\_singular, 981
    - \_M\_get\_mutex, 981
    - \_M\_invalidate\_all, 981
    - \_M\_iterators, 982
    - \_M\_revalidate\_singular, 981
    - \_M\_swap, 981
    - \_M\_version, 982
  - \_\_gnu\_debug::\_is\_same, 957
  - \_\_gnu\_debug::basic\_string, 982
    - \_M\_const\_iterators, 1033
    - \_M\_detach\_all, 991
    - \_M\_detach\_singular, 991
    - \_M\_get\_mutex, 991
    - \_M\_invalidate\_all, 991
    - \_M\_invalidate\_if, 991
    - \_M\_iterators, 1033
    - \_M\_revalidate\_singular, 991
    - \_M\_swap, 992
    - \_M\_transfer\_iter, 992
    - \_M\_version, 1033
  - append, 992–994
  - assign, 995–997
  - at, 998
  - begin, 998, 999
-

- c\_str, 999
- capacity, 999
- cbegin, 999
- cend, 999
- clear, 999
- compare, 1000–1002
- copy, 1003
- crbegin, 1003
- crend, 1003
- data, 1004
- empty, 1004
- end, 1004
- erase, 1004, 1005
- find, 1006, 1007
- find\_first\_not\_of, 1007, 1008
- find\_first\_of, 1009, 1010
- find\_last\_not\_of, 1011, 1012
- find\_last\_of, 1012–1014
- get\_allocator, 1014
- insert, 1014–1018
- length, 1019
- max\_size, 1019
- npos, 1033
- operator+=, 1019, 1020
- push\_back, 1021
- rbegin, 1021, 1022
- rend, 1022
- replace, 1022–1028
- reserve, 1029
- resize, 1029
- rfind, 1030, 1031
- shrink\_to\_fit, 1031
- size, 1032
- substr, 1032
- swap, 1032
- \_\_gnu\_internal, 347
- \_\_gnu\_parallel, 347
  - \_AlgorithmStrategy, 365
  - \_BinIndex, 364
  - \_CASable, 364
  - \_CASable\_bits, 412
  - \_CASable\_mask, 412
  - \_FindAlgorithm, 365
  - \_MultiwayMergeAlgorithm, 365
  - \_Parallelism, 365
  - \_PartialSumAlgorithm, 365
  - \_SequenceIndex, 364
  - \_SortAlgorithm, 365
  - \_SplittingAlgorithm, 366
  - \_ThreadIndex, 364
  - \_\_calc\_borders, 366
  - \_\_compare\_and\_swap, 366
  - \_\_compare\_and\_swap\_32, 367
  - \_\_compare\_and\_swap\_64, 367
  - \_\_decode2, 367
  - \_\_determine\_samples, 368
  - \_\_encode2, 368
  - \_\_fetch\_and\_add, 369
  - \_\_fetch\_and\_add\_32, 369
  - \_\_fetch\_and\_add\_64, 370
  - \_\_find\_template, 370–372
  - \_\_for\_each\_template\_random\_access, 373
  - \_\_for\_each\_template\_random\_access\_ed, 373
  - \_\_for\_each\_template\_random\_access\_omp\_loop, 374
  - \_\_for\_each\_template\_random\_access\_omp\_loop\_static, 375
  - \_\_for\_each\_template\_random\_access\_workstealing, 375
  - \_\_is\_sorted, 376
  - \_\_median\_of\_three\_iterators, 377
  - \_\_merge\_advance, 377
  - \_\_merge\_advance\_movc, 378
  - \_\_merge\_advance\_usual, 378
  - \_\_parallel\_merge\_advance, 379, 380
  - \_\_parallel\_nth\_element, 380
  - \_\_parallel\_partial\_sort, 381
  - \_\_parallel\_partial\_sum, 381
  - \_\_parallel\_partial\_sum\_basecase, 382
  - \_\_parallel\_partial\_sum\_linear, 382
  - \_\_parallel\_partition, 383
  - \_\_parallel\_random\_shuffle, 384
  - \_\_parallel\_random\_shuffle\_drs, 384
  - \_\_parallel\_random\_shuffle\_drs\_pu, 385
  - \_\_parallel\_sort, 385–389
  - \_\_parallel\_sort\_qs, 390
  - \_\_parallel\_sort\_qs\_conquer, 390
  - \_\_parallel\_sort\_qs\_divide, 391

- \_\_parallel\_sort\_qsb, 391
- \_\_parallel\_unique\_copy, 392
- \_\_qsb\_conquer, 393
- \_\_qsb\_divide, 393
- \_\_qsb\_local\_sort\_with\_helping, 394
- \_\_random\_number\_pow2, 394
- \_\_rd\_log2, 395
- \_\_round\_up\_to\_pow2, 395
- \_\_search\_template, 396
- \_\_sequential\_multiway\_merge, 396
- \_\_sequential\_random\_shuffle, 397
- \_\_shrink, 397
- \_\_shrink\_and\_double, 398
- \_\_yield, 398
- equally\_split, 398
- equally\_split\_point, 399
- list\_partition, 399
- max, 400
- min, 400
- multiseq\_partition, 400
- multiseq\_selection, 401
- multiway\_merge, 402
- multiway\_merge\_3\_variant, 404
- multiway\_merge\_4\_variant, 404
- multiway\_merge\_exact\_splitting, 405
- multiway\_merge\_loser\_tree, 406
- multiway\_merge\_loser\_tree\_-sentinel, 406
- multiway\_merge\_loser\_tree\_-unguarded, 407
- multiway\_merge\_sampling\_splitting, 408
- multiway\_merge\_sentinels, 408
- parallel\_multiway\_merge, 410
- parallel\_sort\_mwms, 411
- parallel\_sort\_mwms\_pu, 411
- \_\_gnu\_parallel::\_DRSSorterPU, 1082
  - \_M\_bins\_begin, 1083
  - \_M\_num\_threads, 1083
  - \_M\_sd, 1083
  - \_M\_seed, 1084
  - \_\_bins\_end, 1083
- \_\_gnu\_parallel::\_-
  - DRandomShufflingGlobalData, 1079
  - \_DRandomShufflingGlobalData, 1080
  - \_M\_bin\_proc, 1080
  - \_M\_dist, 1080
  - \_M\_num\_bins, 1081
  - \_M\_num\_bits, 1081
  - \_M\_source, 1081
  - \_M\_starts, 1081
  - \_M\_temporaries, 1082
  - \_\_gnu\_parallel::\_DummyReduct, 1084
  - \_\_gnu\_parallel::\_EqualFromLess, 1085
    - first\_argument\_type, 1086
    - result\_type, 1086
    - second\_argument\_type, 1086
  - \_\_gnu\_parallel::\_EqualTo, 1086
    - first\_argument\_type, 1088
    - result\_type, 1088
    - second\_argument\_type, 1088
  - \_\_gnu\_parallel::\_GuardedIterator, 1088
    - \_GuardedIterator, 1089
    - operator\_RAIter, 1089
    - operator<, 1090
    - operator<=, 1091
    - operator\*, 1090
    - operator++, 1090
  - \_\_gnu\_parallel::\_IteratorPair, 1091
    - first, 1093
    - first\_type, 1093
    - second, 1093
    - second\_type, 1093
  - \_\_gnu\_parallel::\_IteratorTriple, 1094
  - \_\_gnu\_parallel::\_Job, 1095
    - \_M\_first, 1096
    - \_M\_last, 1096
    - \_M\_load, 1096
  - \_\_gnu\_parallel::\_Less, 1096
    - first\_argument\_type, 1098
    - result\_type, 1098
    - second\_argument\_type, 1098
  - \_\_gnu\_parallel::\_Lexicographic, 1098
    - first\_argument\_type, 1100
    - result\_type, 1100
    - second\_argument\_type, 1100
  - \_\_gnu\_parallel::\_LexicographicReverse, 1100
    - first\_argument\_type, 1102

- result\_type, 1102
- second\_argument\_type, 1102
- \_\_gnu\_parallel::\_LoserTree, 1102
  - \_M\_comp, 1105
  - \_M\_first\_insert, 1105
  - \_M\_log\_k, 1105
  - \_M\_losers, 1105
  - \_\_delete\_min\_insert, 1104
  - \_\_get\_min\_source, 1104
  - \_\_insert\_start, 1104
- \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >, 1106
  - \_M\_comp, 1109
  - \_M\_first\_insert, 1109
  - \_M\_log\_k, 1109
  - \_M\_losers, 1109
  - \_\_delete\_min\_insert, 1107
  - \_\_get\_min\_source, 1107
  - \_\_init\_winner, 1108
  - \_\_insert\_start, 1108
- \_\_gnu\_parallel::\_LoserTreeBase, 1110
  - ~\_LoserTreeBase, 1111
  - \_LoserTreeBase, 1111
  - \_M\_comp, 1112
  - \_M\_first\_insert, 1112
  - \_M\_log\_k, 1113
  - \_M\_losers, 1113
  - \_\_get\_min\_source, 1112
  - \_\_insert\_start, 1112
- \_\_gnu\_parallel::\_LoserTreeBase::\_Loser, 1113
  - \_M\_key, 1114
  - \_M\_source, 1114
  - \_M\_sup, 1114
- \_\_gnu\_parallel::\_LoserTreePointer, 1115
- \_\_gnu\_parallel::\_LoserTreePointer< false, \_Tp, \_Compare >, 1116
- \_\_gnu\_parallel::\_LoserTreePointerBase, 1117
- \_\_gnu\_parallel::\_-
  - LoserTreePointerBase::\_Loser, 1118
- \_\_gnu\_parallel::\_-
  - LoserTreePointerUnguarded, 1119
- \_\_gnu\_parallel::\_-
  - LoserTreePointerUnguarded< false, \_Tp, \_Compare >, 1120
- \_\_gnu\_parallel::\_-
  - LoserTreePointerUnguardedBase, 1122
- \_\_gnu\_parallel::\_LoserTreeTraits, 1123
  - \_M\_use\_pointer, 1124
- \_\_gnu\_parallel::\_LoserTreeUnguarded, 1124
- \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare >, 1126
- \_\_gnu\_parallel::\_-
  - LoserTreeUnguardedBase, 1127
- \_\_gnu\_parallel::\_Multiplies, 1128
  - first\_argument\_type, 1130
  - result\_type, 1130
  - second\_argument\_type, 1130
- \_\_gnu\_parallel::\_Nothing, 1130
  - operator(), 1131
- \_\_gnu\_parallel::\_PMWMSortingData, 1134
  - \_M\_num\_threads, 1135
  - \_M\_offsets, 1135
  - \_M\_pieces, 1135
  - \_M\_samples, 1136
  - \_M\_source, 1136
  - \_M\_starts, 1136
  - \_M\_temporary, 1136
- \_\_gnu\_parallel::\_Piece, 1131
  - \_M\_begin, 1132
  - \_M\_end, 1132
- \_\_gnu\_parallel::\_Plus, 1132
  - first\_argument\_type, 1134
  - result\_type, 1134
  - second\_argument\_type, 1134
- \_\_gnu\_parallel::\_PseudoSequence, 1137
  - \_PseudoSequence, 1137
  - begin, 1138
  - end, 1138
- \_\_gnu\_parallel::\_-
  - PseudoSequenceIterator, 1138
- \_\_gnu\_parallel::\_QSBThreadLocal, 1139
  - \_M\_elements\_leftover, 1141

- `_M_global`, 1141
- `_M_initial`, 1141
- `_M_leftover_parts`, 1141
- `_M_num_threads`, 1141
- `_Piece`, 1140
- `_QSBThreadLocal`, 1140
- `__gnu_parallel::_RandomNumber`, 1142
- `_RandomNumber`, 1142
- `__genrand_bits`, 1143
- `operator()`, 1143
- `__gnu_parallel::_-`
  - `RestrictedBoundedConcurrentQueue`, 1143
  - `~RestrictedBoundedConcurrentQueue`, 1145
  - `_RestrictedBoundedConcurrentQueue`, 1145
  - `pop_back`, 1145
  - `pop_front`, 1145
  - `push_front`, 1145
- `__gnu_parallel::_SamplingSorter`, 1146
- `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`, 1147
- `__gnu_parallel::_Settings`, 1147
- `accumulate_minimal_n`, 1149
- `adjacent_difference_minimal_n`, 1149
- `cache_line_size`, 1150
- `count_minimal_n`, 1150
- `fill_minimal_n`, 1150
- `find_increasing_factor`, 1150
- `find_initial_block_size`, 1150
- `find_maximum_block_size`, 1150
- `find_sequential_search_size`, 1151
- `for_each_minimal_n`, 1151
- `generate_minimal_n`, 1151
- `get`, 1149
- `L1_cache_size`, 1151
- `L2_cache_size`, 1151
- `max_element_minimal_n`, 1151
- `merge_minimal_n`, 1152
- `merge_oversampling`, 1152
- `min_element_minimal_n`, 1152
- `multiway_merge_minimal_k`, 1152
- `multiway_merge_minimal_n`, 1152
- `multiway_merge_oversampling`, 1152
- `nth_element_minimal_n`, 1153
- `partial_sort_minimal_n`, 1153
- `partial_sum_dilation`, 1153
- `partial_sum_minimal_n`, 1153
- `partition_chunk_share`, 1153
- `partition_chunk_size`, 1153
- `partition_minimal_n`, 1154
- `qsb_steals`, 1154
- `random_shuffle_minimal_n`, 1154
- `replace_minimal_n`, 1154
- `search_minimal_n`, 1154
- `set`, 1149
- `set_difference_minimal_n`, 1154
- `set_intersection_minimal_n`, 1154
- `set_symmetric_difference_minimal_n`, 1155
- `set_union_minimal_n`, 1155
- `sort_minimal_n`, 1155
- `sort_mwms_oversampling`, 1155
- `sort_qs_num_samples_preset`, 1155
- `sort_qsb_base_case_maximal_n`, 1155
- `TLB_size`, 1156
- `transform_minimal_n`, 1156
- `unique_copy_minimal_n`, 1156
- `__gnu_parallel::_SplitConsistently`, 1156
- `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`, 1157
- `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`, 1158
- `__gnu_parallel::_accumulate_binop_reduct`, 1034
- `__gnu_parallel::_accumulate_selector`, 1034
- `_M_finish_iterator`, 1036
- `operator()`, 1035
- `__gnu_parallel::_adjacent_difference_selector`, 1036
- `_M_finish_iterator`, 1037
- `__gnu_parallel::_adjacent_find_selector`, 1038
- `_M_sequential_algorithm`, 1039

- operator(), 1039
- \_\_gnu\_parallel::\_\_binder1st, 1039
  - argument\_type, 1041
  - result\_type, 1041
- \_\_gnu\_parallel::\_\_binder2nd, 1041
  - argument\_type, 1043
  - result\_type, 1043
- \_\_gnu\_parallel::\_\_count\_if\_selector, 1043
  - \_M\_finish\_iterator, 1044
  - operator(), 1044
- \_\_gnu\_parallel::\_\_count\_selector, 1045
  - \_M\_finish\_iterator, 1046
  - operator(), 1046
- \_\_gnu\_parallel::\_\_fill\_selector, 1047
  - \_M\_finish\_iterator, 1048
  - operator(), 1048
- \_\_gnu\_parallel::\_\_find\_first\_of\_selector, 1048
  - \_M\_sequential\_algorithm, 1050
  - operator(), 1050
- \_\_gnu\_parallel::\_\_find\_if\_selector, 1050
  - \_M\_sequential\_algorithm, 1051
  - operator(), 1052
- \_\_gnu\_parallel::\_\_for\_each\_selector, 1052
  - \_M\_finish\_iterator, 1054
  - operator(), 1053
- \_\_gnu\_parallel::\_\_generate\_selector, 1054
  - \_M\_finish\_iterator, 1055
  - operator(), 1055
- \_\_gnu\_parallel::\_\_generic\_find\_selector, 1056
- \_\_gnu\_parallel::\_\_generic\_for\_each\_selector, 1056
  - \_M\_finish\_iterator, 1058
- \_\_gnu\_parallel::\_\_identity\_selector, 1058
  - \_M\_finish\_iterator, 1059
  - operator(), 1059
- \_\_gnu\_parallel::\_\_inner\_product\_selector, 1060
  - \_M\_finish\_iterator, 1062
  - \_\_begin1\_iterator, 1061
  - \_\_begin2\_iterator, 1062
  - \_\_inner\_product\_selector, 1061
  - operator(), 1061
- \_\_gnu\_parallel::\_\_max\_element\_reduct, 1062
- \_\_gnu\_parallel::\_\_min\_element\_reduct, 1063
- \_\_gnu\_parallel::\_\_mismatch\_selector, 1064
  - \_M\_sequential\_algorithm, 1065
  - operator(), 1065
- \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch, 1065
- \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch<true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1066
- \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch, 1067
- \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch<true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1067
- \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch, 1068
- \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch<false, \_\_stable, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1069
- \_\_gnu\_parallel::\_\_replace\_if\_selector, 1070
  - \_M\_finish\_iterator, 1071
  - \_\_new\_val, 1071
  - \_\_replace\_if\_selector, 1071
  - operator(), 1071
- \_\_gnu\_parallel::\_\_replace\_selector, 1072
  - \_M\_finish\_iterator, 1074
  - \_\_new\_val, 1073
  - \_\_replace\_selector, 1073
  - operator(), 1073
- \_\_gnu\_parallel::\_\_transform1\_selector, 1074

- `_M_finish_iterator`, 1075
- `operator()`, 1075
- `__gnu_parallel::__transform2_selector`, 1076
  - `_M_finish_iterator`, 1077
  - `operator()`, 1077
- `__gnu_parallel::__unary_negate`, 1077
  - `argument_type`, 1079
  - `result_type`, 1079
- `__gnu_parallel::balanced_quicksort_tag`, 1158
  - `__get_num_threads`, 1159
  - `set_num_threads`, 1159
- `__gnu_parallel::balanced_tag`, 1160
  - `__get_num_threads`, 1161
  - `set_num_threads`, 1161
- `__gnu_parallel::constant_size_blocks_tag`, 1161
- `__gnu_parallel::default_parallel_tag`, 1162
  - `__get_num_threads`, 1163
  - `set_num_threads`, 1163
- `__gnu_parallel::equal_split_tag`, 1164
- `__gnu_parallel::exact_tag`, 1165
  - `__get_num_threads`, 1166
  - `set_num_threads`, 1166
- `__gnu_parallel::find_tag`, 1166
- `__gnu_parallel::growing_blocks_tag`, 1167
- `__gnu_parallel::multiway_mergesort_exact_tag`, 1168
  - `__get_num_threads`, 1169
  - `set_num_threads`, 1169
- `__gnu_parallel::multiway_mergesort_sampling_tag`, 1170
  - `__get_num_threads`, 1171
  - `set_num_threads`, 1171
- `__gnu_parallel::multiway_mergesort_tag`, 1171
  - `__get_num_threads`, 1172
  - `set_num_threads`, 1172
- `__gnu_parallel::omp_loop_static_tag`, 1173
  - `__get_num_threads`, 1174
  - `set_num_threads`, 1174
- `__gnu_parallel::omp_loop_tag`, 1174
  - `__get_num_threads`, 1175
  - `set_num_threads`, 1175
- `__gnu_parallel::parallel_tag`, 1176
  - `__get_num_threads`, 1178
  - `parallel_tag`, 1178
  - `set_num_threads`, 1178
- `__gnu_parallel::quicksort_tag`, 1179
  - `__get_num_threads`, 1180
  - `set_num_threads`, 1180
- `__gnu_parallel::sampling_tag`, 1180
  - `__get_num_threads`, 1181
  - `set_num_threads`, 1181
- `__gnu_parallel::sequential_tag`, 1182
- `__gnu_parallel::unbalanced_tag`, 1182
  - `__get_num_threads`, 1183
  - `set_num_threads`, 1183
- `__gnu_pbds`, 412
  - `__gnu_pbds::associative_container_tag`, 1184
  - `__gnu_pbds::basic_hash_table`, 1185
  - `__gnu_pbds::basic_hash_tag`, 1186
  - `__gnu_pbds::basic_tree`, 1187
  - `__gnu_pbds::basic_tree_tag`, 1188
  - `__gnu_pbds::binary_heap_tag`, 1189
  - `__gnu_pbds::binomial_heap_tag`, 1190
  - `__gnu_pbds::cc_hash_table`, 1191
  - `__gnu_pbds::cc_hash_tag`, 1193
  - `__gnu_pbds::container_base`, 1194
  - `__gnu_pbds::container_tag`, 1195
  - `__gnu_pbds::container_traits`, 1196
  - `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`, 1196
  - `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >`, 1197
  - `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >`, 1198
  - `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >`, 1199
  - `__gnu_pbds::gp_hash_table`, 1200
  - `__gnu_pbds::gp_hash_tag`, 1202
  - `__gnu_pbds::list_update`, 1203
  - `__gnu_pbds::list_update_tag`, 1205

- 
- `__gnu_pbds::null_mapped_type`, 1206
  - `__gnu_pbds::ov_tree_tag`, 1207
  - `__gnu_pbds::pairing_heap_tag`, 1208
  - `__gnu_pbds::pat_trie_tag`, 1208
  - `__gnu_pbds::priority_queue_tag`, 1210
  - `__gnu_pbds::rb_tree_tag`, 1210
  - `__gnu_pbds::rc_binomial_heap_tag`, 1212
  - `__gnu_pbds::sequence_tag`, 1212
  - `__gnu_pbds::splay_tree_tag`, 1213
  - `__gnu_pbds::string_tag`, 1214
  - `__gnu_pbds::thin_heap_tag`, 1215
  - `__gnu_pbds::tree`, 1216
  - `__gnu_pbds::tree_tag`, 1218
  - `__gnu_pbds::trie`, 1219
  - `__gnu_pbds::trie_tag`, 1221
  - `__gnu_profile`, 415
    - `__env_t`, 420
    - `__get__global_lock`, 420
    - `__profcxx_init`, 420
    - `__report`, 421
  - `__gnu_profile::__container_size_info`, 1222
  - `__gnu_profile::__container_size_stack_info`, 1223
  - `__gnu_profile::__hashfunc_info`, 1224
  - `__gnu_profile::__hashfunc_stack_info`, 1226
  - `__gnu_profile::__list2vector_info`, 1227
  - `__gnu_profile::__map2umap_info`, 1228
  - `__gnu_profile::__map2umap_stack_info`, 1230
  - `__gnu_profile::__object_info_base`, 1231
  - `__gnu_profile::__reentrance_guard`, 1232
  - `__gnu_profile::__stack_hash`, 1233
  - `__gnu_profile::__stack_info_base`, 1233
  - `__gnu_profile::__trace_base`, 1234
  - `__gnu_profile::__trace_container_size`, 1235
  - `__gnu_profile::__trace_hash_func`, 1235
  - `__gnu_profile::__trace_hashtable_size`, 1237
  - `__gnu_profile::__trace_map2umap`, 1237
  - `__gnu_profile::__trace_vector_size`, 1239
  - `__gnu_profile::__trace_vector_to_list`, 1239
  - `__gnu_profile::__vector2list_info`, 1241
  - `__gnu_profile::__vector2list_stack_info`, 1242
  - `__gnu_profile::__warning_data`, 1244
  - `__gnu_sequential`, 421
  - `__heap_select`
    - std, 557
  - `__init_winner`
    - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1108
  - `__inner_product_selector`
    - `__gnu_parallel::__inner_product_selector`, 1061
  - `__inplace_stable_partition`
    - std, 558
  - `__inplace_stable_sort`
    - std, 558
  - `__insert_start`
    - `__gnu_parallel::_LoserTree`, 1104
    - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1108
    - `__gnu_parallel::_LoserTreeBase`, 1112
  - `__insertion_sort`
    - std, 558, 559
  - `__introsort_loop`
    - std, 559
  - `__invoke`
    - std, 639
  - `__is_sorted`
    - `__gnu_parallel`, 376
  - `__iterator_category`
    - iterators, 230
  - `__lg`
    - std, 559
  - `__median`
    - SGIextensions, 12, 13
  - `__median_of_three_iterators`
    - `__gnu_parallel`, 377
  - `__merge_adaptive`
    - std, 559, 560
  - `__merge_advance`
    - `__gnu_parallel`, 377
  - `__merge_advance_movc`
    - `__gnu_parallel`, 378
  - `__merge_advance_usual`
-

- \_\_gnu\_parallel, 378
- \_\_merge\_backward
  - std, 560
- \_\_merge\_without\_buffer
  - std, 561
- \_\_move\_median\_first
  - std, 561, 562
- \_\_new\_val
  - \_\_gnu\_parallel::\_\_replace\_if\_-selector, 1071
  - \_\_gnu\_parallel::\_\_replace\_selector, 1073
- \_\_num\_bitmaps
  - \_\_gnu\_cxx::\_\_detail, 336
- \_\_num\_blocks
  - \_\_gnu\_cxx::\_\_detail, 337
- \_\_num\_get\_type
  - std::basic\_fstream, 1491
  - std::basic\_ifstream, 1556
  - std::basic\_ios, 1607
  - std::basic\_iostream, 1641
  - std::basic\_istream, 1704
  - std::basic\_istream, 1758
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1854
  - std::basic\_ostringstream, 1899
  - std::basic\_stringstream, 2053
- \_\_num\_put\_type
  - std::basic\_fstream, 1491
  - std::basic\_ifstream, 1556
  - std::basic\_ios, 1607
  - std::basic\_iostream, 1641, 1642
  - std::basic\_istream, 1705
  - std::basic\_istream, 1758
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1854
  - std::basic\_ostringstream, 1899
  - std::basic\_stringstream, 2053
- \_\_parallel\_merge\_advance
  - \_\_gnu\_parallel, 379, 380
- \_\_parallel\_nth\_element
  - \_\_gnu\_parallel, 380
- \_\_parallel\_partial\_sort
  - \_\_gnu\_parallel, 381
- \_\_parallel\_partial\_sum
  - \_\_gnu\_parallel, 381
- \_\_parallel\_partial\_sum\_basecase
  - \_\_gnu\_parallel, 382
- \_\_parallel\_partial\_sum\_linear
  - \_\_gnu\_parallel, 382
- \_\_parallel\_partition
  - \_\_gnu\_parallel, 383
- \_\_parallel\_random\_shuffle
  - \_\_gnu\_parallel, 384
- \_\_parallel\_random\_shuffle\_drs
  - \_\_gnu\_parallel, 384
- \_\_parallel\_random\_shuffle\_drs\_pu
  - \_\_gnu\_parallel, 385
- \_\_parallel\_sort
  - \_\_gnu\_parallel, 385–389
- \_\_parallel\_sort\_qs
  - \_\_gnu\_parallel, 390
- \_\_parallel\_sort\_qs\_conquer
  - \_\_gnu\_parallel, 390
- \_\_parallel\_sort\_qs\_divide
  - \_\_gnu\_parallel, 391
- \_\_parallel\_sort\_qsb
  - \_\_gnu\_parallel, 391
- \_\_parallel\_unique\_copy
  - \_\_gnu\_parallel, 392
- \_\_partition
  - std, 562
- \_\_profcxx\_init
  - \_\_gnu\_profile, 420
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, 393
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 393
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 394
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 394
- \_\_rd\_log2
  - \_\_gnu\_parallel, 395
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_\_replace\_if\_-selector, 1071
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_\_replace\_selector, 1073
- \_\_report
  - \_\_gnu\_profile, 421

- \_\_reverse
  - std, 562, 563
- \_\_rotate
  - std, 563
- \_\_rotate\_adaptive
  - std, 563
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 395
- \_\_search\_n
  - std, 564
- \_\_search\_template
  - \_\_gnu\_parallel, 396
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 396
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 397
- \_\_shrink
  - \_\_gnu\_parallel, 397
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 398
- \_\_stable\_partition\_adaptive
  - std, 565
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 323
- \_\_streambuf\_type
  - \_\_gnu\_cxx::enc\_filebuf, 820
  - \_\_gnu\_cxx::stdio\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 921
  - std::basic\_filebuf, 1455
  - std::basic\_streambuf, 1940
  - std::basic\_stringbuf, 2022
- \_\_unguarded\_insertion\_sort
  - std, 565
- \_\_unguarded\_linear\_insert
  - std, 565, 566
- \_\_unguarded\_partition
  - std, 566
- \_\_unguarded\_partition\_pivot
  - std, 566, 567
- \_\_unique\_copy
  - std, 567, 568
- \_\_valid\_range
  - \_\_gnu\_debug, 346
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 346
- \_\_valid\_range\_aux2
  - \_\_gnu\_debug, 347
- \_\_verbose\_terminate\_handler
  - exceptions, 32
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string, 736–739
- \_\_yield
  - \_\_gnu\_parallel, 398
- a
  - std::extreme\_value\_distribution, 2356
  - std::weibull\_distribution, 3066
- abi, 421
- abs
  - complex\_numbers, 39
- accumulate
  - std, 569, 570
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1149
- acos
  - complex\_numbers, 40
- acosh
  - complex\_numbers, 40
- Adaptors for pointers to functions, 216
- Adaptors for pointers to members, 218
- adjacent\_difference
  - std, 570
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1149
- adjacent\_find
  - non\_mutating\_algorithms, 147
- adjustfield
  - std::basic\_fstream, 1543
  - std::basic\_ifstream, 1596
  - std::basic\_ios, 1627
  - std::basic\_iostream, 1692
  - std::basic\_istream, 1743
  - std::basic\_istream, 1798
  - std::basic\_ofstream, 1842
  - std::basic\_ostream, 1884
  - std::basic\_ostringstream, 1930
  - std::basic\_stringstream, 2104
  - std::ios\_base, 2489
- advance
  - std, 571

- algo.h, 3069
- algbase.h, 3083
- algorithm, 3085, 3087
- algorithmfwd.h, 3087, 3094
- Algorithms, 121
- all
  - std::bitset, 2133
  - std::locale, 2562
- all\_of
  - non\_mutating\_algorithms, 148
- allocate\_shared
  - pointer\_abstractions, 64
  - std::shared\_ptr, 2906
- allocator.h, 3107
- allocator\_type
  - std::set, 2876
- Allocators, 195
- alpha
  - std::gamma\_distribution, 2411
- any
  - std::bitset, 2133
- any\_of
  - non\_mutating\_algorithms, 148
- app
  - std::basic\_fstream, 1543
  - std::basic\_ifstream, 1597
  - std::basic\_ios, 1627
  - std::basic\_iostream, 1692
  - std::basic\_istream, 1743
  - std::basic\_istream, 1798
  - std::basic\_ofstream, 1842
  - std::basic\_ostream, 1884
  - std::basic\_ostringstream, 1930
  - std::basic\_stringstream, 2105
  - std::ios\_base, 2489
- append
  - \_\_gnu\_cxx::\_\_versa\_string, 740–742
  - \_\_gnu\_debug::basic\_string, 992–994
  - std::basic\_string, 1970–1973
- apply
  - numeric\_arrays, 86
- apply\_generator
  - \_\_gnu\_cxx::typelist, 337
- arg
  - complex\_numbers, 40
- argument\_type
  - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder, 716
  - \_\_gnu\_cxx::binary\_compose, 807
  - \_\_gnu\_cxx::select1st, 880
  - \_\_gnu\_cxx::select2nd, 881
  - \_\_gnu\_cxx::subtractive\_rng, 943
  - \_\_gnu\_cxx::unary\_compose, 957
  - \_\_gnu\_parallel::\_\_binder1st, 1041
  - \_\_gnu\_parallel::\_\_binder2nd, 1043
  - \_\_gnu\_parallel::\_\_unary\_negate, 1079
  - std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >, 1392
  - std::binder1st, 2119
  - std::binder2nd, 2121
  - std::const\_mem\_fun\_ref\_t, 2210
  - std::const\_mem\_fun\_t, 2212
  - std::hash, 2430
  - std::hash< \_\_debug::bitset< \_Nb > >, 2432
  - std::hash< \_\_debug::vector< bool, \_Alloc > >, 2434
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2436
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2438
  - std::hash< \_\_profile::bitset< \_Nb > >, 2440
  - std::hash< \_\_profile::vector< bool, \_Alloc > >, 2442
  - std::hash< \_Tp \* >, 2444
  - std::hash< error\_code >, 2446
  - std::hash< string >, 2448
  - std::hash< thread::id >, 2450
  - std::hash< u16string >, 2452
  - std::hash< u32string >, 2454
  - std::hash< wstring >, 2456
  - std::hash<::bitset< \_Nb > >, 2458
  - std::hash<::vector< bool, \_Alloc > >, 2460
  - std::logical\_not, 2574
  - std::mem\_fun\_ref\_t, 2610
  - std::mem\_fun\_t, 2612
  - std::negate, 2703

- std::pointer\_to\_unary\_function, 2832
- std::unary\_function, 3008
- std::unary\_negate, 3010
- Arithmetic Classes, 212
- array, 3107, 3108
- array\_allocator.h, 3109
- asin
  - complex\_numbers, 40
- asinh
  - complex\_numbers, 40
- assign
  - \_\_gnu\_cxx::\_\_versa\_string, 743–746
  - \_\_gnu\_debug::basic\_string, 995–997
  - std::basic\_string, 1973–1976
  - std::deque, 2314, 2315
  - std::forward\_list, 2371
  - std::list, 2538, 2539
  - std::vector, 3045, 3046
- assoc\_container.hpp, 3110
- assoc\_laguerre
  - tr1\_math\_spec\_func, 107
- assoc\_legendre
  - tr1\_math\_spec\_func, 107
- Associative, 25
- at
  - \_\_gnu\_cxx::\_\_versa\_string, 747
  - \_\_gnu\_debug::basic\_string, 998
  - std::basic\_string, 1976, 1977
  - std::deque, 2315
  - std::map, 2587
  - std::vector, 3046, 3047
- atan
  - complex\_numbers, 41
- atanh
  - complex\_numbers, 41
- ate
  - std::basic\_fstream, 1543
  - std::basic\_ifstream, 1597
  - std::basic\_ios, 1627
  - std::basic\_iostream, 1692
  - std::basic\_istream, 1743
  - std::basic\_istream, 1798
  - std::basic\_ofstream, 1842
  - std::basic\_ostream, 1885
  - std::basic\_ostringstream, 1930
  - std::basic\_stringstream, 2105
  - std::ios\_base, 2489
- atomic, 3111
- atomic\_0.h, 3115
- atomic\_2.h, 3116
- atomic\_base.h, 3116
- atomic\_char
  - atomics, 202
- atomic\_char16\_t
  - atomics, 202
- atomic\_char32\_t
  - atomics, 202
- atomic\_int
  - atomics, 203
- atomic\_llong
  - atomics, 203
- atomic\_long
  - atomics, 203
- atomic\_schar
  - atomics, 203
- atomic\_short
  - atomics, 203
- atomic\_uchar
  - atomics, 203
- atomic\_uint
  - atomics, 203
- atomic\_ullong
  - atomics, 204
- atomic\_ulong
  - atomics, 204
- atomic\_ushort
  - atomics, 204
- atomic\_wchar\_t
  - atomics, 204
- atomic\_word.h, 3118
- atomicfwd\_c.h, 3118
- atomicfwd\_cxx.h, 3120
- atomicity.h, 3120
- Atomics, 196
- atomics
  - \_GLIBCXX\_ATOMIC\_PROPERTY, 202
  - atomic\_char, 202
  - atomic\_char16\_t, 202

- atomic\_char32\_t, 202
- atomic\_int, 203
- atomic\_llong, 203
- atomic\_long, 203
- atomic\_schar, 203
- atomic\_short, 203
- atomic\_uchar, 203
- atomic\_uint, 203
- atomic\_ullong, 204
- atomic\_ulong, 204
- atomic\_ushort, 204
- atomic\_wchar\_t, 204
- kill\_dependency, 205
- memory\_order, 204
- auto\_ptr
  - std::auto\_ptr, 1437, 1438
- auto\_ptr.h, 3121
- b
  - std::extreme\_value\_distribution, 2356
  - std::weibull\_distribution, 3066
- back
  - \_\_gnu\_cxx::\_\_versa\_string, 748
  - std::deque, 2316
  - std::list, 2539
  - std::queue, 2846
  - std::vector, 3047
- back\_insert\_iterator
  - std::back\_insert\_iterator, 1444
- back\_inserter
  - iterators, 230
- bad
  - std::basic\_fstream, 1499
  - std::basic\_ifstream, 1562
  - std::basic\_ios, 1613
  - std::basic\_iostream, 1649
  - std::basic\_istream, 1710
  - std::basic\_istreamstream, 1764
  - std::basic\_ofstream, 1817
  - std::basic\_ostream, 1860
  - std::basic\_ostreamstream, 1905
  - std::basic\_stringstream, 2061
- badbit
  - std::basic\_fstream, 1544
  - std::basic\_ifstream, 1597
  - std::basic\_istream, 1628
  - std::basic\_iostream, 1692
  - std::basic\_istream, 1744
  - std::basic\_istreamstream, 1799
  - std::basic\_ofstream, 1843
  - std::basic\_ostream, 1885
  - std::basic\_ostreamstream, 1930
  - std::basic\_stringstream, 2105
  - std::ios\_base, 2490
- balanced\_quicksort.h, 3122
- base
  - \_\_gnu\_debug::\_Safe\_iterator, 966
  - std::discard\_block\_engine, 2332
  - std::independent\_bits\_engine, 2464
  - std::reverse\_iterator, 2867
  - std::shuffle\_order\_engine, 2910
- base.h, 3123, 3125
- basefield
  - std::basic\_fstream, 1544
  - std::basic\_ifstream, 1597
  - std::basic\_ios, 1628
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1744
  - std::basic\_istreamstream, 1799
  - std::basic\_ofstream, 1843
  - std::basic\_ostream, 1885
  - std::basic\_ostreamstream, 1930
  - std::basic\_stringstream, 2105
  - std::ios\_base, 2490
- basic\_file.h, 3125
- basic\_filebuf
  - std::basic\_filebuf, 1457
- basic\_fstream
  - std::basic\_fstream, 1497, 1498
- basic\_ifstream
  - std::basic\_ifstream, 1561
- basic\_ios
  - std::basic\_ios, 1612
- basic\_ios.h, 3126
- basic\_ios.tcc, 3126
- basic\_iostream
  - std::basic\_iostream, 1648
- basic\_istream
  - std::basic\_istream, 1709
- basic\_istreamstream
  - std::basic\_istreamstream, 1763

- basic\_iterator.h, 3127
- basic\_ofstream
  - std::basic\_ofstream, 1815, 1816
- basic\_ostream
  - std::basic\_ostream, 1859
- basic\_ostringstream
  - std::basic\_ostringstream, 1903
- basic\_streambuf
  - std::basic\_streambuf, 1942
- basic\_string
  - std::basic\_string, 1966–1969
- basic\_string.h, 3127
- basic\_string.tcc, 3131
- basic\_stringbuf
  - std::basic\_stringbuf, 2024
- basic\_stringstream
  - std::basic\_stringstream, 2059, 2060
- basic\_types.hpp, 3131
- before\_begin
  - std::forward\_list, 2372
- beg
  - std::basic\_fstream, 1544
  - std::basic\_ifstream, 1598
  - std::basic\_ios, 1628
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1744
  - std::basic\_istream, 1799
  - std::basic\_ofstream, 1843
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1931
  - std::basic\_stringstream, 2106
  - std::ios\_base, 2490
- begin
  - \_\_gnu\_cxx::\_\_versa\_string, 748
  - \_\_gnu\_cxx::temporary\_buffer, 946
  - \_\_gnu\_debug::basic\_string, 998, 999
  - \_\_gnu\_parallel::\_PseudoSequence, 1138
  - std::\_Temporary\_buffer, 1408
  - std::basic\_string, 1977, 1978
  - std::deque, 2316, 2317
  - std::forward\_list, 2372
  - std::list, 2539, 2540
  - std::map, 2588
  - std::multimap, 2669
  - std::multiset, 2688
  - std::set, 2881
  - std::vector, 3047, 3048
- Bernoulli, 252
- bernoulli\_distribution
  - std::bernoulli\_distribution, 2112
- beta
  - std::gamma\_distribution, 2411
  - tr1\_math\_spec\_func, 107
- binary
  - std::basic\_fstream, 1544
  - std::basic\_ifstream, 1598
  - std::basic\_ios, 1628
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1744
  - std::basic\_istream, 1799
  - std::basic\_ofstream, 1843
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1931
  - std::basic\_stringstream, 2106
  - std::ios\_base, 2490
- Binary Search, 189
- binary\_search
  - binary\_search\_algorithms, 190, 191
- binary\_search\_algorithms
  - binary\_search, 190, 191
  - equal\_range, 191, 192
  - lower\_bound, 192, 193
  - upper\_bound, 193, 194
- bind
  - binders, 120
  - std, 572
- bind1st
  - binders, 120
- bind2nd
  - binders, 120
- Binder Classes, 118
- binders
  - \_GLIBCXX\_DEPRECATED\_-ATTR, 120
  - bind, 120
  - bind1st, 120
  - bind2nd, 120
- binders.h, 3132
- bitmap\_allocator.h, 3133
  - \_BALLOC\_ALIGN\_BYTES, 3134
- bitset, 3134, 3136, 3137

- std::bitset, 2132, 2133
- boolalpha
  - std, 572
  - std::basic\_fstream, 1545
  - std::basic\_ifstream, 1598
  - std::basic\_ios, 1629
  - std::basic\_iostream, 1693
  - std::basic\_istream, 1744
  - std::basic\_istream, 1800
  - std::basic\_ofstream, 1844
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1931
  - std::basic\_stringstream, 2106
  - std::ios\_base, 2491
- Boolean Operations Classes, 214
- boost\_concept\_check.h, 3138
- boost\_sp\_counted\_base.h, 3139
- c
  - std::queue, 2848
- c++0x\_warning.h, 3139
- c++allocator.h, 3139
- c++config.h, 3140
- c++io.h, 3145
- c++locale.h, 3145
- c++locale\_internal.h, 3146
- c\_str
  - \_\_gnu\_cxx::\_\_versa\_string, 748
  - \_\_gnu\_debug::basic\_string, 999
  - std::basic\_string, 1978
- cache\_line\_size
  - \_\_gnu\_parallel::\_Settings, 1150
- call\_once
  - mutexes, 67
  - std::once\_flag, 2799
- capacity
  - \_\_gnu\_cxx::\_\_versa\_string, 749
  - \_\_gnu\_debug::basic\_string, 999
  - std::basic\_string, 1978
  - std::vector, 3048
- cassert, 3146
- category
  - std::locale, 2556
- cbefore\_begin
  - std::forward\_list, 2373
- cbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 749
  - \_\_gnu\_debug::basic\_string, 999
  - std::basic\_string, 1978
  - std::deque, 2317
  - std::forward\_list, 2373
  - std::list, 2540
  - std::map, 2588
  - std::multimap, 2669
  - std::multiset, 2688
  - std::set, 2881
  - std::vector, 3048
- ccomplex, 3146, 3147
- cctype, 3147, 3148
- cend
  - \_\_gnu\_cxx::\_\_versa\_string, 749
  - \_\_gnu\_debug::basic\_string, 999
  - std::basic\_string, 1979
  - std::deque, 2317
  - std::forward\_list, 2373
  - std::list, 2540
  - std::map, 2588
  - std::multimap, 2669
  - std::multiset, 2689
  - std::set, 2882
  - std::vector, 3048
- cerr
  - std, 639
- cerrno, 3148
- cfenv, 3148, 3149
- cfloat, 3149
- char\_traits.h, 3150
- char\_type
  - \_\_gnu\_cxx::enc\_filebuf, 821
  - \_\_gnu\_cxx::stdio\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 921
  - std::\_\_ctype\_abstract\_base, 1254
  - std::basic\_filebuf, 1455
  - std::basic\_fstream, 1492
  - std::basic\_ifstream, 1557
  - std::basic\_ios, 1607
  - std::basic\_iostream, 1642
  - std::basic\_istream, 1705
  - std::basic\_istream, 1759
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1854
  - std::basic\_ostringstream, 1899

- std::basic\_streambuf, 1940
- std::basic\_stringbuf, 2022
- std::basic\_stringstream, 2054
- std::collate, 2190
- std::collate\_byname, 2197
- std::ctype, 2215
- std::ctype< char >, 2231
- std::ctype< wchar\_t >, 2245
- std::ctype\_byname, 2266
- std::ctype\_byname< char >, 2281
- std::istreambuf\_iterator, 2511
- std::messages, 2615
- std::messages\_byname, 2620
- std::money\_get, 2628
- std::money\_put, 2634
- std::moneypunct, 2640
- std::moneypunct\_byname, 2653
- std::num\_get, 2720
- std::num\_put, 2740
- std::numpunct, 2786
- std::numpunct\_byname, 2794
- std::ostream\_iterator, 2800
- std::ostreambuf\_iterator, 2805
- std::time\_get, 2932
- std::time\_get\_byname, 2943
- std::time\_put, 2954
- std::time\_put\_byname, 2959
- checkers.h, 3151
- chrono, 3151
- cin
  - std, 639
- cinttypes, 3154, 3155
- ciso646, 3155
- classic
  - std::locale, 2559
- classic\_table
  - std::ctype< char >, 2232
  - std::ctype\_byname< char >, 2282
- clear
  - \_\_gnu\_cxx::\_\_versa\_string, 749
  - \_\_gnu\_debug::basic\_string, 999
  - std::basic\_fstream, 1500
  - std::basic\_ifstream, 1562
  - std::basic\_ios, 1613
  - std::basic\_iostream, 1649
  - std::basic\_istream, 1710
  - std::basic\_istream, 1764
  - std::basic\_ofstream, 1817
  - std::basic\_ostream, 1860
  - std::basic\_ostringstream, 1905
  - std::basic\_string, 1979
  - std::basic\_stringstream, 2061
  - std::deque, 2317
  - std::forward\_list, 2373
  - std::list, 2540
  - std::map, 2588
  - std::multimap, 2669
  - std::multiset, 2689
  - std::set, 2882
  - std::vector, 3049
- climits, 3155, 3156
- locale, 3156
- clog
  - std, 639
- close
  - \_\_gnu\_cxx::enc\_filebuf, 823
  - \_\_gnu\_cxx::stdio\_filebuf, 892
  - std::basic\_filebuf, 1458
  - std::basic\_fstream, 1500
  - std::basic\_ifstream, 1563
  - std::basic\_ofstream, 1818
- cmath, 3157, 3160, 3163
- cmath.tcc, 3164
- codecvt.h, 3164
- codecvt\_specializations.h, 3165
- collate
  - std::collate, 2190
  - std::locale, 2562
- combine
  - std::locale, 2559
- comp\_ellint\_1
  - tr1\_math\_spec\_func, 107
- comp\_ellint\_2
  - tr1\_math\_spec\_func, 108
- comp\_ellint\_3
  - tr1\_math\_spec\_func, 108
- compare
  - \_\_gnu\_cxx::\_\_versa\_string, 750–753
  - \_\_gnu\_debug::basic\_string, 1000–1002
  - std::basic\_string, 1979–1982

- std::collate, 2191
- std::collate\_byname, 2197
- Comparison Classes, 213
- compatibility.h, 3166
- compiletime\_settings.h, 3167
  - \_GLIBCXX\_ASSERTIONS, 3167
  - \_GLIBCXX\_CALL, 3167
  - \_GLIBCXX\_RANDOM\_-SHUFFLE\_CONSIDER\_L1, 3168
  - \_GLIBCXX\_RANDOM\_-SHUFFLE\_CONSIDER\_TLB, 3168
  - \_GLIBCXX\_SCALE\_DOWN\_-FPU, 3168
  - \_GLIBCXX\_VERBOSE\_LEVEL, 3168
- complex, 3169, 3172, 3173
  - std::complex, 2202
- Complex Numbers, 34
- complex.h, 3175
- complex\_numbers
  - abs, 39
  - acos, 40
  - acosh, 40
  - arg, 40
  - asin, 40
  - asinh, 40
  - atan, 41
  - atanh, 41
  - conj, 41
  - cos, 41
  - cosh, 41
  - exp, 41
  - fabs, 42
  - log, 42
  - log10, 42
  - norm, 42
  - operator<<, 46
  - operator>>, 47
  - operator\*, 43
  - operator\*=:, 43, 44
  - operator+, 44
  - operator+=, 44
  - operator-, 45
  - operator-=, 45
  - operator/, 45, 46
  - operator/=, 46
  - operator=, 47
  - operator==, 47
  - polar, 48
  - pow, 48
  - sin, 48
  - sinh, 49
  - sqrt, 49
  - tan, 49
  - tanh, 49
- compose1
  - SGIextensions, 15
- compose2
  - SGIextensions, 15
- concept\_check.h, 3175
- concurrency.h, 3175
- Concurrency, 28
- cond\_dealtor.hpp, 3176
- Condition Variables, 50
- condition\_variable, 3177
- condition\_variables
  - cv\_status, 50
- conf\_hypereg
  - tr1\_math\_spec\_func, 108
- conj
  - complex\_numbers, 41
- const\_iterator
  - std::set, 2876
- const\_pointer
  - std::set, 2876
- const\_reference
  - std::set, 2876
- const\_reverse\_iterator
  - std::set, 2876
- constant0
  - SGIextensions, 15
- constant1
  - SGIextensions, 15
- constant2
  - SGIextensions, 15
- constructors\_destructor\_fn\_imps.hpp, 3177
- container\_base\_dispatch.hpp, 3178
- container\_type
  - std::back\_insert\_iterator, 1443

- std::front\_insert\_iterator, 2390
- std::insert\_iterator, 2473
- Containers, 22
- copy
  - \_\_gnu\_cxx::\_\_versa\_string, 754
  - \_\_gnu\_debug::basic\_string, 1003
  - mutating\_algorithms, 125
  - std::basic\_string, 1983
- copy\_backward
  - mutating\_algorithms, 125
- copy\_exception
  - exceptions, 32
- copy\_if
  - mutating\_algorithms, 126
- copy\_n
  - mutating\_algorithms, 126
  - SGIextensions, 16
- copyfmt
  - std::basic\_fstream, 1500
  - std::basic\_ifstream, 1563
  - std::basic\_ios, 1613
  - std::basic\_istream, 1650
  - std::basic\_istream, 1711
  - std::basic\_istream, 1765
  - std::basic\_ofstream, 1818
  - std::basic\_ostream, 1861
  - std::basic\_ostringstream, 1906
  - std::basic\_stringstream, 2062
- cos
  - complex\_numbers, 41
- cosh
  - complex\_numbers, 41
- count
  - non\_mutating\_algorithms, 149
  - std::bitset, 2134
  - std::map, 2589
  - std::multimap, 2670
  - std::multiset, 2689
  - std::set, 2882
- count\_if
  - non\_mutating\_algorithms, 149
- count\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1150
- cout
  - std, 639
- cpp\_type\_traits.h, 3178
- cpu\_defines.h, 3179
- crbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 754
  - \_\_gnu\_debug::basic\_string, 1003
  - std::basic\_string, 1983
  - std::deque, 2317
  - std::list, 2540
  - std::map, 2589
  - std::multimap, 2670
  - std::multiset, 2689
  - std::set, 2882
  - std::vector, 3049
- cref
  - std, 572
- crend
  - \_\_gnu\_cxx::\_\_versa\_string, 755
  - \_\_gnu\_debug::basic\_string, 1003
  - std::basic\_string, 1984
  - std::deque, 2318
  - std::list, 2541
  - std::map, 2589
  - std::multimap, 2670
  - std::multiset, 2690
  - std::set, 2883
  - std::vector, 3049
- csetjmp, 3179
- cshift
  - numeric\_arrays, 87
- csignal, 3180
- cstdarg, 3180, 3181
- cstdbool, 3181
- cstddef, 3181
- cstdint, 3182
- cstdio, 3183
- cstdlib, 3184, 3185
- cstring, 3185
- ctgmath, 3186
- ctime, 3186
- ctype
  - std::ctype< char >, 2231
  - std::ctype< wchar\_t >, 2246
  - std::locale, 2562
- ctype\_base.h, 3187
- ctype\_inline.h, 3187
- ctype\_noninline.h, 3187
- cur

- std::basic\_fstream, 1545
- std::basic\_ifstream, 1598
- std::basic\_ios, 1629
- std::basic\_iostream, 1694
- std::basic\_istream, 1745
- std::basic\_istream, 1800
- std::basic\_ofstream, 1844
- std::basic\_ostream, 1886
- std::basic\_ostringstream, 1931
- std::basic\_stringstream, 2106
- std::ios\_base, 2491
- curr\_symbol
  - std::moneypunct, 2642
  - std::moneypunct\_byname, 2654
- current\_exception
  - exceptions, 32
- cv\_status
  - condition\_variables, 50
- cwchar, 3188, 3189
- cwctype, 3189, 3190
- cxxabi-forced.h, 3190
- cxxabi.h, 3190
- cxxabi\_tweaks.h, 3192
- cyl\_bessel\_i
  - tr1\_math\_spec\_func, 108
- cyl\_bessel\_j
  - tr1\_math\_spec\_func, 108
- cyl\_bessel\_k
  - tr1\_math\_spec\_func, 108
- cyl\_neumann
  - tr1\_math\_spec\_func, 109
- data
  - \_\_gnu\_cxx::\_\_versa\_string, 755
  - \_\_gnu\_debug::basic\_string, 1004
  - std::basic\_string, 1984
  - std::vector, 3049
- date\_order
  - std::time\_get, 2933
  - std::time\_get\_byname, 2944
- debug.h, 3193
- debug\_allocator.h, 3194
- debug\_map\_base.hpp, 3194
- dec
  - std, 572
  - std::basic\_fstream, 1545
  - std::basic\_ifstream, 1598
  - std::basic\_ios, 1629
  - std::basic\_iostream, 1694
  - std::basic\_istream, 1745
  - std::basic\_istream, 1800
  - std::basic\_ofstream, 1844
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1931
  - std::basic\_stringstream, 2106
  - std::ios\_base, 2491
- decimal, 3194
- Decimal Floating-Point Arithmetic, 117
- decimal128
  - std::decimal::decimal128, 2294
- decimal32
  - std::decimal::decimal32, 2296
- decimal32\_to\_long\_long
  - std::decimal, 694
- decimal64
  - std::decimal::decimal64, 2298
- decimal\_point
  - std::moneypunct, 2642
  - std::moneypunct\_byname, 2654
  - std::numpunct, 2787
  - std::numpunct\_byname, 2794
- denorm\_absent
  - std, 555
- denorm\_indeterminate
  - std, 555
- denorm\_present
  - std, 555
- denorm\_min
  - std::numeric\_limits, 2754
- densities
  - std::piecewise\_constant\_distribution, 2817
  - std::piecewise\_linear\_distribution, 2823
- deque, 3206–3208
  - std::deque, 2306–2308
- deque.tcc, 3209
- Diagnostics, 27
- difference\_type
  - std::back\_insert\_iterator, 1443
  - std::front\_insert\_iterator, 2390
  - std::insert\_iterator, 2473

- std::istream\_iterator, 2508
- std::istreambuf\_iterator, 2511
- std::iterator, 2515
- std::ostream\_iterator, 2800
- std::ostreambuf\_iterator, 2805
- std::raw\_storage\_iterator, 2859
- std::reverse\_iterator, 2865
- std::set, 2877
- digits
  - std::\_\_numeric\_limits\_base, 1339
  - std::numeric\_limits, 2755
- digits10
  - std::\_\_numeric\_limits\_base, 1339
  - std::numeric\_limits, 2756
- discard
  - std::discard\_block\_engine, 2332
  - std::independent\_bits\_engine, 2464
  - std::linear\_congruential\_engine, 2527
  - std::shuffle\_order\_engine, 2910
- discard\_block\_engine
  - std::discard\_block\_engine, 2330, 2331
- distance
  - SGIextensions, 16
  - std, 573
- do\_compare
  - std::collate, 2191
  - std::collate\_byname, 2197
- do\_curr\_symbol
  - std::moneypunct, 2642
  - std::moneypunct\_byname, 2654
- do\_date\_order
  - std::time\_get, 2933
  - std::time\_get\_byname, 2944
- do\_decimal\_point
  - std::moneypunct, 2643
  - std::moneypunct\_byname, 2655
  - std::numpunct, 2788
  - std::numpunct\_byname, 2794
- do\_falsename
  - std::numpunct, 2788
  - std::numpunct\_byname, 2795
- do\_frac\_digits
  - std::moneypunct, 2643
  - std::moneypunct\_byname, 2655
- do\_get
  - std::money\_get, 2629
  - std::num\_get, 2721–2727
- do\_get\_date
  - std::time\_get, 2934
  - std::time\_get\_byname, 2944
- do\_get\_monthname
  - std::time\_get, 2934
  - std::time\_get\_byname, 2945
- do\_get\_time
  - std::time\_get, 2935
  - std::time\_get\_byname, 2946
- do\_get\_weekday
  - std::time\_get, 2936
  - std::time\_get\_byname, 2946
- do\_get\_year
  - std::time\_get, 2936
  - std::time\_get\_byname, 2947
- do\_grouping
  - std::moneypunct, 2644
  - std::moneypunct\_byname, 2655
  - std::numpunct, 2788
  - std::numpunct\_byname, 2795
- do\_hash
  - std::collate, 2192
  - std::collate\_byname, 2198
- do\_is
  - std::\_\_ctype\_abstract\_base, 1255
  - std::ctype, 2216
  - std::ctype< wchar\_t >, 2246, 2247
  - std::ctype\_byname, 2267
- do\_narrow
  - std::\_\_ctype\_abstract\_base, 1255, 1256
  - std::ctype, 2217
  - std::ctype< char >, 2232
  - std::ctype< wchar\_t >, 2248, 2249
  - std::ctype\_byname, 2267, 2268
  - std::ctype\_byname< char >, 2282
- do\_neg\_format
  - std::moneypunct, 2644
  - std::moneypunct\_byname, 2656
- do\_negative\_sign
  - std::moneypunct, 2644
  - std::moneypunct\_byname, 2656
- do\_out

- std::\_\_codecvt\_abstract\_base, 1248
- std::codecvt, 2161
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2167
- std::codecvt< char, char, mbstate\_t >, 2172
- std::codecvt< wchar\_t, char, mbstate\_t >, 2177
- std::codecvt\_byname, 2184
- do\_pos\_format
  - std::moneypunct, 2645
  - std::moneypunct\_byname, 2657
- do\_positive\_sign
  - std::moneypunct, 2645
  - std::moneypunct\_byname, 2657
- do\_put
  - std::money\_put, 2635
  - std::num\_put, 2741–2744
  - std::time\_put, 2955
  - std::time\_put\_byname, 2959
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base, 1257
  - std::ctype, 2218
  - std::ctype< wchar\_t >, 2250
  - std::ctype\_byname, 2269
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base, 1257
  - std::ctype, 2218
  - std::ctype< wchar\_t >, 2251
  - std::ctype\_byname, 2269
- do\_thousands\_sep
  - std::moneypunct, 2646
  - std::moneypunct\_byname, 2658
  - std::numpunct, 2789
  - std::numpunct\_byname, 2795
- do\_tolower
  - std::\_\_ctype\_abstract\_base, 1258
  - std::ctype, 2219
  - std::ctype< char >, 2233
  - std::ctype< wchar\_t >, 2252, 2253
  - std::ctype\_byname, 2270
  - std::ctype\_byname< char >, 2283
- do\_toupper
  - std::\_\_ctype\_abstract\_base, 1259
  - std::ctype, 2220
  - std::ctype< char >, 2234
- std::ctype< wchar\_t >, 2254, 2255
- std::ctype\_byname, 2271
- std::ctype\_byname< char >, 2284
- do\_transform
  - std::collate, 2192
  - std::collate\_byname, 2198
- do\_truename
  - std::numpunct, 2789
  - std::numpunct\_byname, 2796
- do\_widen
  - std::\_\_ctype\_abstract\_base, 1260
  - std::ctype, 2220, 2221
  - std::ctype< char >, 2234, 2235
  - std::ctype< wchar\_t >, 2255, 2256
  - std::ctype\_byname, 2271, 2272
  - std::ctype\_byname< char >, 2284, 2285
- duration\_cast
  - std::chrono, 683
- eback
  - \_\_gnu\_cxx::enc\_filebuf, 823
  - \_\_gnu\_cxx::stdio\_filebuf, 893
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 923
  - std::basic\_filebuf, 1458
  - std::basic\_streambuf, 1942
  - std::basic\_stringbuf, 2025
- egptr
  - \_\_gnu\_cxx::enc\_filebuf, 823
  - \_\_gnu\_cxx::stdio\_filebuf, 893
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 923
  - std::basic\_filebuf, 1459
  - std::basic\_streambuf, 1942
  - std::basic\_stringbuf, 2025
- element\_type
  - std::auto\_ptr, 1437
- ellint\_1
  - tr1\_math\_spec\_func, 109
- ellint\_2
  - tr1\_math\_spec\_func, 109
- ellint\_3
  - tr1\_math\_spec\_func, 109
- emplace
  - std::deque, 2318
  - std::list, 2541
  - std::vector, 3049

- emplace\_after
  - std::forward\_list, 2374
- emplace\_front
  - std::forward\_list, 2374
- empty
  - \_\_gnu\_cxx::\_\_versa\_string, 755
  - \_\_gnu\_debug::basic\_string, 1004
  - std::basic\_string, 1984
  - std::deque, 2318
  - std::forward\_list, 2374
  - std::list, 2541
  - std::map, 2590
  - std::multimap, 2670
  - std::multiset, 2690
  - std::priority\_queue, 2840
  - std::queue, 2846
  - std::set, 2883
  - std::stack, 2919
  - std::vector, 3050
- enc\_filebuf.h, 3209
- end
  - \_\_gnu\_cxx::\_\_versa\_string, 755, 756
  - \_\_gnu\_cxx::temporary\_buffer, 946
  - \_\_gnu\_debug::basic\_string, 1004
  - \_\_gnu\_parallel::\_PseudoSequence, 1138
  - std::\_Temporary\_buffer, 1408
  - std::basic\_fstream, 1545
  - std::basic\_ifstream, 1598
  - std::basic\_ios, 1629
  - std::basic\_iostream, 1694
  - std::basic\_istream, 1745
  - std::basic\_istream, 1800
  - std::basic\_ofstream, 1844
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1932
  - std::basic\_string, 1984, 1985
  - std::basic\_stringstream, 2106
  - std::deque, 2319
  - std::forward\_list, 2375
  - std::ios\_base, 2491
  - std::list, 2541, 2542
  - std::map, 2590
  - std::multimap, 2671
  - std::multiset, 2690
  - std::set, 2883
  - std::vector, 3050
- endl
  - std, 573
- ends
  - std, 573
- eof
  - std::basic\_fstream, 1501
  - std::basic\_ifstream, 1564
  - std::basic\_ios, 1614
  - std::basic\_iostream, 1650
  - std::basic\_istream, 1711
  - std::basic\_istream, 1765
  - std::basic\_ofstream, 1819
  - std::basic\_ostream, 1861
  - std::basic\_ostringstream, 1906
  - std::basic\_stringstream, 2062
- eofbit
  - std::basic\_fstream, 1545
  - std::basic\_ifstream, 1599
  - std::basic\_ios, 1629
  - std::basic\_iostream, 1694
  - std::basic\_istream, 1745
  - std::basic\_istream, 1800
  - std::basic\_ofstream, 1844
  - std::basic\_ostream, 1886
  - std::basic\_ostringstream, 1932
  - std::basic\_stringstream, 2106
  - std::ios\_base, 2491
- epptr
  - \_\_gnu\_cxx::enc\_filebuf, 824
  - \_\_gnu\_cxx::stdio\_filebuf, 894
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 924
  - std::basic\_filebuf, 1459
  - std::basic\_streambuf, 1943
  - std::basic\_stringbuf, 2025
- epsilon
  - std::numeric\_limits, 2754
- equal
  - non\_mutating\_algorithms, 149, 150
  - std::istreambuf\_iterator, 2513
- equal\_range
  - binary\_search\_algorithms, 191, 192
  - std::map, 2590, 2591
  - std::multimap, 2671, 2672
  - std::multiset, 2690, 2691

- std::set, 2883, 2884
- equally\_split
  - \_\_gnu\_parallel, 398
- equally\_split.h, 3210
- equally\_split\_point
  - \_\_gnu\_parallel, 399
- erase
  - \_\_gnu\_cxx::\_\_versa\_string, 756, 757
  - \_\_gnu\_debug::basic\_string, 1004, 1005
  - std::basic\_string, 1985, 1986
  - std::deque, 2319
  - std::list, 2542
  - std::map, 2592
  - std::multimap, 2672, 2673
  - std::multiset, 2691, 2692
  - std::set, 2884, 2885
  - std::vector, 3051
- erase\_after
  - std::forward\_list, 2375, 2376
- error\_constants.h, 3210
- event
  - std::basic\_fstream, 1497
  - std::basic\_ifstream, 1560
  - std::basic\_ios, 1612
  - std::basic\_iostream, 1647
  - std::basic\_istream, 1709
  - std::basic\_istringstream, 1762
  - std::basic\_ofstream, 1815
  - std::basic\_ostream, 1859
  - std::basic\_ostringstream, 1903
  - std::basic\_stringstream, 2059
  - std::ios\_base, 2483
- event\_callback
  - std::basic\_fstream, 1492
  - std::basic\_ifstream, 1557
  - std::basic\_ios, 1608
  - std::basic\_iostream, 1643
  - std::basic\_istream, 1705
  - std::basic\_istringstream, 1759
  - std::basic\_ofstream, 1811
  - std::basic\_ostream, 1855
  - std::basic\_ostringstream, 1899
  - std::basic\_stringstream, 2054
  - std::ios\_base, 2480
- exception, 3212
- exception.hpp, 3212
- exception\_ptr.h, 3213
- Exceptions, 29
- exceptions
  - \_\_verbose\_terminate\_handler, 32
  - copy\_exception, 32
  - current\_exception, 32
  - rethrow\_exception, 32
  - rethrow\_if\_nested, 32
  - set\_terminate, 33
  - set\_unexpected, 33
  - std::basic\_fstream, 1501
  - std::basic\_ifstream, 1564
  - std::basic\_ios, 1614, 1615
  - std::basic\_iostream, 1650, 1651
  - std::basic\_istream, 1711, 1712
  - std::basic\_istringstream, 1766
  - std::basic\_ofstream, 1819
  - std::basic\_ostream, 1862
  - std::basic\_ostringstream, 1907
  - std::basic\_stringstream, 2063
  - terminate, 33
  - terminate\_handler, 31
  - throw\_with\_nested, 33
  - uncaught\_exception, 33
  - unexpected, 33
  - unexpected\_handler, 31
- exp
  - complex\_numbers, 41
- expint
  - tr1\_math\_spec\_func, 109
- exponential\_distribution
  - std::exponential\_distribution, 2352
- Extensions, 7
- extptr\_allocator.h, 3214
- fabs
  - complex\_numbers, 42
- facet
  - std::locale::facet, 2566
- fail
  - std::basic\_fstream, 1502
  - std::basic\_ifstream, 1565
  - std::basic\_ios, 1615
  - std::basic\_iostream, 1651

- std::basic\_istream, 1712
- std::basic\_istream, 1767
- std::basic\_ofstream, 1820
- std::basic\_ostream, 1863
- std::basic\_ostringstream, 1908
- std::basic\_stringstream, 2064
- failbit
  - std::basic\_fstream, 1546
  - std::basic\_ifstream, 1599
  - std::basic\_ios, 1630
  - std::basic\_iostream, 1694
  - std::basic\_istream, 1745
  - std::basic\_istream, 1801
  - std::basic\_ofstream, 1845
  - std::basic\_ostream, 1887
  - std::basic\_ostringstream, 1932
  - std::basic\_stringstream, 2107
  - std::ios\_base, 2492
- failed
  - std::ostreambuf\_iterator, 2807
- false\_type
  - metaprogramming, 117
- falsename
  - std::numpunct, 2790
  - std::numpunct\_byname, 2796
- fd
  - \_\_gnu\_cxx::stdio\_filebuf, 894
- features.h, 3214
  - \_GLIBCXX\_BAL\_QUICKSORT, 3215
  - \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS, 3215
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT, 3215
  - \_GLIBCXX\_FIND\_GROWING\_BLOCKS, 3215
  - \_GLIBCXX\_MERGESORT, 3216
  - \_GLIBCXX\_QUICKSORT, 3216
  - \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING, 3216
  - \_GLIBCXX\_TREE\_FULL\_COPY, 3216
  - \_GLIBCXX\_TREE\_INITIAL\_SPLITTING, 3217
- fenv.h, 3217
- file
  - \_\_gnu\_cxx::stdio\_filebuf, 894
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 924
- filebuf
  - io, 57
- fill
  - mutating\_algorithms, 127
  - std::basic\_fstream, 1503
  - std::basic\_ifstream, 1565, 1566
  - std::basic\_ios, 1616
  - std::basic\_iostream, 1652
  - std::basic\_istream, 1713
  - std::basic\_istream, 1767
  - std::basic\_ofstream, 1820, 1821
  - std::basic\_ostream, 1863
  - std::basic\_ostringstream, 1908
  - std::basic\_stringstream, 2064
- fill\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1150
- fill\_n
  - mutating\_algorithms, 127
- find
  - \_\_gnu\_cxx::\_\_versa\_string, 757–759
  - \_\_gnu\_debug::basic\_string, 1006, 1007
  - non\_mutating\_algorithms, 150
  - std::basic\_string, 1986–1988
  - std::map, 2593
  - std::multimap, 2674
  - std::multiset, 2693
  - std::set, 2886
- find.h, 3217
- find\_end
  - non\_mutating\_algorithms, 151, 152
- find\_first\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 759–761
  - \_\_gnu\_debug::basic\_string, 1007, 1008
  - std::basic\_string, 1988–1990
- find\_first\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 762, 763
  - \_\_gnu\_debug::basic\_string, 1009, 1010
  - non\_mutating\_algorithms, 152, 153

- std::basic\_string, 1990–1992
- find\_if
  - non\_mutating\_algorithms, 154
- find\_if\_not
  - non\_mutating\_algorithms, 154
- find\_increasing\_factor
  - \_\_gnu\_parallel::\_Settings, 1150
- find\_initial\_block\_size
  - \_\_gnu\_parallel::\_Settings, 1150
- find\_last\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 764, 765
  - \_\_gnu\_debug::basic\_string, 1011, 1012
  - std::basic\_string, 1992–1994
- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string, 766, 767
  - \_\_gnu\_debug::basic\_string, 1012–1014
  - std::basic\_string, 1994, 1995
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::\_Settings, 1150
- find\_selectors.h, 3218
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::\_Settings, 1151
- first
  - \_\_gnu\_parallel::\_IteratorPair, 1093
  - std::pair, 2815
- first\_argument\_type
  - \_\_gnu\_cxx::project1st, 864
  - \_\_gnu\_cxx::project2nd, 865
  - \_\_gnu\_parallel::\_EqualFromLess, 1086
  - \_\_gnu\_parallel::\_EqualTo, 1088
  - \_\_gnu\_parallel::\_Less, 1098
  - \_\_gnu\_parallel::\_Lexicographic, 1100
  - \_\_gnu\_parallel::\_-
    - LexicographicReverse, 1102
  - \_\_gnu\_parallel::\_Multiplies, 1130
  - \_\_gnu\_parallel::\_Plus, 1134
  - std::\_Maybe\_unary\_or\_binary\_-
    - function< \_Res, \_T1, \_T2 >, 1394
  - std::binary\_function, 2116
  - std::binary\_negate, 2117
  - std::const\_mem\_fun1\_ref\_t, 2206
  - std::const\_mem\_fun1\_t, 2208
  - std::divides, 2341
  - std::equal\_to, 2346
  - std::greater, 2420
  - std::greater\_equal, 2422
  - std::less, 2522
  - std::less\_equal, 2524
  - std::logical\_and, 2572
  - std::logical\_or, 2576
  - std::mem\_fun1\_ref\_t, 2606
  - std::mem\_fun1\_t, 2608
  - std::minus, 2622
  - std::modulus, 2624
  - std::multiplies, 2683
  - std::not\_equal\_to, 2717
  - std::plus, 2828
  - std::pointer\_to\_binary\_function, 2830
  - first\_type
    - \_\_gnu\_parallel::\_IteratorPair, 1093
    - std::pair, 2814
  - fixed
    - std, 574
    - std::basic\_fstream, 1546
    - std::basic\_ifstream, 1599
    - std::basic\_ios, 1630
    - std::basic\_iostream, 1695
    - std::basic\_istream, 1746
    - std::basic\_istreamstream, 1801
    - std::basic\_ofstream, 1845
    - std::basic\_ostream, 1887
    - std::basic\_ostreamstream, 1932
    - std::basic\_stringstream, 2107
    - std::ios\_base, 2492
  - flags
    - std::basic\_fstream, 1503, 1504
    - std::basic\_ifstream, 1566
    - std::basic\_ios, 1616, 1617
    - std::basic\_iostream, 1653
    - std::basic\_istream, 1714
    - std::basic\_istreamstream, 1768
    - std::basic\_ofstream, 1821
    - std::basic\_ostream, 1864
    - std::basic\_ostreamstream, 1909

- std::basic\_stringstream, 2065
- std::ios\_base, 2483, 2484
- flip
  - std::bitset, 2134
- float\_denorm\_style
  - std, 554
- float\_round\_style
  - std, 555
- floatfield
  - std::basic\_fstream, 1546
  - std::basic\_ifstream, 1599
  - std::basic\_ios, 1630
  - std::basic\_iostream, 1695
  - std::basic\_istream, 1746
  - std::basic\_istream, 1801
  - std::basic\_ofstream, 1845
  - std::basic\_ostream, 1887
  - std::basic\_ostringstream, 1932
  - std::basic\_stringstream, 2107
  - std::ios\_base, 2492
- flush
  - std, 574
  - std::basic\_fstream, 1504
  - std::basic\_iostream, 1653
  - std::basic\_ofstream, 1822
  - std::basic\_ostream, 1865
  - std::basic\_ostringstream, 1909
  - std::basic\_stringstream, 2066
- fmtflags
  - std::basic\_fstream, 1493
  - std::basic\_ifstream, 1557
  - std::basic\_ios, 1608
  - std::basic\_iostream, 1643
  - std::basic\_istream, 1706
  - std::basic\_istream, 1759
  - std::basic\_ofstream, 1812
  - std::basic\_ostream, 1855
  - std::basic\_ostringstream, 1900
  - std::basic\_stringstream, 2055
  - std::ios\_base, 2480
- for\_each
  - non\_mutating\_algorithms, 154
- for\_each.h, 3219
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1151
- for\_each\_selectors.h, 3220
- formatter.h, 3222
- forward
  - std, 574
- forward\_list
  - std::forward\_list, 2368–2370
- forward\_list.h, 3223
- forward\_list.tcc, 3224
- fpos
  - std::fpos, 2387
- frac\_digits
  - std::moneypunct, 2646
  - std::moneypunct\_byname, 2658
- front
  - \_\_gnu\_cxx::\_versa\_string, 768
  - std::deque, 2320
  - std::forward\_list, 2376
  - std::list, 2543
  - std::queue, 2846, 2847
  - std::vector, 3052
- front\_insert\_iterator
  - std::front\_insert\_iterator, 2391
- front\_inserter
  - iterators, 230
- fstream, 3225
  - io, 57
- fstream.tcc, 3226
- funcexcept.h, 3226
- function
  - std::function<\_Res(\_-ArgTypes...)>, 2395, 2396
- Function Objects, 210
- functional, 3227, 3231
- functional\_hash.h, 3232
- functions.h, 3233
- functors
  - mem\_fn, 212
- future, 3235
  - std::future, 2402
  - std::future<\_Res & >, 2405
  - std::future< void >, 2407
- future\_category
  - futures, 53
- future\_errc
  - futures, 53
- Futures, 51
- futures

- future\_category, 53
- future\_errc, 53
- gamma\_distribution
  - std::gamma\_distribution, 2411
- gbump
  - \_\_gnu\_cxx::enc\_filebuf, 824
  - \_\_gnu\_cxx::stdio\_filebuf, 895
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 924
  - std::basic\_filebuf, 1460
  - std::basic\_streambuf, 1943
  - std::basic\_stringbuf, 2026
- gcount
  - std::basic\_fstream, 1504
  - std::basic\_ifstream, 1567
  - std::basic\_iostream, 1654
  - std::basic\_istream, 1714
  - std::basic\_istream, 1769
  - std::basic\_stringstream, 2066
- generate
  - mutating\_algorithms, 128
- generate\_canonical
  - random, 209
- generate\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1151
- generate\_n
  - mutating\_algorithms, 128
- get
  - \_\_gnu\_parallel::\_Settings, 1149
  - std::auto\_ptr, 1439
  - std::basic\_fstream, 1505–1508
  - std::basic\_ifstream, 1567–1570
  - std::basic\_iostream, 1654–1657
  - std::basic\_istream, 1715–1717
  - std::basic\_istream, 1769–1772
  - std::basic\_stringstream, 2066–2069
  - std::future, 2402
  - std::future< \_Res & >, 2405
  - std::future< void >, 2407
  - std::money\_get, 2630, 2631
  - std::num\_get, 2728–2736
  - std::shared\_future, 2895
  - std::shared\_future< \_Res & >, 2898
- get\_allocator
  - \_\_gnu\_cxx::\_versa\_string, 768
  - \_\_gnu\_debug::basic\_string, 1014
  - std::basic\_string, 1996
  - std::deque, 2320
  - std::forward\_list, 2376
  - std::list, 2543
  - std::map, 2594
  - std::multimap, 2674
  - std::multiset, 2694
  - std::set, 2886
- get\_date
  - std::time\_get, 2937
  - std::time\_get\_byname, 2948
- get\_deleter
  - pointer\_abstractions, 64
- get\_id
  - std::this\_thread, 697
- get\_money
  - std, 575
- get\_monthname
  - std::time\_get, 2938
  - std::time\_get\_byname, 2948
- get\_temporary\_buffer
  - std, 575
- get\_time
  - std::time\_get, 2938
  - std::time\_get\_byname, 2949
- get\_weekday
  - std::time\_get, 2939
  - std::time\_get\_byname, 2950
- get\_year
  - std::time\_get, 2940
  - std::time\_get\_byname, 2950
- getline
  - std, 575–577
  - std::basic\_fstream, 1508
  - std::basic\_ifstream, 1570, 1571
  - std::basic\_iostream, 1657, 1658
  - std::basic\_istream, 1718, 1719
  - std::basic\_istream, 1772, 1773
  - std::basic\_stringstream, 2070, 2071
- getloc
  - \_\_gnu\_cxx::enc\_filebuf, 824
  - \_\_gnu\_cxx::stdio\_filebuf, 895
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 925
  - std::basic\_filebuf, 1460
  - std::basic\_fstream, 1509
  - std::basic\_ifstream, 1572

- std::basic\_ios, 1617
- std::basic\_iostream, 1659
- std::basic\_istream, 1719
- std::basic\_istream, 1774
- std::basic\_ofstream, 1822
- std::basic\_ostream, 1865
- std::basic\_ostringstream, 1910
- std::basic\_streambuf, 1944
- std::basic\_stringbuf, 2026
- std::basic\_stringstream, 2071
- std::ios\_base, 2484
- global
  - std::locale, 2559
- good
  - std::basic\_fstream, 1510
  - std::basic\_ifstream, 1572
  - std::basic\_ios, 1617
  - std::basic\_iostream, 1659
  - std::basic\_istream, 1719
  - std::basic\_istream, 1774
  - std::basic\_ofstream, 1822
  - std::basic\_ostream, 1865
  - std::basic\_ostringstream, 1910
  - std::basic\_stringstream, 2071
- goodbit
  - std::basic\_fstream, 1546
  - std::basic\_ifstream, 1599
  - std::basic\_ios, 1630
  - std::basic\_iostream, 1695
  - std::basic\_istream, 1746
  - std::basic\_istream, 1801
  - std::basic\_ofstream, 1845
  - std::basic\_ostream, 1887
  - std::basic\_ostringstream, 1933
  - std::basic\_stringstream, 2107
  - std::ios\_base, 2492
- gptr
  - \_\_gnu\_cxx::enc\_filebuf, 825
  - \_\_gnu\_cxx::stdio\_filebuf, 895
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 925
  - std::basic\_filebuf, 1460
  - std::basic\_streambuf, 1944
  - std::basic\_stringbuf, 2026
- grouping
  - std::moneypunct, 2646
  - std::moneypunct\_byname, 2658
- std::numpunct, 2790
- std::numpunct\_byname, 2796
- gslice
  - numeric\_arrays, 83
- gslice.h, 3238
- gslice\_array
  - numeric\_arrays, 83
- gslice\_array.h, 3238
- has\_denorm
  - std::\_\_numeric\_limits\_base, 1339
  - std::numeric\_limits, 2756
- has\_denorm\_loss
  - std::\_\_numeric\_limits\_base, 1339
  - std::numeric\_limits, 2756
- has\_facet
  - std, 578
  - std::locale, 2561
  - std::locale::id, 2567
- has\_infinity
  - std::\_\_numeric\_limits\_base, 1339
  - std::numeric\_limits, 2756
- has\_quiet\_NaN
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2756
- has\_signaling\_NaN
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2756
- hash
  - std::collate, 2193
  - std::collate\_byname, 2199
- hash\_fun.h, 3239
- hash\_map, 3239
- hash\_policy.hpp, 3240
- hash\_set, 3241
- Hashes, 205
- hashtable.h, 3242, 3243
- hashtable\_policy.h, 3244
- Heap, 219
- heap\_algorithms
  - is\_heap, 221
  - is\_heap\_until, 221, 222
  - make\_heap, 222, 223
  - pop\_heap, 223
  - push\_heap, 224
  - sort\_heap, 225

- 
- hermite
    - tr1\_math\_spec\_func, 110
  - hex
    - std, 578
    - std::basic\_fstream, 1547
    - std::basic\_ifstream, 1600
    - std::basic\_ios, 1631
    - std::basic\_iostream, 1695
    - std::basic\_istream, 1746
    - std::basic\_istreamstream, 1802
    - std::basic\_ofstream, 1846
    - std::basic\_ostream, 1888
    - std::basic\_ostringstream, 1933
    - std::basic\_stringstream, 2108
    - std::ios\_base, 2493
  - hours
    - std::chrono, 682
  - hyperg
    - tr1\_math\_spec\_func, 110
  - I/O, 53
  - id
    - std::collate, 2194
    - std::collate\_byname, 2200
    - std::ctype, 2227
    - std::ctype< char >, 2241
    - std::ctype< wchar\_t >, 2262
    - std::ctype\_byname, 2278
    - std::ctype\_byname< char >, 2291
    - std::locale::id, 2567
    - std::messages, 2616
    - std::messages\_byname, 2620
    - std::money\_get, 2631
    - std::money\_put, 2637
    - std::moneypunct, 2650
    - std::moneypunct\_byname, 2662
    - std::num\_get, 2737
    - std::num\_put, 2752
    - std::numpunct, 2791
    - std::numpunct\_byname, 2798
    - std::time\_get, 2941
    - std::time\_get\_byname, 2951
    - std::time\_put, 2957
    - std::time\_put\_byname, 2961
  - identity\_element
    - SGlextensions, 16, 17
  - ifstream
    - io, 57
  - ignore
    - std::basic\_fstream, 1510, 1511
    - std::basic\_ifstream, 1572–1574
    - std::basic\_iostream, 1659–1661
    - std::basic\_istream, 1720, 1721
    - std::basic\_istreamstream, 1774, 1775
    - std::basic\_stringstream, 2072, 2073
  - imbue
    - \_\_gnu\_cxx::enc\_filebuf, 825
    - \_\_gnu\_cxx::stdio\_filebuf, 896
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 925
    - std::basic\_filebuf, 1461
    - std::basic\_fstream, 1512
    - std::basic\_ifstream, 1574
    - std::basic\_ios, 1618
    - std::basic\_iostream, 1661
    - std::basic\_istream, 1721
    - std::basic\_istreamstream, 1776
    - std::basic\_ofstream, 1823
    - std::basic\_ostream, 1866
    - std::basic\_ostringstream, 1910
    - std::basic\_streambuf, 1944
    - std::basic\_stringbuf, 2027
    - std::basic\_stringstream, 2073
    - std::ios\_base, 2484
  - in
    - std::\_\_codecvt\_abstract\_base, 1248
    - std::basic\_fstream, 1547
    - std::basic\_ifstream, 1600
    - std::basic\_ios, 1631
    - std::basic\_iostream, 1696
    - std::basic\_istream, 1747
    - std::basic\_istreamstream, 1802
    - std::basic\_ofstream, 1846
    - std::basic\_ostream, 1888
    - std::basic\_ostringstream, 1933
    - std::basic\_stringstream, 2108
    - std::codecvt, 2161
    - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2167
    - std::codecvt< char, char, mbstate\_t >, 2172
    - std::codecvt< wchar\_t, char, mbstate\_t >, 2177
-

- std::codecvt\_byname, 2184
- std::ios\_base, 2493
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf, 825
  - \_\_gnu\_cxx::stdio\_filebuf, 896
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 926
  - std::basic\_filebuf, 1461
  - std::basic\_streambuf, 1945
  - std::basic\_stringbuf, 2027
- include/ Directory Reference, 285
- include/backward/ Directory Reference, 268
- include/bits/ Directory Reference, 271
- include/debug/ Directory Reference, 274
- include/decimal/ Directory Reference, 276
- include/ext/ Directory Reference, 280
- include/ext/pb\_ds/ Directory Reference, 293
- include/ext/pb\_ds/detail/ Directory Reference, 277
- include/parallel/ Directory Reference, 290
- include/profile/ Directory Reference, 295
- include/profile/impl/ Directory Reference, 283
- include/tr1/ Directory Reference, 297
- include/tr1\_impl/ Directory Reference, 298
- include/x86\_64-unknown-linux-gnu/ Directory Reference, 299
- include/x86\_64-unknown-linux-gnu/bits/ Directory Reference, 269
- includes
  - set\_algorithms, 183
- increment
  - std::linear\_congruential\_engine, 2530
- independent\_bits\_engine
  - std::independent\_bits\_engine, 2462, 2463
- indirect\_array
  - numeric\_arrays, 83
- indirect\_array.h, 3245
- infinity
  - std::numeric\_limits, 2754
- init
  - std::basic\_fstream, 1512
  - std::basic\_ifstream, 1575
  - std::basic\_ios, 1618
  - std::basic\_iostream, 1662
  - std::basic\_istream, 1722
  - std::basic\_istreamstream, 1776
  - std::basic\_ofstream, 1823
  - std::basic\_ostream, 1866
  - std::basic\_ostreamstream, 1911
  - std::basic\_stringstream, 2074
- initializer\_list, 3245
- inner\_product
  - std, 578, 579
- inplace\_merge
  - sorting\_algorithms, 163
- insert
  - \_\_gnu\_cxx::\_\_versa\_string, 769–773
  - \_\_gnu\_debug::basic\_string, 1014–1018
  - std::basic\_string, 1996–2000
  - std::deque, 2321, 2322
  - std::list, 2544, 2545
  - std::map, 2594–2596
  - std::multimap, 2675, 2676
  - std::multiset, 2694, 2695
  - std::set, 2887, 2888
  - std::vector, 3052–3054
- insert\_after
  - std::forward\_list, 2377, 2378
- insert\_iterator
  - std::insert\_iterator, 2474
- inserter
  - iterators, 231
- int\_type
  - \_\_gnu\_cxx::enc\_filebuf, 821
  - \_\_gnu\_cxx::stdio\_filebuf, 889
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 922
  - std::basic\_filebuf, 1456
  - std::basic\_fstream, 1493, 1494
  - std::basic\_ifstream, 1558
  - std::basic\_ios, 1609
  - std::basic\_iostream, 1644
  - std::basic\_istream, 1706
  - std::basic\_istreamstream, 1760
  - std::basic\_ofstream, 1813

- std::basic\_ostream, 1856
- std::basic\_ostringstream, 1901
- std::basic\_streambuf, 1940
- std::basic\_stringbuf, 2023
- std::basic\_stringstream, 2055, 2056
- std::istreambuf\_iterator, 2511
- internal
  - std, 580
  - std::basic\_fstream, 1547
  - std::basic\_ifstream, 1600
  - std::basic\_ios, 1631
  - std::basic\_iostream, 1696
  - std::basic\_istream, 1747
  - std::basic\_istream, 1802
  - std::basic\_ofstream, 1846
  - std::basic\_ostream, 1888
  - std::basic\_ostringstream, 1933
  - std::basic\_stringstream, 2108
  - std::ios\_base, 2493
- intervals
  - std::piecewise\_constant\_-distribution, 2817
  - std::piecewise\_linear\_distribution, 2823
- intl
  - std::moneypunct, 2650
  - std::moneypunct\_byname, 2662
- io
  - filebuf, 57
  - fstream, 57
  - ifstream, 57
  - ios, 57
  - iostream, 57
  - istream, 57
  - istringstream, 57
  - ofstream, 57
  - ostream, 58
  - ostringstream, 58
  - streambuf, 58
  - stringbuf, 58
  - stringstream, 58
  - wfilebuf, 58
  - wfstream, 58
  - wifstream, 59
  - wios, 59
  - wiostream, 59
  - wistream, 59
  - wstringstream, 59
  - wofstream, 59
  - wostream, 59
- io manip, 3246
- ios, 3247
  - io, 57
- ios\_base.h, 3247
- ios\_fwd, 3250
- iostate
  - std::basic\_fstream, 1494
  - std::basic\_ifstream, 1558
  - std::basic\_ios, 1609
  - std::basic\_iostream, 1644
  - std::basic\_istream, 1707
  - std::basic\_istream, 1760
  - std::basic\_ofstream, 1813
  - std::basic\_ostream, 1856
  - std::basic\_ostringstream, 1901
  - std::basic\_stringstream, 2056
  - std::ios\_base, 2481
- iostream, 3251
  - io, 57
- iota
  - SGIextensions, 17
  - std, 580
- is
  - std::\_\_ctype\_abstract\_base, 1261
  - std::ctype, 2221, 2222
  - std::ctype< char >, 2235, 2236
  - std::ctype< wchar\_t >, 2257
  - std::ctype\_byname, 2272, 2273
  - std::ctype\_byname< char >, 2285, 2286
- is\_bounded
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2756
- is\_exact
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2757
- is\_heap
  - heap\_algorithms, 221

- SGIextensions, 17
- is\_heap\_until
  - heap\_algorithms, 221, 222
- is\_iec559
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2757
- is\_integer
  - std::\_\_numeric\_limits\_base, 1340
  - std::numeric\_limits, 2757
- is\_modulo
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2757
- is\_open
  - \_\_gnu\_cxx::enc\_filebuf, 826
  - \_\_gnu\_cxx::stdio\_filebuf, 897
  - std::basic\_filebuf, 1462
  - std::basic\_fstream, 1512
  - std::basic\_ifstream, 1575
  - std::basic\_ofstream, 1823
- is\_partitioned
  - mutating\_algorithms, 129
- is\_signed
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2757
- is\_sorted
  - SGIextensions, 18
  - sorting\_algorithms, 164
- is\_sorted\_until
  - sorting\_algorithms, 165
- is\_specialized
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2758
- isalnum
  - std, 580
- isalpha
  - std, 580
- isctrl
  - std, 580
- isdigit
  - std, 581
- isgraph
  - std, 581
- islower
  - std, 581
- isprint
  - std, 581
- ispunct
  - std, 581
- isspace
  - std, 581
- istream, 3252
  - io, 57
- istream.tcc, 3253
- istream\_iterator
  - std::istream\_iterator, 2509
- istream\_type
  - std::istreambuf\_iterator, 2511
- istreambuf\_iterator
  - std::istreambuf\_iterator, 2513
- istreamstring
  - io, 57
- isupper
  - std, 581
- isxdigit
  - std, 581
- iter\_swap
  - mutating\_algorithms, 129
- iter\_type
  - std::money\_get, 2628
  - std::money\_put, 2634
  - std::num\_get, 2720
  - std::num\_put, 2740
  - std::time\_get, 2932
  - std::time\_get\_byname, 2943
  - std::time\_put, 2954
  - std::time\_put\_byname, 2959
- iterator, 3254
  - std::set, 2877
- Iterator Tags, 232
- iterator.h, 3254
- iterator\_category
  - std::back\_insert\_iterator, 1444
  - std::front\_insert\_iterator, 2390
  - std::insert\_iterator, 2473
  - std::istream\_iterator, 2508
  - std::istreambuf\_iterator, 2512
  - std::iterator, 2515
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2805
  - std::raw\_storage\_iterator, 2859
  - std::reverse\_iterator, 2865
- iterator\_tracker.h, 3255

- Iterators, 226
- iterators
  - \_\_iterator\_category, 230
  - back\_inserter, 230
  - front\_inserter, 230
  - inserter, 231
  - operator==, 231, 232
- isword
  - std::basic\_fstream, 1513
  - std::basic\_ifstream, 1575
  - std::basic\_ios, 1618
  - std::basic\_iostream, 1662
  - std::basic\_istream, 1722
  - std::basic\_istreamstream, 1777
  - std::basic\_ofstream, 1824
  - std::basic\_ostream, 1866
  - std::basic\_ostreamstream, 1911
  - std::basic\_stringstream, 2074
  - std::ios\_base, 2485
- k
  - std::negative\_binomial\_distribution, 2705
- key\_comp
  - std::map, 2596
  - std::multimap, 2676
  - std::multiset, 2696
  - std::set, 2888
- key\_compare
  - std::set, 2877
- key\_type
  - std::set, 2877
- kill\_dependency
  - atomics, 205
- L1\_cache\_size
  - \_\_gnu\_parallel::\_Settings, 1151
- L2\_cache\_size
  - \_\_gnu\_parallel::\_Settings, 1151
- laguerre
  - tr1\_math\_spec\_func, 110
- lambda
  - std::exponential\_distribution, 2352
- left
  - std, 582
  - std::basic\_fstream, 1547
  - std::basic\_ifstream, 1600
  - std::basic\_ios, 1631
  - std::basic\_iostream, 1696
  - std::basic\_istream, 1747
  - std::basic\_istreamstream, 1802
  - std::basic\_ofstream, 1846
  - std::basic\_ostream, 1888
  - std::basic\_ostreamstream, 1934
  - std::basic\_stringstream, 2108
  - std::ios\_base, 2493
- legendre
  - tr1\_math\_spec\_func, 110
- length
  - \_\_gnu\_cxx::\_\_versa\_string, 774
  - \_\_gnu\_debug::basic\_string, 1019
  - std::basic\_string, 2001
- lexicographical\_compare
  - sorting\_algorithms, 166
- lexicographical\_compare\_3way
  - SGIextensions, 18
- limits, 3257
- linear\_congruential\_engine
  - std::linear\_congruential\_engine, 2526
- list, 3259–3261
  - std::list, 2535–2537
- list.tcc, 3262
- list\_partition
  - \_\_gnu\_parallel, 399
- list\_partition.h, 3262
- list\_update\_policy.hpp, 3263
- locale, 3263
  - std::locale, 2557, 2558
- locale\_classes.h, 3263
- locale\_classes.tcc, 3264
- locale\_facets.h, 3265
- locale\_facets.tcc, 3268
- locale\_facets\_nonio.h, 3268
- locale\_facets\_nonio.tcc, 3270
- localefwd.h, 3270
- Locales, 206
- lock
  - mutexes, 67
- log
  - complex\_numbers, 42
- log10

- complex\_numbers, 42
- logic\_error
  - std::logic\_error, 2570
- losertree.h, 3271
- lower\_bound
  - binary\_search\_algorithms, 192, 193
  - std::map, 2596, 2597
  - std::multimap, 2677
  - std::multiset, 2696
  - std::set, 2889
- lowest
  - std::numeric\_limits, 2754
- macros.h, 3273
  - \_\_GLIBCXX\_DEBUG\_VERIFY, 3275
  - \_\_glibcxx\_check\_erase, 3274
  - \_\_glibcxx\_check\_erase\_range, 3274
  - \_\_glibcxx\_check\_heap\_pred, 3274
  - \_\_glibcxx\_check\_insert, 3274
  - \_\_glibcxx\_check\_insert\_range, 3274
  - \_\_glibcxx\_check\_partitioned\_lower, 3275
  - \_\_glibcxx\_check\_partitioned\_lower\_pred, 3275
  - \_\_glibcxx\_check\_partitioned\_upper\_pred, 3275
  - \_\_glibcxx\_check\_sorted\_pred, 3275
- make\_heap
  - heap\_algorithms, 222, 223
- make\_shared
  - pointer\_abstractions, 64
- malloc\_allocator.h, 3276
- map, 3276, 3277
  - std::map, 2585–2587
- map.h, 3277, 3278
- mask\_array
  - numeric\_arrays, 84
- mask\_array.h, 3279
- Mathematical Special Functions, 104
- max
  - \_\_gnu\_parallel, 400
  - numeric\_arrays, 87
  - sorting\_algorithms, 167
  - std::bernoulli\_distribution, 2112
  - std::binomial\_distribution, 2123
  - std::cauchy\_distribution, 2142
  - std::chi\_squared\_distribution, 2151
  - std::discard\_block\_engine, 2332
  - std::discrete\_distribution, 2337
  - std::exponential\_distribution, 2352
  - std::extreme\_value\_distribution, 2356
  - std::fisher\_f\_distribution, 2360
  - std::gamma\_distribution, 2411
  - std::geometric\_distribution, 2416
  - std::independent\_bits\_engine, 2464
  - std::linear\_congruential\_engine, 2527
  - std::lognormal\_distribution, 2578
  - std::negative\_binomial\_distribution, 2705
  - std::normal\_distribution, 2711
  - std::numeric\_limits, 2754
  - std::piecewise\_constant\_distribution, 2818
  - std::piecewise\_linear\_distribution, 2823
  - std::poisson\_distribution, 2834
  - std::shuffle\_order\_engine, 2910
  - std::student\_t\_distribution, 2922
  - std::uniform\_int\_distribution, 3013
  - std::uniform\_real\_distribution, 3017
  - std::weibull\_distribution, 3066
- max\_digits10
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2758
- max\_element
  - sorting\_algorithms, 168
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1151
- max\_exponent
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2758
- max\_exponent10
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2758
- max\_size
  - \_\_gnu\_cxx::\_\_versa\_string, 774
  - \_\_gnu\_debug::basic\_string, 1019
  - std::basic\_string, 2001
  - std::deque, 2323

- std::forward\_list, 2379
- std::list, 2546
- std::map, 2597
- std::multimap, 2678
- std::multiset, 2697
- std::set, 2889
- std::vector, 3054
- mean
  - std::normal\_distribution, 2711
  - std::poisson\_distribution, 2834
- mem\_fn
  - functors, 212
- Memory, 61
- memory, 3280
- memory\_order
  - atomics, 204
- merge
  - sorting\_algorithms, 169
  - std::forward\_list, 2379
  - std::list, 2546
- merge.h, 3281
- merge\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1152
- merge\_oversampling
  - \_\_gnu\_parallel::\_Settings, 1152
- messages
  - std::locale, 2563
  - std::messages, 2615
- messages\_members.h, 3282
- metaprogramming
  - false\_type, 117
  - true\_type, 117
- microseconds
  - std::chrono, 682
- milliseconds
  - std::chrono, 682
- min
  - \_\_gnu\_parallel, 400
  - numeric\_arrays, 87
  - sorting\_algorithms, 170
  - std::bernoulli\_distribution, 2112
  - std::binomial\_distribution, 2123
  - std::cauchy\_distribution, 2142
  - std::chi\_squared\_distribution, 2151
  - std::discard\_block\_engine, 2333
  - std::discrete\_distribution, 2337
  - std::exponential\_distribution, 2352
  - std::extreme\_value\_distribution, 2356
  - std::fisher\_f\_distribution, 2360
  - std::gamma\_distribution, 2411
  - std::geometric\_distribution, 2416
  - std::independent\_bits\_engine, 2464
  - std::linear\_congruential\_engine, 2527
  - std::lognormal\_distribution, 2578
  - std::negative\_binomial\_distribution, 2705
  - std::normal\_distribution, 2712
  - std::numeric\_limits, 2755
  - std::piecewise\_constant\_distribution, 2818
  - std::piecewise\_linear\_distribution, 2823
  - std::poisson\_distribution, 2834
  - std::shuffle\_order\_engine, 2911
  - std::student\_t\_distribution, 2922
  - std::uniform\_int\_distribution, 3013
  - std::uniform\_real\_distribution, 3017
  - std::weibull\_distribution, 3066
- min\_element
  - sorting\_algorithms, 171
- min\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1152
- min\_exponent
  - std::\_\_numeric\_limits\_base, 1341
  - std::numeric\_limits, 2758
- min\_exponent10
  - std::\_\_numeric\_limits\_base, 1342
  - std::numeric\_limits, 2758
- minmax
  - sorting\_algorithms, 172
- minmax\_element
  - sorting\_algorithms, 172, 173
- minstd\_rand
  - random\_generators, 239
- minstd\_rand0
  - random\_generators, 239
- minutes
  - std::chrono, 682
- mismatch
  - non\_mutating\_algorithms, 155

- modulus
  - std::linear\_congruential\_engine, 2530
- monetary
  - std::locale, 2563
- money\_get
  - std::money\_get, 2629
- money\_put
  - std::money\_put, 2634
- money\_punct
  - std::money\_punct, 2641
- move
  - mutating\_algorithms, 130
- move.h, 3282
- move\_backward
  - mutating\_algorithms, 130
- mt19937
  - random\_generators, 239
- mt19937\_64
  - random\_generators, 239
- mt\_allocator.h, 3283
- multimap
  - std::multimap, 2666–2668
- multimap.h, 3285, 3286
- multiplier
  - std::linear\_congruential\_engine, 2530
- multiseq\_partition
  - \_\_gnu\_parallel, 400
- multiseq\_selection
  - \_\_gnu\_parallel, 401
- multiseq\_selection.h, 3287
- multiset
  - std::multiset, 2686–2688
- multiset.h, 3288, 3289
- multiway\_merge
  - \_\_gnu\_parallel, 402
- multiway\_merge.h, 3290
  - \_GLIBCXX\_PARALLEL\_LENGTH, 3296
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, 404
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, 404
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, 405
- multiway\_merge\_loser\_tree
  - \_\_gnu\_parallel, 406
- multiway\_merge\_loser\_tree\_sentinel
  - \_\_gnu\_parallel, 406
- multiway\_merge\_loser\_tree\_unguarded
  - \_\_gnu\_parallel, 407
- multiway\_merge\_minimal\_k
  - \_\_gnu\_parallel::\_Settings, 1152
- multiway\_merge\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1152
- multiway\_merge\_oversampling
  - \_\_gnu\_parallel::\_Settings, 1152
- multiway\_merge\_samplng\_splitting
  - \_\_gnu\_parallel, 408
- multiway\_merge\_sentinels
  - \_\_gnu\_parallel, 408
- multiway\_mergesort.h, 3296
- Mutating, 122
- mutating\_algorithms
  - copy, 125
  - copy\_backward, 125
  - copy\_if, 126
  - copy\_n, 126
  - fill, 127
  - fill\_n, 127
  - generate, 128
  - generate\_n, 128
  - is\_partitioned, 129
  - iter\_swap, 129
  - move, 130
  - move\_backward, 130
  - partition, 131
  - partition\_copy, 131
  - partition\_point, 132
  - random\_shuffle, 132, 133
  - remove, 133
  - remove\_copy, 134
  - remove\_copy\_if, 134
  - remove\_if, 135
  - replace, 136
  - replace\_copy\_if, 136
  - replace\_if, 137
  - reverse, 137
  - reverse\_copy, 138
  - rotate, 138
  - rotate\_copy, 139

- shuffle, 139
- stable\_partition, 140
- swap, 141
- swap\_ranges, 141
- transform, 141, 142
- unique, 143
- unique\_copy, 144
- mutex, 3297
- Mutexes, 65
- mutexes
  - call\_once, 67
  - lock, 67
  - try\_lock, 67
- name
  - std::locale, 2559
  - std::type\_info, 3005
- nanoseconds
  - std::chrono, 682
- narrow
  - std::\_\_ctype\_abstract\_base, 1262
  - std::basic\_fstream, 1513
  - std::basic\_ifstream, 1576
  - std::basic\_ios, 1619
  - std::basic\_iostream, 1662
  - std::basic\_istream, 1722
  - std::basic\_istream, 1777
  - std::basic\_ofstream, 1824
  - std::basic\_ostream, 1867
  - std::basic\_ostringstream, 1912
  - std::basic\_stringstream, 2075
  - std::ctype, 2222, 2223
  - std::ctype< char >, 2236, 2237
  - std::ctype< wchar\_t >, 2258
  - std::ctype\_byname, 2273, 2274
  - std::ctype\_byname< char >, 2286, 2287
- native\_handle
  - std::thread, 2927
- neg\_format
  - std::moneypunct, 2647
  - std::moneypunct\_byname, 2659
- negative\_sign
  - std::moneypunct, 2648
  - std::moneypunct\_byname, 2660
- Negators, 215
- negators
  - not1, 216
  - not2, 216
- nested\_exception.h, 3299
- new, 3300
  - operator delete, 3301
  - operator new, 3303
- new\_allocator.h, 3305
- new\_handler
  - std, 553
- next\_permutation
  - sorting\_algorithms, 173, 174
- noboolalpha
  - std, 582
- Non-Mutating, 145
- non\_mutating\_algorithms
  - adjacent\_find, 147
  - all\_of, 148
  - any\_of, 148
  - count, 149
  - count\_if, 149
  - equal, 149, 150
  - find, 150
  - find\_end, 151, 152
  - find\_first\_of, 152, 153
  - find\_if, 154
  - find\_if\_not, 154
  - for\_each, 154
  - mismatch, 155
  - none\_of, 156
  - search, 156, 157
  - search\_n, 158
- none
  - std::bitset, 2134
  - std::locale, 2563
- none\_of
  - non\_mutating\_algorithms, 156
- norm
  - complex\_numbers, 42
- Normal, 248
- normal\_distribution
  - std::normal\_distribution, 2711
- noshowbase
  - std, 582
- noshowpoint
  - std, 582

- noshowpos
  - std, 582
- noskipws
  - std, 582
- not1
  - negators, 216
- not2
  - negators, 216
- nounitbuf
  - std, 583
- nouppercase
  - std, 583
- npos
  - \_\_gnu\_cxx::\_\_versa\_string, 792
  - \_\_gnu\_debug::basic\_string, 1033
  - std::basic\_string, 2018
- nth\_element
  - sorting\_algorithms, 174, 175
- nth\_element\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1153
- num\_get
  - std::num\_get, 2721
- num\_put
  - std::num\_put, 2741
- numeric, 3306, 3307
  - std::locale, 2563
- Numeric Arrays, 73
- numeric\_arrays
  - ~gslice, 86
  - apply, 86
  - cshift, 87
  - gslice, 83
  - gslice\_array, 83
  - indirect\_array, 83
  - mask\_array, 84
  - max, 87
  - min, 87
  - operator<<=, 91
  - operator>>=, 96
  - operator\*=, 89
  - operator~, 101
  - operator^=, 100, 101
  - operator+, 89
  - operator+=, 89, 90
  - operator-, 90
  - operator-=, 90
  - operator/=: 91
  - operator=: 92–96
  - operator%=, 88
  - operator&=: 88
  - resize, 102
  - shift, 102
  - size, 102, 103
  - slice, 84
  - slice\_array, 84
  - start, 103
  - stride, 103
  - sum, 103
  - valarray, 84–86
- numeric\_traits.h, 3310
- numeric\_fwd.h, 3310
- Numerics, 68
- numpunct
  - std::numpunct, 2786, 2787
- oct
  - std, 583
  - std::basic\_fstream, 1547
  - std::basic\_ifstream, 1601
  - std::basic\_ios, 1631
  - std::basic\_iostream, 1696
  - std::basic\_istream, 1747
  - std::basic\_istreamstream, 1802
  - std::basic\_ofstream, 1846
  - std::basic\_ostream, 1889
  - std::basic\_ostreamstream, 1934
  - std::basic\_stringstream, 2109
  - std::ios\_base, 2493
- off\_type
  - \_\_gnu\_cxx::enc\_filebuf, 821
  - \_\_gnu\_cxx::stdio\_filebuf, 890
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 922
  - std::basic\_filebuf, 1456
  - std::basic\_fstream, 1494, 1495
  - std::basic\_ifstream, 1559
  - std::basic\_ios, 1610
  - std::basic\_iostream, 1645
  - std::basic\_istream, 1707
  - std::basic\_istreamstream, 1761
  - std::basic\_ofstream, 1813
  - std::basic\_ostream, 1857
  - std::basic\_ostreamstream, 1901

- std::basic\_streambuf, 1940
- std::basic\_stringbuf, 2023
- std::basic\_stringstream, 2056, 2057
- ofstream
  - io, 57
- omp\_loop.h, 3313
- omp\_loop\_static.h, 3314
- open
  - \_\_gnu\_cxx::enc\_filebuf, 826
  - \_\_gnu\_cxx::stdio\_filebuf, 897, 898
  - std::basic\_filebuf, 1462, 1463
  - std::basic\_fstream, 1514
  - std::basic\_ifstream, 1576
  - std::basic\_ofstream, 1825
- openmode
  - std::basic\_fstream, 1495
  - std::basic\_ifstream, 1559
  - std::basic\_ios, 1610
  - std::basic\_iostream, 1645
  - std::basic\_istream, 1707
  - std::basic\_istream, 1761
  - std::basic\_ofstream, 1814
  - std::basic\_ostream, 1857
  - std::basic\_ostringstream, 1902
  - std::basic\_stringstream, 2057
  - std::ios\_base, 2482
- operator \_Iterator
  - \_\_gnu\_debug::\_Safe\_iterator, 967
- operator \_RAIter
  - \_\_gnu\_parallel::\_GuardedIterator, 1089
- operator bool
  - std::basic\_istream::sentry, 1751
  - std::basic\_ostream::sentry, 1892
  - std::function< \_Res(\_- ArgTypes...)>, 2396
- operator delete
  - new, 3301
- operator new
  - new, 3303
- operator streamoff
  - std::fpos, 2387
- operator void \*
  - std::basic\_fstream, 1514
  - std::basic\_ifstream, 1577
  - std::basic\_ios, 1619
- std::basic\_iostream, 1663
- std::basic\_istream, 1723
- std::basic\_istream, 1778
- std::basic\_ofstream, 1825
- std::basic\_ostream, 1867
- std::basic\_ostringstream, 1912
- std::basic\_stringstream, 2075
- operator<
  - \_\_gnu\_cxx, 327, 328
  - \_\_gnu\_parallel::\_GuardedIterator, 1090
  - std, 589–595
  - std::multiset, 2700
- operator<<
  - complex\_numbers, 46
  - pointer\_abstractions, 65
  - random\_distributions\_bernoulli, 254
  - random\_distributions\_normal, 250
  - random\_distributions\_poisson, 260, 261
  - random\_distributions\_uniform, 245, 246
  - random\_generators, 242
  - std, 596–602
  - std::basic\_fstream, 1515–1521
  - std::basic\_iostream, 1663–1670
  - std::basic\_ofstream, 1826–1832
  - std::basic\_ostream, 1868–1874
  - std::basic\_ostringstream, 1913–1919
  - std::basic\_stringstream, 2075–2082
  - std::binomial\_distribution, 2125
  - std::bitset, 2135
  - std::chi\_squared\_distribution, 2152
  - std::discard\_block\_engine, 2334
  - std::discrete\_distribution, 2338
  - std::fisher\_f\_distribution, 2361
  - std::gamma\_distribution, 2413
  - std::linear\_congruential\_engine, 2529
  - std::lognormal\_distribution, 2579
  - std::negative\_binomial\_distribution, 2707
  - std::normal\_distribution, 2714
  - std::piecewise\_constant\_distribution, 2819

- std::piecewise\_linear\_distribution, 2824
- std::poisson\_distribution, 2836
- std::shuffle\_order\_engine, 2912
- std::student\_t\_distribution, 2923
- operator<<=
  - numeric\_arrays, 91
  - std::bitset, 2135
  - std::indirect\_array, 2469
  - std::mask\_array, 2604
  - std::slice\_array, 2916
- operator<=
  - \_\_gnu\_cxx, 329, 330
  - \_\_gnu\_parallel::\_GuardedIterator, 1091
  - std, 603–606
  - std::rel\_ops, 696
- operator>
  - \_\_gnu\_cxx, 332
  - std, 613–616
  - std::rel\_ops, 696
- operator>>
  - complex\_numbers, 47
  - random\_distributions\_bernoulli, 255, 256
  - random\_distributions\_normal, 251
  - random\_distributions\_poisson, 263, 264
  - random\_distributions\_uniform, 247
  - std, 620–625
  - std::basic\_fstream, 1522–1528
  - std::basic\_ifstream, 1577–1584
  - std::basic\_istream, 1670–1676
  - std::basic\_istream, 1723–1730
  - std::basic\_istream, 1778–1784
  - std::basic\_stringstream, 2082–2088
  - std::binomial\_distribution, 2125
  - std::bitset, 2136
  - std::chi\_squared\_distribution, 2153
  - std::discard\_block\_engine, 2335
  - std::discrete\_distribution, 2339
  - std::fisher\_f\_distribution, 2361
  - std::gamma\_distribution, 2414
  - std::independent\_bits\_engine, 2466
  - std::linear\_congruential\_engine, 2529
  - std::lognormal\_distribution, 2580
  - std::negative\_binomial\_distribution, 2707
  - std::normal\_distribution, 2714
  - std::piecewise\_constant\_distribution, 2819
  - std::piecewise\_linear\_distribution, 2825
  - std::poisson\_distribution, 2836
  - std::shuffle\_order\_engine, 2913
  - std::student\_t\_distribution, 2923
- operator>>=
  - numeric\_arrays, 96
  - std::bitset, 2136
  - std::indirect\_array, 2469
  - std::mask\_array, 2604
  - std::slice\_array, 2916
- operator>=
  - \_\_gnu\_cxx, 333, 334
  - std, 616–619
  - std::rel\_ops, 696
- operator\*
  - \_\_gnu\_debug::\_Safe\_iterator, 967
  - \_\_gnu\_parallel::\_GuardedIterator, 1090
  - complex\_numbers, 43
  - std::auto\_ptr, 1439
  - std::back\_insert\_iterator, 1445
  - std::front\_insert\_iterator, 2391
  - std::insert\_iterator, 2474
  - std::istreambuf\_iterator, 2513
  - std::ostreambuf\_iterator, 2807
  - std::reverse\_iterator, 2867
- operator\*=
  - complex\_numbers, 43, 44
  - numeric\_arrays, 89
  - std::indirect\_array, 2469
  - std::mask\_array, 2604
  - std::slice\_array, 2916
- operator~
  - numeric\_arrays, 101
  - std::bitset, 2138
- operator^
  - std, 626
- operator^=
  - numeric\_arrays, 100, 101

- std::bitset, 2137
- std::indirect\_array, 2469
- std::mask\_array, 2604
- std::slice\_array, 2916
- operator()
  - \_\_gnu\_cxx::subtractive\_rng, 943
  - \_\_gnu\_parallel::\_Nothing, 1131
  - \_\_gnu\_parallel::\_RandomNumber, 1143
  - \_\_gnu\_parallel::\_\_accumulate\_selector, 1035
  - \_\_gnu\_parallel::\_\_adjacent\_find\_selector, 1039
  - \_\_gnu\_parallel::\_\_count\_if\_selector, 1044
  - \_\_gnu\_parallel::\_\_count\_selector, 1046
  - \_\_gnu\_parallel::\_\_fill\_selector, 1048
  - \_\_gnu\_parallel::\_\_find\_first\_of\_selector, 1050
  - \_\_gnu\_parallel::\_\_find\_if\_selector, 1052
  - \_\_gnu\_parallel::\_\_for\_each\_selector, 1053
  - \_\_gnu\_parallel::\_\_generate\_selector, 1055
  - \_\_gnu\_parallel::\_\_identity\_selector, 1059
  - \_\_gnu\_parallel::\_\_inner\_product\_selector, 1061
  - \_\_gnu\_parallel::\_\_mismatch\_selector, 1065
  - \_\_gnu\_parallel::\_\_replace\_if\_selector, 1071
  - \_\_gnu\_parallel::\_\_replace\_selector, 1073
  - \_\_gnu\_parallel::\_\_transform1\_selector, 1075
  - \_\_gnu\_parallel::\_\_transform2\_selector, 1077
- std::bernoulli\_distribution, 2112
- std::binomial\_distribution, 2123
- std::cauchy\_distribution, 2143
- std::chi\_squared\_distribution, 2151
- std::discard\_block\_engine, 2333
- std::discrete\_distribution, 2337
- std::exponential\_distribution, 2352
- std::extreme\_value\_distribution, 2356
- std::fisher\_f\_distribution, 2360
- std::function<\_Res(\_ArgTypes...)>, 2396
- std::gamma\_distribution, 2412
- std::geometric\_distribution, 2417
- std::independent\_bits\_engine, 2465
- std::linear\_congruential\_engine, 2527
- std::locale, 2560
- std::lognormal\_distribution, 2578
- std::negative\_binomial\_distribution, 2705
- std::normal\_distribution, 2712
- std::piecewise\_constant\_distribution, 2818
- std::piecewise\_linear\_distribution, 2823
- std::poisson\_distribution, 2834
- std::shuffle\_order\_engine, 2911
- std::student\_t\_distribution, 2922
- std::uniform\_int\_distribution, 3014
- std::uniform\_real\_distribution, 3017
- std::weibull\_distribution, 3066
- operator+
  - \_\_gnu\_cxx, 325–327
  - complex\_numbers, 44
  - numeric\_arrays, 89
  - std, 587–589
  - std::fpos, 2387
  - std::reverse\_iterator, 2868
- operator++
  - \_\_gnu\_debug::\_Safe\_iterator, 967
  - \_\_gnu\_parallel::\_GuardedIterator, 1090
  - std::back\_insert\_iterator, 1445
  - std::front\_insert\_iterator, 2391, 2392
  - std::insert\_iterator, 2474, 2475
  - std::istreambuf\_iterator, 2514
  - std::ostreambuf\_iterator, 2807
  - std::reverse\_iterator, 2868
- operator+=
  - \_\_gnu\_cxx::\_\_versa\_string, 774–776

- [\\_\\_gnu\\_debug::basic\\_string](#), [1019](#), [1020](#)
  - [complex\\_numbers](#), [44](#)
  - [numeric\\_arrays](#), [89](#), [90](#)
  - [std::basic\\_string](#), [2001](#), [2002](#)
  - [std::complex](#), [2202](#)
  - [std::fpos](#), [2387](#)
  - [std::indirect\\_array](#), [2469](#)
  - [std::mask\\_array](#), [2604](#)
  - [std::reverse\\_iterator](#), [2868](#)
  - [std::slice\\_array](#), [2916](#)
- operator-
  - [complex\\_numbers](#), [45](#)
  - [numeric\\_arrays](#), [90](#)
  - [std::fpos](#), [2388](#)
  - [std::reverse\\_iterator](#), [2869](#)
- operator->
  - [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#), [968](#)
  - [std::auto\\_ptr](#), [1439](#)
  - [std::reverse\\_iterator](#), [2870](#)
- operator--
  - [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#), [968](#)
  - [std::reverse\\_iterator](#), [2869](#)
- operator=
  - [complex\\_numbers](#), [45](#)
  - [numeric\\_arrays](#), [90](#)
  - [std::complex](#), [2202](#)
  - [std::fpos](#), [2388](#)
  - [std::indirect\\_array](#), [2469](#)
  - [std::mask\\_array](#), [2604](#)
  - [std::reverse\\_iterator](#), [2870](#)
  - [std::slice\\_array](#), [2916](#)
- operator/
  - [complex\\_numbers](#), [45](#), [46](#)
- operator/=
  - [complex\\_numbers](#), [46](#)
  - [numeric\\_arrays](#), [91](#)
  - [std::indirect\\_array](#), [2469](#)
  - [std::mask\\_array](#), [2604](#)
  - [std::slice\\_array](#), [2916](#)
- operator=
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#), [776](#), [777](#)
  - [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#), [969](#)
  - [complex\\_numbers](#), [47](#)
  - [numeric\\_arrays](#), [92–96](#)
  - [std::auto\\_ptr](#), [1440](#)
  - [std::back\\_insert\\_iterator](#), [1445](#)
  - [std::basic\\_string](#), [2003](#), [2004](#)
  - [std::deque](#), [2323](#), [2324](#)
  - [std::forward\\_list](#), [2380](#)
  - [std::front\\_insert\\_iterator](#), [2392](#)
  - [std::function<\\_Res\(\\_ArgTypes...\)>](#), [2397–2399](#)
  - [std::insert\\_iterator](#), [2475](#)
  - [std::list](#), [2547](#)
  - [std::locale](#), [2560](#)
  - [std::map](#), [2597](#), [2598](#)
  - [std::multimap](#), [2678](#), [2679](#)
  - [std::multiset](#), [2697](#), [2698](#)
  - [std::ostream\\_iterator](#), [2803](#)
  - [std::ostreambuf\\_iterator](#), [2807](#)
  - [std::set](#), [2890](#)
  - [std::vector](#), [3054](#), [3055](#)
- operator==
  - [\\_\\_gnu\\_cxx](#), [330](#), [331](#)
  - [complex\\_numbers](#), [47](#)
  - [iterators](#), [231](#), [232](#)
  - [random\\_distributions\\_bernoulli](#), [255](#)
  - [random\\_distributions\\_normal](#), [251](#)
  - [random\\_distributions\\_poisson](#), [261](#), [262](#)
  - [random\\_distributions\\_uniform](#), [246](#)
  - [std](#), [606–613](#)
  - [std::binomial\\_distribution](#), [2125](#)
  - [std::bitset](#), [2135](#)
  - [std::chi\\_squared\\_distribution](#), [2152](#)
  - [std::discard\\_block\\_engine](#), [2334](#)
  - [std::fisher\\_f\\_distribution](#), [2361](#)
  - [std::gamma\\_distribution](#), [2413](#)
  - [std::independent\\_bits\\_engine](#), [2466](#)
  - [std::linear\\_congruential\\_engine](#), [2529](#)
  - [std::locale](#), [2561](#)
  - [std::lognormal\\_distribution](#), [2579](#)
  - [std::multiset](#), [2701](#)
  - [std::negative\\_binomial\\_distribution](#), [2707](#)
  - [std::normal\\_distribution](#), [2714](#)
  - [std::poisson\\_distribution](#), [2836](#)
  - [std::shuffle\\_order\\_engine](#), [2912](#)
  - [std::student\\_t\\_distribution](#), [2923](#)

- operator%=
  - numeric\_arrays, 88
  - std::indirect\_array, 2468
  - std::mask\_array, 2603
  - std::slice\_array, 2916
- operator&
  - std, 587
- operator&=
  - numeric\_arrays, 88
  - std::bitset, 2135
  - std::indirect\_array, 2468
  - std::mask\_array, 2603
  - std::slice\_array, 2916
- os\_defines.h, 3314
- ostream, 3315
  - io, 58
- ostream.tcc, 3316
- ostream\_insert.h, 3317
- ostream\_iterator
  - std::ostream\_iterator, 2802
- ostream\_type
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2805
- ostreambuf\_iterator
  - std::ostreambuf\_iterator, 2806
- ostringstream
  - io, 58
- out
  - std::\_\_codecvt\_abstract\_base, 1249
  - std::basic\_fstream, 1548
  - std::basic\_ifstream, 1601
  - std::basic\_ios, 1632
  - std::basic\_iostream, 1696
  - std::basic\_istream, 1747
  - std::basic\_istreamstream, 1803
  - std::basic\_ofstream, 1847
  - std::basic\_ostream, 1889
  - std::basic\_ostringstream, 1934
  - std::basic\_stringstream, 2109
  - std::codecvt, 2162
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2168
  - std::codecvt< char, char, mbstate\_t >, 2173
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2178
  - std::codecvt\_byname, 2185
  - std::ios\_base, 2494
- overflow
  - \_\_gnu\_cxx::enc\_filebuf, 827
  - \_\_gnu\_cxx::stdio\_filebuf, 898
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 926
  - std::basic\_filebuf, 1463
  - std::basic\_streambuf, 1945
  - std::basic\_stringbuf, 2028
- p
  - std::bernoulli\_distribution, 2112
  - std::binomial\_distribution, 2124
  - std::geometric\_distribution, 2417
  - std::negative\_binomial\_distribution, 2706
- pair
  - std::pair, 2815
- par\_loop.h, 3317
- parallel.h, 3318
- parallel\_balanced
  - \_\_gnu\_parallel, 365
- parallel\_omp\_loop
  - \_\_gnu\_parallel, 365
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, 365
- parallel\_taskqueue
  - \_\_gnu\_parallel, 365
- parallel\_unbalanced
  - \_\_gnu\_parallel, 365
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, 410
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, 411
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, 411
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, 1178
- param
  - std::bernoulli\_distribution, 2112, 2113
  - std::binomial\_distribution, 2124
  - std::cauchy\_distribution, 2143
  - std::chi\_squared\_distribution, 2151, 2152

- std::discrete\_distribution, 2337, 2338
- std::exponential\_distribution, 2353
- std::extreme\_value\_distribution, 2357
- std::fisher\_f\_distribution, 2360
- std::gamma\_distribution, 2412
- std::geometric\_distribution, 2417
- std::lognormal\_distribution, 2578, 2579
- std::negative\_binomial\_distribution, 2706
- std::normal\_distribution, 2712, 2713
- std::piecewise\_constant\_distribution, 2818
- std::piecewise\_linear\_distribution, 2824
- std::poisson\_distribution, 2835
- std::student\_t\_distribution, 2922
- std::uniform\_int\_distribution, 3014
- std::uniform\_real\_distribution, 3018
- std::weibull\_distribution, 3067
- partial\_sort
  - sorting\_algorithms, 175, 176
- partial\_sort\_copy
  - sorting\_algorithms, 177
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1153
- partial\_sum
  - std, 627, 628
- partial\_sum.h, 3318
- partial\_sum\_dilation
  - \_\_gnu\_parallel::\_Settings, 1153
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1153
- partition
  - mutating\_algorithms, 131
- partition.h, 3319
  - \_GLIBCXX\_VOLATILE, 3320
- partition\_chunk\_share
  - \_\_gnu\_parallel::\_Settings, 1153
- partition\_chunk\_size
  - \_\_gnu\_parallel::\_Settings, 1153
- partition\_copy
  - mutating\_algorithms, 131
- partition\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- partition\_point
  - mutating\_algorithms, 132
- pbackfail
  - \_\_gnu\_cxx::enc\_filebuf, 827
  - \_\_gnu\_cxx::stdio\_filebuf, 899
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 927
  - std::basic\_filebuf, 1464
  - std::basic\_streambuf, 1946
  - std::basic\_stringbuf, 2028
- pbase
  - \_\_gnu\_cxx::enc\_filebuf, 828
  - \_\_gnu\_cxx::stdio\_filebuf, 899
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 927
  - std::basic\_filebuf, 1464
  - std::basic\_streambuf, 1946
  - std::basic\_stringbuf, 2029
- pbump
  - \_\_gnu\_cxx::enc\_filebuf, 828
  - \_\_gnu\_cxx::stdio\_filebuf, 900
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 928
  - std::basic\_filebuf, 1465
  - std::basic\_streambuf, 1947
  - std::basic\_stringbuf, 2029
- peek
  - std::basic\_fstream, 1528
  - std::basic\_ifstream, 1584
  - std::basic\_iostream, 1677
  - std::basic\_istream, 1730
  - std::basic\_istreamstream, 1785
  - std::basic\_stringstream, 2089
- pod\_char\_traits.h, 3320
- pointer
  - std::back\_insert\_iterator, 1444
  - std::front\_insert\_iterator, 2390
  - std::insert\_iterator, 2473
  - std::istream\_iterator, 2508
  - std::istreambuf\_iterator, 2512
  - std::iterator, 2516
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2805
  - std::raw\_storage\_iterator, 2859
  - std::reverse\_iterator, 2865
  - std::set, 2877
- Pointer Abstractions, 61
- pointer.h, 3321

- pointer\_abstractions
  - allocate\_shared, 64
  - get\_deleter, 64
  - make\_shared, 64
  - operator<<, 65
- pointer\_adaptors
  - ptr\_fun, 217
- Poisson, 256
- polar
  - complex\_numbers, 48
- Policy-Based Data Structures, 235
- pool\_allocator.h, 3324
- pop
  - std::priority\_queue, 2840
  - std::queue, 2847
  - std::stack, 2919
- pop\_back
  - \_\_gnu\_parallel::\_-
    - RestrictedBoundedConcurrentQueue, 1145
  - std::deque, 2325
  - std::list, 2548
  - std::vector, 3056
- pop\_front
  - \_\_gnu\_parallel::\_-
    - RestrictedBoundedConcurrentQueue, 1145
  - std::deque, 2325
  - std::forward\_list, 2381
  - std::list, 2548
- pop\_heap
  - heap\_algorithms, 223
- pos\_format
  - std::moneypunct, 2648
  - std::moneypunct\_byname, 2660
- pos\_type
  - \_\_gnu\_cxx::enc\_filebuf, 821
  - \_\_gnu\_cxx::stdio\_filebuf, 890
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 922
  - std::basic\_filebuf, 1456
  - std::basic\_fstream, 1495, 1496
  - std::basic\_ifstream, 1559
  - std::basic\_ios, 1610
  - std::basic\_iostream, 1646
  - std::basic\_istream, 1708
  - std::basic\_istream, 1761
  - std::basic\_ofstream, 1814
  - std::basic\_ostream, 1858
  - std::basic\_ostringstream, 1902
  - std::basic\_streambuf, 1941
  - std::basic\_stringbuf, 2023
  - std::basic\_stringstream, 2057, 2058
- positive\_sign
  - std::moneypunct, 2649
  - std::moneypunct\_byname, 2661
- postypes.h, 3324
- pow
  - complex\_numbers, 48
- power
  - SGIextensions, 19
- pptr
  - \_\_gnu\_cxx::enc\_filebuf, 829
  - \_\_gnu\_cxx::stdio\_filebuf, 900
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 928
  - std::basic\_filebuf, 1465
  - std::basic\_streambuf, 1947
  - std::basic\_stringbuf, 2030
- precision
  - std::basic\_fstream, 1528, 1529
  - std::basic\_ifstream, 1584, 1585
  - std::basic\_ios, 1620
  - std::basic\_iostream, 1677
  - std::basic\_istream, 1730, 1731
  - std::basic\_istream, 1785, 1786
  - std::basic\_ofstream, 1833
  - std::basic\_ostream, 1874, 1875
  - std::basic\_ostringstream, 1919, 1920
  - std::basic\_stringstream, 2089
  - std::ios\_base, 2485, 2486
- prev\_permutation
  - sorting\_algorithms, 178
- priority\_queue
  - std::priority\_queue, 2839
- priority\_queue.hpp, 3325
- priority\_queue\_base\_dispatch.hpp, 3326
- probabilities
  - std::discrete\_distribution, 2338
- profiler.h, 3326
- profiler\_hashtable\_size.h, 3329
- profiler\_list\_to\_slist.h, 3330
- profiler\_list\_to\_vector.h, 3331
- profiler\_map\_to\_unordered\_map.h, 3332

- 
- profiler\_node.h, 3333
  - profiler\_state.h, 3334
  - profiler\_trace.h, 3335
  - profiler\_vector\_size.h, 3337
  - profiler\_vector\_to\_list.h, 3338
  - ptr\_fun
    - pointer\_adaptors, 217
  - pubimbue
    - \_\_gnu\_cxx::enc\_filebuf, 829
    - \_\_gnu\_cxx::stdio\_filebuf, 901
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 929
    - std::basic\_filebuf, 1465
    - std::basic\_streambuf, 1948
    - std::basic\_stringbuf, 2030
  - pubseekoff
    - \_\_gnu\_cxx::enc\_filebuf, 829
    - \_\_gnu\_cxx::stdio\_filebuf, 901
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 929
    - std::basic\_filebuf, 1466
    - std::basic\_streambuf, 1948
    - std::basic\_stringbuf, 2030
  - pubseekpos
    - \_\_gnu\_cxx::enc\_filebuf, 830
    - \_\_gnu\_cxx::stdio\_filebuf, 901
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 929
    - std::basic\_filebuf, 1466
    - std::basic\_streambuf, 1948
    - std::basic\_stringbuf, 2031
  - pubsetbuf
    - \_\_gnu\_cxx::enc\_filebuf, 830
    - \_\_gnu\_cxx::stdio\_filebuf, 902
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 930
    - std::basic\_filebuf, 1467
    - std::basic\_streambuf, 1949
    - std::basic\_stringbuf, 2031
  - pubsync
    - \_\_gnu\_cxx::enc\_filebuf, 830
    - \_\_gnu\_cxx::stdio\_filebuf, 902
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 930
    - std::basic\_filebuf, 1467
    - std::basic\_streambuf, 1949
    - std::basic\_stringbuf, 2031
  - push
    - std::priority\_queue, 2841
    - std::queue, 2847
    - std::stack, 2919
  - push\_back
    - \_\_gnu\_cxx::\_\_versa\_string, 778
    - \_\_gnu\_debug::basic\_string, 1021
    - std::basic\_string, 2005
    - std::deque, 2326
    - std::list, 2548
    - std::vector, 3057
  - push\_front
    - \_\_gnu\_parallel::\_-  
RestrictedBoundedConcurrentQueue,  
1145
    - std::deque, 2326
    - std::forward\_list, 2381
    - std::list, 2549
  - push\_heap
    - heap\_algorithms, 224
  - put
    - std::basic\_fstream, 1529
    - std::basic\_istream, 1678
    - std::basic\_ofstream, 1833
    - std::basic\_ostream, 1875
    - std::basic\_ostringstream, 1920
    - std::basic\_stringstream, 2090
    - std::money\_put, 2636, 2637
    - std::num\_put, 2745–2751
    - std::time\_put, 2955, 2956
    - std::time\_put\_byname, 2960
  - put\_money
    - std, 628
  - putback
    - std::basic\_fstream, 1530
    - std::basic\_ifstream, 1585
    - std::basic\_istream, 1678
    - std::basic\_istream, 1731
    - std::basic\_istreamstream, 1786
    - std::basic\_stringstream, 2090
  - pword
    - std::basic\_fstream, 1530
    - std::basic\_ifstream, 1586
    - std::basic\_ios, 1621
    - std::basic\_istream, 1679
    - std::basic\_istream, 1732
    - std::basic\_istreamstream, 1787
    - std::basic\_ofstream, 1834
    - std::basic\_ostream, 1876
    - std::basic\_ostringstream, 1920
-

- std::basic\_stringstream, 2091
  - std::ios\_base, 2486
- qsb\_steals
  - \_\_gnu\_parallel::\_Settings, 1154
- queue, 3339
  - std::queue, 2846
- queue.h, 3339
  - \_GLIBCXX\_VOLATILE, 3340
- quicksort.h, 3340
- quiet\_NaN
  - std::numeric\_limits, 2755
- radix
  - std::\_\_numeric\_limits\_base, 1342
  - std::numeric\_limits, 2759
- random, 3341
  - generate\_canonical, 209
- Random Number Distributions, 243
- Random Number Generation, 208
- Random Number Generators, 236
- Random Number Utilities, 264
- random.h, 3341
- random.tc, 3348
- random\_distributions\_bernoulli
  - operator<<, 254
  - operator>>, 255, 256
  - operator==, 255
- random\_distributions\_normal
  - operator<<, 250
  - operator>>, 251
  - operator==, 251
- random\_distributions\_poisson
  - operator<<, 260, 261
  - operator>>, 263, 264
  - operator==, 261, 262
- random\_distributions\_uniform
  - operator<<, 245, 246
  - operator>>, 247
  - operator==, 246
- random\_generators
  - minstd\_rand, 239
  - minstd\_rand0, 239
  - mt19937, 239
  - mt19937\_64, 239
  - operator<<, 242
- random\_number.h, 3353
- random\_sample
  - SGIextensions, 19, 20
- random\_sample\_n
  - SGIextensions, 20
- random\_shuffle
  - mutating\_algorithms, 132, 133
- random\_shuffle.h, 3354
- random\_shuffle\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- ratio, 3355
- Rational Arithmetic, 69
- rb\_tree, 3357
- rbegin
  - \_\_gnu\_cxx::\_\_versa\_string, 779
  - \_\_gnu\_debug::basic\_string, 1021, 1022
- std::basic\_string, 2005, 2006
- std::deque, 2326, 2327
- std::list, 2549
- std::map, 2599
- std::multimap, 2679
- std::multiset, 2698
- std::set, 2891
- std::vector, 3057
- rc\_string\_base.h, 3357
- rdbuf
  - std::basic\_fstream, 1531
  - std::basic\_ifstream, 1586
  - std::basic\_ios, 1621, 1622
  - std::basic\_iostream, 1679, 1680
  - std::basic\_istream, 1732, 1733
  - std::basic\_istreamstream, 1787
  - std::basic\_ofstream, 1834
  - std::basic\_ostream, 1876, 1877
  - std::basic\_ostreamstream, 1921, 1922
  - std::basic\_stringstream, 2091, 2092
- rdstate
  - std::basic\_fstream, 1532
  - std::basic\_ifstream, 1587
  - std::basic\_ios, 1622
  - std::basic\_iostream, 1680
  - std::basic\_istream, 1733
  - std::basic\_istreamstream, 1788
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1877

- std::basic\_ostringstream, 1922
- std::basic\_stringstream, 2092
- read
  - std::basic\_fstream, 1532
  - std::basic\_ifstream, 1587
  - std::basic\_iostream, 1681
  - std::basic\_istream, 1734
  - std::basic\_istreamstream, 1788
  - std::basic\_stringstream, 2093
- readsome
  - std::basic\_fstream, 1533
  - std::basic\_ifstream, 1588
  - std::basic\_iostream, 1681
  - std::basic\_istream, 1734
  - std::basic\_istreamstream, 1789
  - std::basic\_stringstream, 2093
- ref
  - std, 628, 629
- reference
  - std::back\_insert\_iterator, 1444
  - std::front\_insert\_iterator, 2391
  - std::insert\_iterator, 2474
  - std::istream\_iterator, 2509
  - std::istreambuf\_iterator, 2512
  - std::iterator, 2516
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2805
  - std::raw\_storage\_iterator, 2859
  - std::reverse\_iterator, 2866
  - std::set, 2878
- regex, 3358
- register\_callback
  - std::basic\_fstream, 1533
  - std::basic\_ifstream, 1589
  - std::basic\_ios, 1622
  - std::basic\_iostream, 1682
  - std::basic\_istream, 1735
  - std::basic\_istreamstream, 1790
  - std::basic\_ofstream, 1835
  - std::basic\_ostream, 1878
  - std::basic\_ostringstream, 1922
  - std::basic\_stringstream, 2094
  - std::ios\_base, 2486
- release
  - std::auto\_ptr, 1440
- remove
  - mutating\_algorithms, 133
  - std::forward\_list, 2381
  - std::list, 2549
- remove\_copy
  - mutating\_algorithms, 134
- remove\_copy\_if
  - mutating\_algorithms, 134
- remove\_if
  - mutating\_algorithms, 135
  - std::forward\_list, 2382
  - std::list, 2550
- rend
  - \_\_gnu\_cxx::\_\_versa\_string, 779
  - \_\_gnu\_debug::basic\_string, 1022
  - std::basic\_string, 2006
  - std::deque, 2327
  - std::list, 2550
  - std::map, 2599, 2600
  - std::multimap, 2679, 2680
  - std::multiset, 2698
  - std::set, 2891
  - std::vector, 3057, 3058
- replace
  - \_\_gnu\_cxx::\_\_versa\_string, 780–786
  - \_\_gnu\_debug::basic\_string, 1022–1028
  - mutating\_algorithms, 136
  - std::basic\_string, 2006–2013
- replace\_copy
  - std, 629
- replace\_copy\_if
  - mutating\_algorithms, 136
- replace\_if
  - mutating\_algorithms, 137
- replace\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- requested\_size
  - \_\_gnu\_cxx::temporary\_buffer, 946
  - std::\_Temporary\_buffer, 1408
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string, 787
  - \_\_gnu\_debug::basic\_string, 1029
  - std::basic\_string, 2013
  - std::vector, 3058
- reset

- std::auto\_ptr, 1441
- std::bernoulli\_distribution, 2113
- std::binomial\_distribution, 2124
- std::bitset, 2138
- std::cauchy\_distribution, 2143
- std::chi\_squared\_distribution, 2152
- std::discrete\_distribution, 2338
- std::exponential\_distribution, 2353
- std::extreme\_value\_distribution, 2357
- std::fisher\_f\_distribution, 2361
- std::gamma\_distribution, 2413
- std::geometric\_distribution, 2418
- std::lognormal\_distribution, 2579
- std::negative\_binomial\_distribution, 2706
- std::normal\_distribution, 2713
- std::piecewise\_constant\_distribution, 2819
- std::piecewise\_linear\_distribution, 2824
- std::poisson\_distribution, 2835
- std::student\_t\_distribution, 2923
- std::uniform\_int\_distribution, 3014
- std::uniform\_real\_distribution, 3018
- std::weibull\_distribution, 3067
- resetiosflags
  - std, 629
- resize
  - \_\_gnu\_cxx::\_\_versa\_string, 788
  - \_\_gnu\_debug::basic\_string, 1029
  - numeric\_arrays, 102
  - std::basic\_string, 2014
  - std::deque, 2327
  - std::forward\_list, 2382
  - std::list, 2551
  - std::vector, 3058
- result\_type
  - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder, 716
  - \_\_gnu\_cxx::binary\_compose, 807
  - \_\_gnu\_cxx::project1st, 864
  - \_\_gnu\_cxx::project2nd, 865
  - \_\_gnu\_cxx::select1st, 880
  - \_\_gnu\_cxx::select2nd, 881
  - \_\_gnu\_cxx::subtractive\_rng, 943
  - \_\_gnu\_cxx::unary\_compose, 957
  - \_\_gnu\_parallel::\_EqualFromLess, 1086
  - \_\_gnu\_parallel::\_EqualTo, 1088
  - \_\_gnu\_parallel::\_Less, 1098
  - \_\_gnu\_parallel::\_Lexicographic, 1100
  - \_\_gnu\_parallel::\_-
    - LexicographicReverse, 1102
  - \_\_gnu\_parallel::\_Multiplies, 1130
  - \_\_gnu\_parallel::\_Plus, 1134
  - \_\_gnu\_parallel::\_binder1st, 1041
  - \_\_gnu\_parallel::\_binder2nd, 1043
  - \_\_gnu\_parallel::\_unary\_negate, 1079
- std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >, 1393
- std::\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >, 1394
- std::bernoulli\_distribution, 2111
- std::binary\_function, 2116
- std::binary\_negate, 2118
- std::binder1st, 2119
- std::binder2nd, 2121
- std::binomial\_distribution, 2123
- std::cauchy\_distribution, 2142
- std::chi\_squared\_distribution, 2151
- std::const\_mem\_fun1\_ref\_t, 2206
- std::const\_mem\_fun1\_t, 2208
- std::const\_mem\_fun\_ref\_t, 2210
- std::const\_mem\_fun\_t, 2212
- std::discard\_block\_engine, 2330
- std::discrete\_distribution, 2337
- std::divides, 2341
- std::equal\_to, 2346
- std::exponential\_distribution, 2351
- std::extreme\_value\_distribution, 2355
- std::fisher\_f\_distribution, 2359
- std::gamma\_distribution, 2411
- std::geometric\_distribution, 2416
- std::greater, 2420
- std::greater\_equal, 2422
- std::hash, 2430

- std::hash< \_\_debug::bitset< \_Nb >  
>, 2432
- std::hash< \_\_debug::vector< bool,  
\_Alloc > >, 2434
- std::hash< \_\_gnu\_cxx::throw\_  
value\_limit >, 2436
- std::hash< \_\_gnu\_cxx::throw\_  
value\_random >, 2438
- std::hash< \_\_profile::bitset< \_Nb >  
>, 2440
- std::hash< \_\_profile::vector< bool,  
\_Alloc > >, 2442
- std::hash< \_Tp \* >, 2444
- std::hash< error\_code >, 2446
- std::hash< string >, 2448
- std::hash< thread::id >, 2450
- std::hash< u16string >, 2452
- std::hash< u32string >, 2454
- std::hash< wstring >, 2456
- std::hash<::bitset< \_Nb > >, 2458
- std::hash<::vector< bool, \_Alloc >  
>, 2460
- std::independent\_bits\_engine, 2462
- std::less, 2522
- std::less\_equal, 2524
- std::linear\_congruential\_engine,  
2526
- std::logical\_and, 2572
- std::logical\_not, 2574
- std::logical\_or, 2576
- std::lognormal\_distribution, 2578
- std::mem\_fun1\_ref\_t, 2606
- std::mem\_fun1\_t, 2608
- std::mem\_fun\_ref\_t, 2610
- std::mem\_fun\_t, 2612
- std::minus, 2622
- std::modulus, 2624
- std::multiplies, 2683
- std::negate, 2703
- std::negative\_binomial\_distribution,  
2705
- std::normal\_distribution, 2711
- std::not\_equal\_to, 2717
- std::piecewise\_constant\_  
distribution, 2817
- std::piecewise\_linear\_distribution,  
2822
- std::plus, 2828
- std::pointer\_to\_binary\_function,  
2830
- std::pointer\_to\_unary\_function,  
2832
- std::poisson\_distribution, 2833
- std::random\_device, 2850
- std::seed\_seq, 2873
- std::shuffle\_order\_engine, 2908
- std::student\_t\_distribution, 2921
- std::unary\_function, 3008
- std::unary\_negate, 3010
- std::uniform\_int\_distribution, 3013
- std::uniform\_real\_distribution, 3017
- std::weibull\_distribution, 3066
- rethrow\_exception  
exceptions, 32
- rethrow\_if\_nested  
exceptions, 32
- return\_temporary\_buffer  
std, 630
- reverse  
mutating\_algorithms, 137  
std::forward\_list, 2383  
std::list, 2551
- reverse\_copy  
mutating\_algorithms, 138
- reverse\_iterator  
std::reverse\_iterator, 2866, 2867  
std::set, 2878
- rfind  
\_\_gnu\_cxx::\_\_versa\_string, 788–  
790  
\_\_gnu\_debug::basic\_string, 1030,  
1031  
std::basic\_string, 2015, 2016
- riemann\_zeta  
tr1\_math\_spec\_func, 110
- right  
std, 630  
std::basic\_fstream, 1548  
std::basic\_ifstream, 1601  
std::basic\_ios, 1632  
std::basic\_iostream, 1697

- std::basic\_istream, 1748
- std::basic\_istringstream, 1803
- std::basic\_ofstream, 1847
- std::basic\_ostream, 1889
- std::basic\_ostringstream, 1934
- std::basic\_stringstream, 2109
- std::ios\_base, 2494
- rope, 3358
- ropeimpl.h, 3362
- rotate
  - mutating\_algorithms, 138
- rotate\_copy
  - mutating\_algorithms, 139
- round\_indeterminate
  - std, 555
- round\_to\_nearest
  - std, 555
- round\_toward\_infinity
  - std, 555
- round\_toward\_neg\_infinity
  - std, 555
- round\_toward\_zero
  - std, 555
- round\_error
  - std::numeric\_limits, 2755
- round\_style
  - std::\_\_numeric\_limits\_base, 1342
  - std::numeric\_limits, 2759
- runtime\_error
  - std::runtime\_error, 2872
- safe\_base.h, 3363
- safe\_iterator.h, 3363
- safe\_iterator.tcc, 3365
- safe\_sequence.h, 3365
- sbumpc
  - \_\_gnu\_cxx::enc\_filebuf, 831
  - \_\_gnu\_cxx::stdio\_filebuf, 902
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 930
  - std::basic\_filebuf, 1467
  - std::basic\_streambuf, 1949
  - std::basic\_stringbuf, 2032
- scan\_is
  - std::\_\_ctype\_abstract\_base, 1263
  - std::ctype, 2224
  - std::ctype< char >, 2237
  - std::ctype< wchar\_t >, 2259
  - std::ctype\_byname, 2275
  - std::ctype\_byname< char >, 2287
- scan\_not
  - std::\_\_ctype\_abstract\_base, 1263
  - std::ctype, 2224
  - std::ctype< char >, 2238
  - std::ctype< wchar\_t >, 2259
  - std::ctype\_byname, 2275
  - std::ctype\_byname< char >, 2288
- scientific
  - std, 630
  - std::basic\_fstream, 1548
  - std::basic\_ifstream, 1601
  - std::basic\_ios, 1632
  - std::basic\_iostream, 1697
  - std::basic\_istream, 1748
  - std::basic\_istringstream, 1803
  - std::basic\_ofstream, 1847
  - std::basic\_ostream, 1889
  - std::basic\_ostringstream, 1934
  - std::basic\_stringstream, 2109
  - std::ios\_base, 2494
- search
  - non\_mutating\_algorithms, 156, 157
- search.h, 3366
- search\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- search\_n
  - non\_mutating\_algorithms, 158
- second
  - \_\_gnu\_parallel::\_IteratorPair, 1093
  - std::pair, 2815
- second\_argument\_type
  - \_\_gnu\_cxx::project1st, 864
  - \_\_gnu\_cxx::project2nd, 865
  - \_\_gnu\_parallel::\_EqualFromLess, 1086
  - \_\_gnu\_parallel::\_EqualTo, 1088
  - \_\_gnu\_parallel::\_Less, 1098
  - \_\_gnu\_parallel::\_Lexicographic, 1100
  - \_\_gnu\_parallel::\_-LexicographicReverse, 1102
  - \_\_gnu\_parallel::\_Multiplies, 1130
  - \_\_gnu\_parallel::\_Plus, 1134

- std::\_Maybe\_unary\_or\_binary\_-  
function< \_Res, \_T1, \_T2 >,  
1394
- std::binary\_function, 2116
- std::binary\_negate, 2118
- std::const\_mem\_fun1\_ref\_t, 2206
- std::const\_mem\_fun1\_t, 2208
- std::divides, 2341
- std::equal\_to, 2346
- std::greater, 2420
- std::greater\_equal, 2422
- std::less, 2522
- std::less\_equal, 2524
- std::logical\_and, 2572
- std::logical\_or, 2576
- std::mem\_fun1\_ref\_t, 2606
- std::mem\_fun1\_t, 2608
- std::minus, 2622
- std::modulus, 2624
- std::multiplies, 2683
- std::not\_equal\_to, 2717
- std::plus, 2828
- std::pointer\_to\_binary\_function,  
2830
- second\_type
  - \_\_gnu\_parallel::\_IteratorPair, 1093
  - std::pair, 2814
- seconds
  - std::chrono, 683
- seed
  - std::discard\_block\_engine, 2333,  
2334
  - std::independent\_bits\_engine, 2465
  - std::linear\_congruential\_engine,  
2528
  - std::shuffle\_order\_engine, 2911
- seed\_seq
  - std::seed\_seq, 2873
- seekdir
  - std::basic\_fstream, 1496
  - std::basic\_ifstream, 1560
  - std::basic\_ios, 1611
  - std::basic\_iostream, 1646
  - std::basic\_istream, 1708
  - std::basic\_istreamstream, 1762
  - std::basic\_ofstream, 1814
  - std::basic\_ostream, 1858
  - std::basic\_ostringstream, 1902
  - std::basic\_stringstream, 2058
  - std::ios\_base, 2482
- seekg
  - std::basic\_fstream, 1534
  - std::basic\_ifstream, 1589, 1590
  - std::basic\_iostream, 1682, 1683
  - std::basic\_istream, 1735, 1736
  - std::basic\_istreamstream, 1790, 1791
  - std::basic\_stringstream, 2094, 2095
- seekoff
  - \_\_gnu\_cxx::enc\_filebuf, 831
  - \_\_gnu\_cxx::stdio\_filebuf, 903
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 931
  - std::basic\_filebuf, 1468
  - std::basic\_streambuf, 1950
  - std::basic\_stringbuf, 2032
- seekp
  - std::basic\_fstream, 1535
  - std::basic\_iostream, 1684
  - std::basic\_ofstream, 1836
  - std::basic\_ostream, 1878
  - std::basic\_ostringstream, 1922, 1923
  - std::basic\_stringstream, 2096
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf, 831
  - \_\_gnu\_cxx::stdio\_filebuf, 903
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 931
  - std::basic\_filebuf, 1468
  - std::basic\_streambuf, 1950
  - std::basic\_stringbuf, 2033
- sentry
  - std::basic\_istream::sentry, 1750
  - std::basic\_ostream::sentry, 1891
- Sequences, 23
- sequential
  - \_\_gnu\_parallel, 365
- set, 3366, 3367
  - \_\_gnu\_parallel::\_Settings, 1149
  - std::bitset, 2138, 2139
  - std::set, 2879–2881
- Set Operation, 181
- set.h, 3367, 3368
- set\_algorithms
  - includes, 183

- set\_difference, 184
- set\_intersection, 185, 186
- set\_symmetric\_difference, 186, 187
- set\_union, 187, 188
- set\_difference
  - set\_algorithms, 184
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- set\_intersection
  - set\_algorithms, 185, 186
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1154
- set\_new\_handler
  - std, 630
- set\_num\_threads
  - \_\_gnu\_parallel::balanced\_-quicksort\_tag, 1159
  - \_\_gnu\_parallel::balanced\_tag, 1161
  - \_\_gnu\_parallel::default\_parallel\_tag, 1163
  - \_\_gnu\_parallel::exact\_tag, 1166
  - \_\_gnu\_parallel::multiway\_-mergesort\_exact\_tag, 1169
  - \_\_gnu\_parallel::multiway\_-mergesort\_sampling\_tag, 1171
  - \_\_gnu\_parallel::multiway\_-mergesort\_tag, 1172
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, 1174
  - \_\_gnu\_parallel::omp\_loop\_tag, 1175
  - \_\_gnu\_parallel::parallel\_tag, 1178
  - \_\_gnu\_parallel::quicksort\_tag, 1180
  - \_\_gnu\_parallel::sampling\_tag, 1181
  - \_\_gnu\_parallel::unbalanced\_tag, 1183
- set\_operations.h, 3369
- set\_symmetric\_difference
  - set\_algorithms, 186, 187
- set\_symmetric\_difference\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1155
- set\_terminate
  - exceptions, 33
- set\_unexpected
  - exceptions, 33
- set\_union
  - set\_algorithms, 187, 188
- set\_union\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1155
- setbase
  - std, 630
- setbuf
  - \_\_gnu\_cxx::enc\_filebuf, 832
  - \_\_gnu\_cxx::stdio\_filebuf, 904
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 932
  - std::basic\_filebuf, 1468, 1469
  - std::basic\_streambuf, 1951
  - std::basic\_stringbuf, 2033
- setf
  - std::basic\_fstream, 1536
  - std::basic\_ifstream, 1590, 1591
  - std::basic\_ios, 1623
  - std::basic\_iostream, 1685
  - std::basic\_istream, 1737
  - std::basic\_istreamstream, 1791, 1792
  - std::basic\_ofstream, 1837
  - std::basic\_ostream, 1879
  - std::basic\_ostreamstream, 1923, 1924
  - std::basic\_stringstream, 2097
  - std::ios\_base, 2487
- setfill
  - std, 631
- setg
  - \_\_gnu\_cxx::enc\_filebuf, 832
  - \_\_gnu\_cxx::stdio\_filebuf, 904
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 932
  - std::basic\_filebuf, 1469
  - std::basic\_streambuf, 1951
  - std::basic\_stringbuf, 2034
- setiosflags
  - std, 631
- setp
  - \_\_gnu\_cxx::enc\_filebuf, 833
  - \_\_gnu\_cxx::stdio\_filebuf, 905
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 932
  - std::basic\_filebuf, 1470
  - std::basic\_streambuf, 1951
  - std::basic\_stringbuf, 2034
- setprecision
  - std, 631
- setstate

- std::basic\_fstream, 1537
- std::basic\_ifstream, 1591
- std::basic\_ios, 1624
- std::basic\_iostream, 1685
- std::basic\_istream, 1738
- std::basic\_istream, 1792
- std::basic\_ofstream, 1837
- std::basic\_ostream, 1880
- std::basic\_ostringstream, 1924
- std::basic\_stringstream, 2097
- settings.h, 3370
- \_GLIBCXX\_PARALLEL\_-  
  CONDITION, 3372
- setw
- std, 632
- sgetc
- \_\_gnu\_cxx::enc\_filebuf, 833
- \_\_gnu\_cxx::stdio\_filebuf, 905
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 933
- std::basic\_filebuf, 1470
- std::basic\_streambuf, 1952
- std::basic\_stringbuf, 2035
- sgetn
- \_\_gnu\_cxx::enc\_filebuf, 833
- \_\_gnu\_cxx::stdio\_filebuf, 906
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 933
- std::basic\_filebuf, 1470
- std::basic\_streambuf, 1952
- std::basic\_stringbuf, 2035
- SGI, 8
- SGIextensions
- \_Find\_first, 13
- \_Find\_next, 13
- \_Unchecked\_flip, 14
- \_Unchecked\_reset, 14
- \_Unchecked\_set, 14
- \_Unchecked\_test, 14
- \_\_median, 12, 13
- compose1, 15
- compose2, 15
- constant0, 15
- constant1, 15
- constant2, 15
- copy\_n, 16
- distance, 16
- identity\_element, 16, 17
- iota, 17
- is\_heap, 17
- is\_sorted, 18
- lexicographical\_compare\_3way, 18
- power, 19
- random\_sample, 19, 20
- random\_sample\_n, 20
- uninitialized\_copy\_n, 21
- shared\_future
- std::shared\_future, 2894, 2895
- std::shared\_future< \_Res & >, 2897
- std::shared\_future< void >, 2900
- shared\_ptr
- std::shared\_ptr, 2902–2905
- shared\_ptr.h, 3372
- shared\_ptr\_base.h, 3374
- shift
- numeric\_arrays, 102
- showbase
- std, 632
- std::basic\_fstream, 1548
- std::basic\_ifstream, 1601
- std::basic\_ios, 1632
- std::basic\_iostream, 1697
- std::basic\_istream, 1748
- std::basic\_istream, 1803
- std::basic\_ofstream, 1847
- std::basic\_ostream, 1889
- std::basic\_ostringstream, 1934
- std::basic\_stringstream, 2109
- std::ios\_base, 2494
- showmanyc
- \_\_gnu\_cxx::enc\_filebuf, 834
- \_\_gnu\_cxx::stdio\_filebuf, 906
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 933
- std::basic\_filebuf, 1471
- std::basic\_streambuf, 1952
- std::basic\_stringbuf, 2035
- showpoint
- std, 632
- std::basic\_fstream, 1548
- std::basic\_ifstream, 1601
- std::basic\_ios, 1632
- std::basic\_iostream, 1697
- std::basic\_istream, 1748
- std::basic\_istream, 1803

- std::basic\_ofstream, 1847
- std::basic\_ostream, 1889
- std::basic\_ostringstream, 1935
- std::basic\_stringstream, 2109
- std::ios\_base, 2494
- showpos
- std, 632
- std::basic\_fstream, 1548
- std::basic\_ifstream, 1602
- std::basic\_ios, 1632
- std::basic\_iostream, 1697
- std::basic\_istream, 1748
- std::basic\_istream, 1803
- std::basic\_ofstream, 1847
- std::basic\_ostream, 1890
- std::basic\_ostringstream, 1935
- std::basic\_stringstream, 2110
- std::ios\_base, 2494
- shrink\_to\_fit
- \_\_gnu\_cxx::\_\_versa\_string, 791
- \_\_gnu\_debug::basic\_string, 1031
- std::basic\_string, 2017
- std::deque, 2328
- std::vector, 3059
- shuffle
- mutating\_algorithms, 139
- shuffle\_order\_engine
- std::shuffle\_order\_engine, 2908, 2909
- signaling\_NaN
- std::numeric\_limits, 2755
- sin
- complex\_numbers, 48
- sinh
- complex\_numbers, 49
- size
- \_\_gnu\_cxx::\_\_versa\_string, 791
- \_\_gnu\_cxx::temporary\_buffer, 946
- \_\_gnu\_debug::basic\_string, 1032
- numeric\_arrays, 102, 103
- std::\_Temporary\_buffer, 1409
- std::basic\_string, 2017
- std::bitset, 2139
- std::deque, 2328
- std::list, 2551
- std::map, 2600
- std::multimap, 2680
- std::multiset, 2698
- std::priority\_queue, 2841
- std::queue, 2847
- std::set, 2891
- std::stack, 2919
- std::vector, 3059
- size\_type
- std::set, 2878
- skipws
- std, 632
- std::basic\_fstream, 1549
- std::basic\_ifstream, 1602
- std::basic\_ios, 1633
- std::basic\_iostream, 1697
- std::basic\_istream, 1748
- std::basic\_istream, 1803
- std::basic\_ofstream, 1848
- std::basic\_ostream, 1890
- std::basic\_ostringstream, 1935
- std::basic\_stringstream, 2110
- std::ios\_base, 2495
- sleep\_for
- std::this\_thread, 697
- sleep\_until
- std::this\_thread, 698
- slice
- numeric\_arrays, 84
- slice\_array
- numeric\_arrays, 84
- slice\_array.h, 3374
- slist, 3374
- snextc
- \_\_gnu\_cxx::enc\_filebuf, 834
- \_\_gnu\_cxx::stdio\_filebuf, 907
- \_\_gnu\_cxx::stdio\_sync\_filebuf, 934
- std::basic\_filebuf, 1471
- std::basic\_streambuf, 1953
- std::basic\_stringbuf, 2036
- sort
- sorting\_algorithms, 179
- std::forward\_list, 2383
- std::list, 2551, 2552
- sort.h, 3376
- sort\_heap
- heap\_algorithms, 225

- 
- sort\_minimal\_n
    - \_\_gnu\_parallel::\_Settings, 1155
  - sort\_mwms\_oversampling
    - \_\_gnu\_parallel::\_Settings, 1155
  - sort\_qs\_num\_samples\_preset
    - \_\_gnu\_parallel::\_Settings, 1155
  - sort\_qsb\_base\_case\_maximal\_n
    - \_\_gnu\_parallel::\_Settings, 1155
  - Sorting, 160
  - sorting\_algorithms
    - inplace\_merge, 163
    - is\_sorted, 164
    - is\_sorted\_until, 165
    - lexicographical\_compare, 166
    - max, 167
    - max\_element, 168
    - merge, 169
    - min, 170
    - min\_element, 171
    - minmax, 172
    - minmax\_element, 172, 173
    - next\_permutation, 173, 174
    - nth\_element, 174, 175
    - partial\_sort, 175, 176
    - partial\_sort\_copy, 177
    - prev\_permutation, 178
    - sort, 179
    - stable\_sort, 180, 181
  - sph\_bessel
    - tr1\_math\_spec\_func, 110
  - sph\_legendre
    - tr1\_math\_spec\_func, 110
  - sph\_neumann
    - tr1\_math\_spec\_func, 111
  - splice
    - std::list, 2552, 2553
  - splice\_after
    - std::forward\_list, 2383, 2384
  - sputback
    - \_\_gnu\_cxx::enc\_filebuf, 835
    - \_\_gnu\_cxx::stdio\_filebuf, 907
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 934
    - std::basic\_filebuf, 1472
    - std::basic\_streambuf, 1953
    - std::basic\_stringbuf, 2036
  - sputc
    - \_\_gnu\_cxx::enc\_filebuf, 835
    - \_\_gnu\_cxx::stdio\_filebuf, 907
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 935
    - std::basic\_filebuf, 1472
    - std::basic\_streambuf, 1954
    - std::basic\_stringbuf, 2037
  - sputn
    - \_\_gnu\_cxx::enc\_filebuf, 835
    - \_\_gnu\_cxx::stdio\_filebuf, 908
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 935
    - std::basic\_filebuf, 1473
    - std::basic\_streambuf, 1954
    - std::basic\_stringbuf, 2037
  - sqrt
    - complex\_numbers, 49
  - sso\_string\_base.h, 3377
  - sstream, 3377
  - sstream.tcc, 3378
  - stable\_partition
    - mutating\_algorithms, 140
  - stable\_sort
    - sorting\_algorithms, 180, 181
  - stack, 3378
    - std::stack, 2918
  - standard\_policies.hpp, 3379
  - start
    - numeric\_arrays, 103
  - state
    - std::fpos, 2388
  - std, 422
    - \_Construct, 568
    - \_Destroy, 569
    - \_final\_insertion\_sort, 555
    - \_find, 556
    - \_find\_if, 556
    - \_find\_if\_not, 556, 557
    - \_gcd, 557
    - \_heap\_select, 557
    - \_inplace\_stable\_partition, 558
    - \_inplace\_stable\_sort, 558
    - \_insertion\_sort, 558, 559
    - \_introsort\_loop, 559
    - \_invoke, 639
    - \_lg, 559
    - \_merge\_adaptive, 559, 560
    - \_merge\_backward, 560
-

- `__merge_without_buffer`, 561
- `__move_median_first`, 561, 562
- `__partition`, 562
- `__reverse`, 562, 563
- `__rotate`, 563
- `__rotate_adaptive`, 563
- `__search_n`, 564
- `__stable_partition_adaptive`, 565
- `__unguarded_insertion_sort`, 565
- `__unguarded_linear_insert`, 565, 566
- `__unguarded_partition`, 566
- `__unguarded_partition_pivot`, 566, 567
- `__unique_copy`, 567, 568
- `accumulate`, 569, 570
- `adjacent_difference`, 570
- `advance`, 571
- `bind`, 572
- `boolalpha`, 572
- `cerr`, 639
- `cin`, 639
- `clog`, 639
- `cout`, 639
- `cref`, 572
- `dec`, 572
- `denorm_absent`, 555
- `denorm_indeterminate`, 555
- `denorm_present`, 555
- `distance`, 573
- `endl`, 573
- `ends`, 573
- `fixed`, 574
- `float_denorm_style`, 554
- `float_round_style`, 555
- `flush`, 574
- `forward`, 574
- `get_money`, 575
- `get_temporary_buffer`, 575
- `getline`, 575–577
- `has_facet`, 578
- `hex`, 578
- `inner_product`, 578, 579
- `internal`, 580
- `iota`, 580
- `isalnum`, 580
- `isalpha`, 580
- `isctrl`, 580
- `isdigit`, 581
- `isgraph`, 581
- `islower`, 581
- `isprint`, 581
- `ispunct`, 581
- `isspace`, 581
- `isupper`, 581
- `isxdigit`, 581
- `left`, 582
- `new_handler`, 553
- `noboolalpha`, 582
- `noshowbase`, 582
- `noshowpoint`, 582
- `noshowpos`, 582
- `noskipws`, 582
- `nounitbuf`, 583
- `nouppercase`, 583
- `oct`, 583
- `operator<`, 589–595
- `operator<<`, 596–602
- `operator<=`, 603–606
- `operator>`, 613–616
- `operator>>`, 620–625
- `operator>=`, 616–619
- `operator^`, 626
- `operator+`, 587–589
- `operator==`, 606–613
- `operator&`, 587
- `partial_sum`, 627, 628
- `put_money`, 628
- `ref`, 628, 629
- `replace_copy`, 629
- `resetiosflags`, 629
- `return_temporary_buffer`, 630
- `right`, 630
- `round_indeterminate`, 555
- `round_to_nearest`, 555
- `round_toward_infinity`, 555
- `round_toward_neg_infinity`, 555
- `round_toward_zero`, 555
- `scientific`, 630
- `set_new_handler`, 630
- `setbase`, 630
- `setfill`, 631
- `setiosflags`, 631

- setprecision, 631
- setw, 632
- showbase, 632
- showpoint, 632
- showpos, 632
- skipws, 632
- streamoff, 553
- streampos, 554
- streamsize, 554
- swap, 633–635
- tolower, 635
- toupper, 636
- u16streampos, 554
- u32streampos, 554
- uninitialized\_copy, 636
- uninitialized\_copy\_n, 636
- uninitialized\_fill, 637
- uninitialized\_fill\_n, 637
- unitbuf, 637
- uppercase, 638
- use\_facet, 638
- wcerr, 640
- wcin, 640
- wclog, 640
- wcout, 640
- ws, 638
- wstreampos, 554
- std::\_\_basic\_future, 1245
  - \_M\_get\_result, 1246
- std::\_\_codecvt\_abstract\_base, 1246
  - do\_out, 1248
  - in, 1248
  - out, 1249
  - unshift, 1250
- std::\_\_ctype\_abstract\_base, 1251
  - char\_type, 1254
  - do\_is, 1255
  - do\_narrow, 1255, 1256
  - do\_scan\_is, 1257
  - do\_scan\_not, 1257
  - do\_tolower, 1258
  - do\_toupper, 1259
  - do\_widen, 1260
  - is, 1261
  - narrow, 1262
  - scan\_is, 1263
  - scan\_not, 1263
  - tolower, 1263, 1264
  - toupper, 1264, 1265
  - widen, 1265, 1266
- std::\_\_debug, 640
- std::\_\_debug::bitset, 1266
  - \_M\_const\_iterators, 1270
  - \_M\_detach\_all, 1269
  - \_M\_detach\_singular, 1269
  - \_M\_get\_mutex, 1269
  - \_M\_invalidate\_all, 1269
  - \_M\_iterators, 1270
  - \_M\_revalidate\_singular, 1269
  - \_M\_swap, 1270
  - \_M\_version, 1270
- std::\_\_debug::deque, 1271
  - \_M\_const\_iterators, 1275
  - \_M\_detach\_all, 1274
  - \_M\_detach\_singular, 1274
  - \_M\_get\_mutex, 1274
  - \_M\_invalidate\_all, 1274
  - \_M\_invalidate\_if, 1275
  - \_M\_iterators, 1276
  - \_M\_revalidate\_singular, 1275
  - \_M\_swap, 1275
  - \_M\_transfer\_iter, 1275
  - \_M\_version, 1276
- std::\_\_debug::list, 1276
  - \_M\_const\_iterators, 1281
  - \_M\_detach\_all, 1280
  - \_M\_detach\_singular, 1280
  - \_M\_get\_mutex, 1280
  - \_M\_invalidate\_all, 1280
  - \_M\_invalidate\_if, 1281
  - \_M\_iterators, 1282
  - \_M\_revalidate\_singular, 1281
  - \_M\_swap, 1281
  - \_M\_transfer\_iter, 1281
  - \_M\_version, 1282
- std::\_\_debug::map, 1282
  - \_M\_const\_iterators, 1287
  - \_M\_detach\_all, 1285
  - \_M\_detach\_singular, 1285
  - \_M\_get\_mutex, 1286
  - \_M\_invalidate\_all, 1286
  - \_M\_invalidate\_if, 1286

- [\\_M\\_iterators, 1287](#)
- [\\_M\\_revalidate\\_singular, 1286](#)
- [\\_M\\_swap, 1286](#)
- [\\_M\\_transfer\\_iter, 1287](#)
- [\\_M\\_version, 1287](#)
- std::[\\_\\_debug::multimap, 1288](#)
- [\\_M\\_const\\_iterators, 1292](#)
- [\\_M\\_detach\\_all, 1291](#)
- [\\_M\\_detach\\_singular, 1291](#)
- [\\_M\\_get\\_mutex, 1291](#)
- [\\_M\\_invalidate\\_all, 1291](#)
- [\\_M\\_invalidate\\_if, 1291](#)
- [\\_M\\_iterators, 1292](#)
- [\\_M\\_revalidate\\_singular, 1291](#)
- [\\_M\\_swap, 1292](#)
- [\\_M\\_transfer\\_iter, 1292](#)
- [\\_M\\_version, 1292](#)
- std::[\\_\\_debug::multiset, 1293](#)
- [\\_M\\_const\\_iterators, 1297](#)
- [\\_M\\_detach\\_all, 1296](#)
- [\\_M\\_detach\\_singular, 1296](#)
- [\\_M\\_get\\_mutex, 1296](#)
- [\\_M\\_invalidate\\_all, 1296](#)
- [\\_M\\_invalidate\\_if, 1296](#)
- [\\_M\\_iterators, 1297](#)
- [\\_M\\_revalidate\\_singular, 1297](#)
- [\\_M\\_swap, 1297](#)
- [\\_M\\_transfer\\_iter, 1297](#)
- [\\_M\\_version, 1298](#)
- std::[\\_\\_debug::set, 1298](#)
- [\\_M\\_const\\_iterators, 1303](#)
- [\\_M\\_detach\\_all, 1301](#)
- [\\_M\\_detach\\_singular, 1301](#)
- [\\_M\\_get\\_mutex, 1302](#)
- [\\_M\\_invalidate\\_all, 1302](#)
- [\\_M\\_invalidate\\_if, 1302](#)
- [\\_M\\_iterators, 1303](#)
- [\\_M\\_revalidate\\_singular, 1302](#)
- [\\_M\\_swap, 1302](#)
- [\\_M\\_transfer\\_iter, 1303](#)
- [\\_M\\_version, 1303](#)
- std::[\\_\\_debug::unordered\\_map, 1304](#)
- [\\_M\\_const\\_iterators, 1308](#)
- [\\_M\\_detach\\_all, 1306](#)
- [\\_M\\_detach\\_singular, 1306](#)
- [\\_M\\_get\\_mutex, 1306](#)
- [\\_M\\_invalidate\\_all, 1307](#)
- [\\_M\\_invalidate\\_if, 1307](#)
- [\\_M\\_iterators, 1308](#)
- [\\_M\\_revalidate\\_singular, 1307](#)
- [\\_M\\_swap, 1307](#)
- [\\_M\\_transfer\\_iter, 1307](#)
- [\\_M\\_version, 1308](#)
- std::[\\_\\_debug::unordered\\_multimap, 1309](#)
- [\\_M\\_const\\_iterators, 1312](#)
- [\\_M\\_detach\\_all, 1311](#)
- [\\_M\\_detach\\_singular, 1311](#)
- [\\_M\\_get\\_mutex, 1311](#)
- [\\_M\\_invalidate\\_all, 1311](#)
- [\\_M\\_invalidate\\_if, 1311](#)
- [\\_M\\_iterators, 1312](#)
- [\\_M\\_revalidate\\_singular, 1312](#)
- [\\_M\\_swap, 1312](#)
- [\\_M\\_transfer\\_iter, 1312](#)
- [\\_M\\_version, 1313](#)
- std::[\\_\\_debug::unordered\\_multiset, 1313](#)
- [\\_M\\_const\\_iterators, 1317](#)
- [\\_M\\_detach\\_all, 1315](#)
- [\\_M\\_detach\\_singular, 1315](#)
- [\\_M\\_get\\_mutex, 1316](#)
- [\\_M\\_invalidate\\_all, 1316](#)
- [\\_M\\_invalidate\\_if, 1316](#)
- [\\_M\\_iterators, 1317](#)
- [\\_M\\_revalidate\\_singular, 1316](#)
- [\\_M\\_swap, 1316](#)
- [\\_M\\_transfer\\_iter, 1316](#)
- [\\_M\\_version, 1317](#)
- std::[\\_\\_debug::unordered\\_set, 1318](#)
- [\\_M\\_const\\_iterators, 1322](#)
- [\\_M\\_detach\\_all, 1320](#)
- [\\_M\\_detach\\_singular, 1320](#)
- [\\_M\\_get\\_mutex, 1320](#)
- [\\_M\\_invalidate\\_all, 1321](#)
- [\\_M\\_invalidate\\_if, 1321](#)
- [\\_M\\_iterators, 1322](#)
- [\\_M\\_revalidate\\_singular, 1321](#)
- [\\_M\\_swap, 1321](#)
- [\\_M\\_transfer\\_iter, 1321](#)
- [\\_M\\_version, 1322](#)
- std::[\\_\\_debug::vector, 1323](#)
- [\\_M\\_const\\_iterators, 1327](#)
- [\\_M\\_detach\\_all, 1326](#)

- [\\_M\\_detach\\_singular](#), 1326
- [\\_M\\_get\\_mutex](#), 1326
- [\\_M\\_invalidate\\_all](#), 1326
- [\\_M\\_invalidate\\_if](#), 1327
- [\\_M\\_iterators](#), 1328
- [\\_M\\_revalidate\\_singular](#), 1327
- [\\_M\\_swap](#), 1327
- [\\_M\\_transfer\\_iter](#), 1327
- [\\_M\\_version](#), 1328
- [vector](#), 1326
- [std::\\_\\_declval\\_protector](#), 1328
- [std::\\_\\_detail](#), 646
- [std::\\_\\_exception\\_ptr::exception\\_ptr](#), 1329
- [std::\\_\\_future\\_base](#), 1330
- [std::\\_\\_future\\_base::\\_Ptr](#), 1331
- [std::\\_\\_future\\_base::\\_Result](#), 1332
- [std::\\_\\_future\\_base::\\_Result< \\_Res & >](#), 1333
- [std::\\_\\_future\\_base::\\_Result< void >](#), 1334
- [std::\\_\\_future\\_base::\\_Result\\_base](#), 1335
- [std::\\_\\_future\\_base::\\_State](#), 1336
- [std::\\_\\_is\\_location\\_invariant](#), 1337
- [std::\\_\\_numeric\\_limits\\_base](#), 1337
  - [digits](#), 1339
  - [digits10](#), 1339
  - [has\\_denorm](#), 1339
  - [has\\_denorm\\_loss](#), 1339
  - [has\\_infinity](#), 1339
  - [has\\_quiet\\_NaN](#), 1340
  - [has\\_signaling\\_NaN](#), 1340
  - [is\\_bounded](#), 1340
  - [is\\_exact](#), 1340
  - [is\\_iec559](#), 1340
  - [is\\_integer](#), 1340
  - [is\\_modulo](#), 1341
  - [is\\_signed](#), 1341
  - [is\\_specialized](#), 1341
  - [max\\_digits10](#), 1341
  - [max\\_exponent](#), 1341
  - [max\\_exponent10](#), 1341
  - [min\\_exponent](#), 1341
  - [min\\_exponent10](#), 1342
  - [radix](#), 1342
  - [round\\_style](#), 1342
  - [tinyness\\_before](#), 1342
  - [traps](#), 1342
- [std::\\_\\_parallel](#), 647
- [std::\\_\\_parallel::\\_CRandNumber](#), 1343
- [std::\\_\\_profile](#), 672
- [std::\\_\\_profile::bitset](#), 1343
- [std::\\_\\_profile::deque](#), 1345
- [std::\\_\\_profile::list](#), 1347
- [std::\\_\\_profile::map](#), 1350
- [std::\\_\\_profile::multimap](#), 1352
- [std::\\_\\_profile::multiset](#), 1354
- [std::\\_\\_profile::set](#), 1356
- [std::\\_\\_profile::unordered\\_map](#), 1358
- [std::\\_\\_profile::unordered\\_multimap](#), 1360
- [std::\\_\\_profile::unordered\\_multiset](#), 1361
- [std::\\_\\_profile::unordered\\_set](#), 1363
- [std::\\_Base\\_bitset](#), 1364
  - [\\_M\\_w](#), 1366
  - [std::\\_Base\\_bitset< 0 >](#), 1366
  - [std::\\_Base\\_bitset< 1 >](#), 1367
- [std::\\_Build\\_index\\_tuple](#), 1369
- [std::\\_Deque\\_base](#), 1369
  - [\\_M\\_initialize\\_map](#), 1371
- [std::\\_Deque\\_iterator](#), 1371
  - [\\_M\\_set\\_node](#), 1373
- [std::\\_Derives\\_from\\_binary\\_function](#), 1374
- [std::\\_Derives\\_from\\_unary\\_function](#), 1374
- [std::\\_Function\\_base](#), 1375
- [std::\\_Function\\_to\\_function\\_pointer](#), 1376
- [std::\\_Fwd\\_list\\_base](#), 1376
- [std::\\_Fwd\\_list\\_const\\_iterator](#), 1378
- [std::\\_Fwd\\_list\\_iterator](#), 1379
- [std::\\_Fwd\\_list\\_node](#), 1380
- [std::\\_Fwd\\_list\\_node\\_base](#), 1382
- [std::\\_Has\\_result\\_type\\_helper](#), 1383
- [std::\\_Index\\_tuple](#), 1383
- [std::\\_List\\_base](#), 1384
- [std::\\_List\\_const\\_iterator](#), 1385
- [std::\\_List\\_iterator](#), 1387
- [std::\\_List\\_node](#), 1388
  - [\\_M\\_data](#), 1389
- [std::\\_List\\_node\\_base](#), 1389
- [std::\\_Maybe\\_get\\_result\\_type](#), 1390
- [std::\\_Maybe\\_unary\\_or\\_binary\\_function](#), 1391

- std::\_Maybe\_unary\_or\_binary\_-  
   function< \_Res, \_T1 >, 1391  
   argument\_type, 1392  
   result\_type, 1393  
 std::\_Maybe\_unary\_or\_binary\_-  
   function< \_Res, \_T1, \_T2  
   >, 1393  
   first\_argument\_type, 1394  
   result\_type, 1394  
   second\_argument\_type, 1394  
 std::\_Maybe\_wrap\_member\_pointer,  
   1395  
 std::\_Maybe\_wrap\_member\_pointer< \_  
   Tp \_Class::\* >, 1395  
 std::\_Mem\_fn< \_Res(\_Class::\*)(\_  
   ArgTypes...) const >, 1396  
 std::\_Mem\_fn< \_Res(\_Class::\*)(\_  
   ArgTypes...) const volatile >,  
   1397  
 std::\_Mem\_fn< \_Res(\_Class::\*)(\_  
   ArgTypes...) volatile >, 1398  
 std::\_Mem\_fn< \_Res(\_Class::\*)(\_  
   ArgTypes...) >, 1399  
 std::\_Mu< \_Arg, false, false >, 1401  
 std::\_Mu< \_Arg, false, true >, 1401  
 std::\_Mu< \_Arg, true, false >, 1402  
 std::\_Mu< reference\_wrapper< \_Tp >,  
   false, false >, 1402  
 std::\_Placeholder, 1403  
 std::\_Reference\_wrapper\_base, 1404  
 std::\_Safe\_tuple\_element, 1405  
 std::\_Safe\_tuple\_element\_impl, 1405  
 std::\_Safe\_tuple\_element\_impl< \_\_i, \_  
   Tuple, false >, 1406  
 std::\_Temporary\_buffer, 1407  
   \_Temporary\_buffer, 1408  
   begin, 1408  
   end, 1408  
   requested\_size, 1408  
   size, 1409  
 std::\_Tuple\_impl< \_Idx >, 1409  
 std::\_Tuple\_impl< \_Idx, \_Head, \_  
   Tail...>, 1410  
 std::\_Vector\_base, 1411  
 std::\_Weak\_result\_type, 1412  
 std::\_Weak\_result\_type\_impl, 1413  
 std::\_Weak\_result\_type\_impl< \_  
   Res(\*)(\_ArgTypes...) >, 1414  
 std::\_Weak\_result\_type\_impl< \_  
   Res(&)(\_ArgTypes...) >, 1413  
 std::\_Weak\_result\_type\_impl< \_Res(\_  
   ArgTypes...) >, 1414  
 std::\_Weak\_result\_type\_impl< \_Res(\_  
   Class::\*)(\_ArgTypes...) const  
   >, 1415  
 std::\_Weak\_result\_type\_impl< \_Res(\_  
   Class::\*)(\_ArgTypes...) const  
   volatile >, 1415  
 std::\_Weak\_result\_type\_impl< \_Res(\_  
   Class::\*)(\_ArgTypes...) volatile  
   >, 1416  
 std::\_Weak\_result\_type\_impl< \_Res(\_  
   Class::\*)(\_ArgTypes...) >,  
   1416  
 std::add\_lvalue\_reference, 1417  
 std::add\_rvalue\_reference, 1418  
 std::adopt\_lock\_t, 1418  
 std::aligned\_storage, 1418  
 std::allocator, 1419  
 std::allocator< void >, 1420  
 std::atomic, 1421  
 std::atomic< \_Tp \* >, 1422  
 std::atomic< bool >, 1423  
 std::atomic< char >, 1424  
 std::atomic< char16\_t >, 1424  
 std::atomic< char32\_t >, 1425  
 std::atomic< int >, 1426  
 std::atomic< long >, 1427  
 std::atomic< long long >, 1428  
 std::atomic< short >, 1428  
 std::atomic< signed char >, 1429  
 std::atomic< unsigned char >, 1430  
 std::atomic< unsigned int >, 1431  
 std::atomic< unsigned long >, 1432  
 std::atomic< unsigned long long >,  
   1433  
 std::atomic< unsigned short >, 1433  
 std::atomic< void \* >, 1434  
 std::atomic< wchar\_t >, 1435  
 std::auto\_ptr, 1436  
   ~auto\_ptr, 1438  
   auto\_ptr, 1437, 1438

- element\_type, 1437
- get, 1439
- operator\*, 1439
- operator->, 1439
- operator=, 1440
- release, 1440
- reset, 1441
- std::auto\_ptr\_ref, 1441
- std::back\_insert\_iterator, 1442
  - back\_insert\_iterator, 1444
  - container\_type, 1443
  - difference\_type, 1443
  - iterator\_category, 1444
  - operator\*, 1445
  - operator++, 1445
  - operator=, 1445
  - pointer, 1444
  - reference, 1444
  - value\_type, 1444
- std::bad\_alloc, 1446
  - what, 1446
- std::bad\_cast, 1447
  - what, 1448
- std::bad\_exception, 1448
  - what, 1449
- std::bad\_function\_call, 1449
  - what, 1450
- std::bad\_typeid, 1450
  - what, 1451
- std::basic\_filebuf, 1451
  - ~basic\_filebuf, 1457
  - \_M\_buf, 1477
  - \_M\_buf\_locale, 1477
  - \_M\_buf\_size, 1477
  - \_M\_create\_pback, 1457
  - \_M\_destroy\_pback, 1457
  - \_M\_ext\_buf, 1477
  - \_M\_ext\_buf\_size, 1478
  - \_M\_ext\_next, 1478
  - \_M\_in\_beg, 1478
  - \_M\_in\_cur, 1478
  - \_M\_in\_end, 1479
  - \_M\_mode, 1479
  - \_M\_out\_beg, 1479
  - \_M\_out\_cur, 1480
  - \_M\_out\_end, 1480
  - \_M\_pback, 1481
  - \_M\_pback\_cur\_save, 1481
  - \_M\_pback\_end\_save, 1481
  - \_M\_pback\_init, 1481
  - \_M\_reading, 1482
  - \_M\_set\_buffer, 1458
  - \_\_streambuf\_type, 1455
- basic\_filebuf, 1457
- char\_type, 1455
- close, 1458
- eback, 1458
- egptr, 1459
- epptr, 1459
- gbump, 1460
- getloc, 1460
- gptr, 1460
- imbue, 1461
- in\_avail, 1461
- int\_type, 1456
- is\_open, 1462
- off\_type, 1456
- open, 1462, 1463
- overflow, 1463
- pbackfail, 1464
- pbase, 1464
- pbump, 1465
- pos\_type, 1456
- pptr, 1465
- pubimbue, 1465
- pubseekoff, 1466
- pubseekpos, 1466
- pubsetbuf, 1467
- pubsync, 1467
- sbumpc, 1467
- seekoff, 1468
- seekpos, 1468
- setbuf, 1468, 1469
- setg, 1469
- setp, 1470
- sgetc, 1470
- sgetn, 1470
- showmanyc, 1471
- snextc, 1471
- sputbackc, 1472
- sputc, 1472
- sputn, 1473

- stossc, 1473
- sungetc, 1473
- sync, 1474
- traits\_type, 1456
- uflow, 1474
- underflow, 1475
- xsgetn, 1475
- xspn, 1476
- std::basic\_fstream, 1482
  - ~basic\_fstream, 1498
  - \_M\_gcount, 1543
  - \_M\_getloc, 1499
  - \_M\_write, 1499
  - \_\_ctype\_type, 1491
  - \_\_num\_get\_type, 1491
  - \_\_num\_put\_type, 1491
- adjustfield, 1543
- app, 1543
- ate, 1543
- bad, 1499
- badbit, 1544
- basefield, 1544
- basic\_fstream, 1497, 1498
- beg, 1544
- binary, 1544
- boolalpha, 1545
- char\_type, 1492
- clear, 1500
- close, 1500
- copyfmt, 1500
- cur, 1545
- dec, 1545
- end, 1545
- eof, 1501
- eofbit, 1545
- event, 1497
- event\_callback, 1492
- exceptions, 1501
- fail, 1502
- failbit, 1546
- fill, 1503
- fixed, 1546
- flags, 1503, 1504
- floatfield, 1546
- flush, 1504
- fmtflags, 1493
- gcount, 1504
- get, 1505–1508
- getline, 1508
- getloc, 1509
- good, 1510
- goodbit, 1546
- hex, 1547
- ignore, 1510, 1511
- imbue, 1512
- in, 1547
- init, 1512
- int\_type, 1493, 1494
- internal, 1547
- iostate, 1494
- is\_open, 1512
- iword, 1513
- left, 1547
- narrow, 1513
- oct, 1547
- off\_type, 1494, 1495
- open, 1514
- openmode, 1495
- operator void \*, 1514
- operator<<, 1515–1521
- operator>>, 1522–1528
- out, 1548
- peek, 1528
- pos\_type, 1495, 1496
- precision, 1528, 1529
- put, 1529
- putback, 1530
- pword, 1530
- rdbuf, 1531
- rdstate, 1532
- read, 1532
- readsome, 1533
- register\_callback, 1533
- right, 1548
- scientific, 1548
- seekdir, 1496
- seekg, 1534
- seekp, 1535
- setf, 1536
- setstate, 1537
- showbase, 1548
- showpoint, 1548

- showpos, 1548
- skipws, 1549
- sync, 1537
- sync\_with\_stdio, 1538
- tellg, 1538
- tellp, 1539
- tie, 1539
- traits\_type, 1496, 1497
- trunc, 1549
- unget, 1540
- unitbuf, 1549
- unsetf, 1540
- uppercase, 1549
- widen, 1541
- width, 1541
- write, 1542
- xalloc, 1542
- std::basic\_ifstream, 1549
  - ~basic\_ifstream, 1561
  - \_M\_gcount, 1596
  - \_M\_getloc, 1562
  - \_\_ctype\_type, 1556
  - \_\_num\_get\_type, 1556
  - \_\_num\_put\_type, 1556
- adjustfield, 1596
- app, 1597
- ate, 1597
- bad, 1562
- badbit, 1597
- basefield, 1597
- basic\_ifstream, 1561
- beg, 1598
- binary, 1598
- boolalpha, 1598
- char\_type, 1557
- clear, 1562
- close, 1563
- copyfmt, 1563
- cur, 1598
- dec, 1598
- end, 1598
- eof, 1564
- eofbit, 1599
- event, 1560
- event\_callback, 1557
- exceptions, 1564
- fail, 1565
- failbit, 1599
- fill, 1565, 1566
- fixed, 1599
- flags, 1566
- floatfield, 1599
- fmtflags, 1557
- gcount, 1567
- get, 1567–1570
- getline, 1570, 1571
- getloc, 1572
- good, 1572
- goodbit, 1599
- hex, 1600
- ignore, 1572–1574
- imbue, 1574
- in, 1600
- init, 1575
- int\_type, 1558
- internal, 1600
- iostate, 1558
- is\_open, 1575
- isword, 1575
- left, 1600
- narrow, 1576
- oct, 1601
- off\_type, 1559
- open, 1576
- openmode, 1559
- operator void \*, 1577
- operator>>, 1577–1584
- out, 1601
- peek, 1584
- pos\_type, 1559
- precision, 1584, 1585
- putback, 1585
- pword, 1586
- rdbuf, 1586
- rdstate, 1587
- read, 1587
- readsome, 1588
- register\_callback, 1589
- right, 1601
- scientific, 1601
- seekdir, 1560
- seekg, 1589, 1590

- setf, 1590, 1591
- setstate, 1591
- showbase, 1601
- showpoint, 1601
- showpos, 1602
- skipws, 1602
- sync, 1592
- sync\_with\_stdio, 1592
- tellg, 1593
- tie, 1593
- traits\_type, 1560
- trunc, 1602
- unget, 1594
- unitbuf, 1602
- unsetf, 1594
- uppercase, 1602
- widen, 1595
- width, 1595
- xalloc, 1596
- std::basic\_ios, 1603
  - ~basic\_ios, 1612
  - \_M\_getloc, 1613
  - \_\_ctype\_type, 1607
  - \_\_num\_get\_type, 1607
  - \_\_num\_put\_type, 1607
- adjustfield, 1627
- app, 1627
- ate, 1627
- bad, 1613
- badbit, 1628
- basefield, 1628
- basic\_ios, 1612
- beg, 1628
- binary, 1628
- boolalpha, 1629
- char\_type, 1607
- clear, 1613
- copyfmt, 1613
- cur, 1629
- dec, 1629
- end, 1629
- eof, 1614
- eofbit, 1629
- event, 1612
- event\_callback, 1608
- exceptions, 1614, 1615
- fail, 1615
- failbit, 1630
- fill, 1616
- fixed, 1630
- flags, 1616, 1617
- floatfield, 1630
- fmtflags, 1608
- getloc, 1617
- good, 1617
- goodbit, 1630
- hex, 1631
- imbue, 1618
- in, 1631
- init, 1618
- int\_type, 1609
- internal, 1631
- iostate, 1609
- isword, 1618
- left, 1631
- narrow, 1619
- oct, 1631
- off\_type, 1610
- openmode, 1610
- operator void \*, 1619
- out, 1632
- pos\_type, 1610
- precision, 1620
- pword, 1621
- rdbuf, 1621, 1622
- rdstate, 1622
- register\_callback, 1622
- right, 1632
- scientific, 1632
- seekdir, 1611
- setf, 1623
- setstate, 1624
- showbase, 1632
- showpoint, 1632
- showpos, 1632
- skipws, 1633
- sync\_with\_stdio, 1624
- tie, 1625
- traits\_type, 1611
- trunc, 1633
- unitbuf, 1633
- unsetf, 1625

- uppercase, 1633
- widen, 1626
- width, 1626
- xalloc, 1627
- std::basic\_istream, 1633
  - ~basic\_istream, 1648
  - \_M\_gcount, 1692
  - \_M\_getloc, 1648
  - \_M\_write, 1648
  - \_\_ctype\_type, 1641
  - \_\_num\_get\_type, 1641
  - \_\_num\_put\_type, 1641, 1642
- adjustfield, 1692
- app, 1692
- ate, 1692
- bad, 1649
- badbit, 1692
- basefield, 1693
- basic\_istream, 1648
- beg, 1693
- binary, 1693
- boolalpha, 1693
- char\_type, 1642
- clear, 1649
- copyfmt, 1650
- cur, 1694
- dec, 1694
- end, 1694
- eof, 1650
- eofbit, 1694
- event, 1647
- event\_callback, 1643
- exceptions, 1650, 1651
- fail, 1651
- failbit, 1694
- fill, 1652
- fixed, 1695
- flags, 1653
- floatfield, 1695
- flush, 1653
- fmtflags, 1643
- gcount, 1654
- get, 1654–1657
- getline, 1657, 1658
- getloc, 1659
- good, 1659
- goodbit, 1695
- hex, 1695
- ignore, 1659–1661
- imbue, 1661
- in, 1696
- init, 1662
- int\_type, 1644
- internal, 1696
- iostate, 1644
- iword, 1662
- left, 1696
- narrow, 1662
- oct, 1696
- off\_type, 1645
- openmode, 1645
- operator void \*, 1663
- operator<<, 1663–1670
- operator>>, 1670–1676
- out, 1696
- peek, 1677
- pos\_type, 1646
- precision, 1677
- put, 1678
- putback, 1678
- pword, 1679
- rdbuf, 1679, 1680
- rdstate, 1680
- read, 1681
- readsome, 1681
- register\_callback, 1682
- right, 1697
- scientific, 1697
- seekdir, 1646
- seekg, 1682, 1683
- seekp, 1684
- setf, 1685
- setstate, 1685
- showbase, 1697
- showpoint, 1697
- showpos, 1697
- skipws, 1697
- sync, 1686
- sync\_with\_stdio, 1687
- tellg, 1687
- tellp, 1687
- tie, 1688

- traits\_type, 1647
- trunc, 1697
- unget, 1689
- unitbuf, 1698
- unsetf, 1689
- uppercase, 1698
- widen, 1689
- width, 1690
- write, 1691
- xalloc, 1691
- std::basic\_istream, 1698
  - ~basic\_istream, 1709
  - \_M\_gcount, 1743
  - \_M\_getloc, 1710
  - \_\_ctype\_type, 1704
  - \_\_num\_get\_type, 1704
  - \_\_num\_put\_type, 1705
- adjustfield, 1743
- app, 1743
- ate, 1743
- bad, 1710
- badbit, 1744
- basefield, 1744
- basic\_istream, 1709
- beg, 1744
- binary, 1744
- boolalpha, 1744
- char\_type, 1705
- clear, 1710
- copyfmt, 1711
- cur, 1745
- dec, 1745
- end, 1745
- eof, 1711
- eofbit, 1745
- event, 1709
- event\_callback, 1705
- exceptions, 1711, 1712
- fail, 1712
- failbit, 1745
- fill, 1713
- fixed, 1746
- flags, 1714
- floatfield, 1746
- fmtflags, 1706
- gcount, 1714
- get, 1715–1717
- getline, 1718, 1719
- getloc, 1719
- good, 1719
- goodbit, 1746
- hex, 1746
- ignore, 1720, 1721
- imbue, 1721
- in, 1747
- init, 1722
- int\_type, 1706
- internal, 1747
- iostate, 1707
- isword, 1722
- left, 1747
- narrow, 1722
- oct, 1747
- off\_type, 1707
- openmode, 1707
- operator void \*, 1723
- operator>>, 1723–1730
- out, 1747
- peek, 1730
- pos\_type, 1708
- precision, 1730, 1731
- putback, 1731
- pword, 1732
- rdbuf, 1732, 1733
- rdstate, 1733
- read, 1734
- readsome, 1734
- register\_callback, 1735
- right, 1748
- scientific, 1748
- seekdir, 1708
- seekg, 1735, 1736
- setf, 1737
- setstate, 1738
- showbase, 1748
- showpoint, 1748
- showpos, 1748
- skipws, 1748
- sync, 1738
- sync\_with\_stdio, 1739
- tellg, 1739
- tie, 1740

- traits\_type, 1708
- trunc, 1748
- unget, 1740
- unitbuf, 1749
- unsetf, 1741
- uppercase, 1749
- widen, 1741
- width, 1742
- xalloc, 1742
- std::basic\_istream::sentry, 1749
- operator bool, 1751
- sentry, 1750
- traits\_type, 1750
- std::basic\_istream, 1751
- ~basic\_istream, 1763
- \_M\_gcount, 1798
- \_M\_getloc, 1764
- \_\_ctype\_type, 1758
- \_\_num\_get\_type, 1758
- \_\_num\_put\_type, 1758
- adjustfield, 1798
- app, 1798
- ate, 1798
- bad, 1764
- badbit, 1799
- basefield, 1799
- basic\_istream, 1763
- beg, 1799
- binary, 1799
- boolalpha, 1800
- char\_type, 1759
- clear, 1764
- copyfmt, 1765
- cur, 1800
- dec, 1800
- end, 1800
- eof, 1765
- eofbit, 1800
- event, 1762
- event\_callback, 1759
- exceptions, 1766
- fail, 1767
- failbit, 1801
- fill, 1767
- fixed, 1801
- flags, 1768
- floatfield, 1801
- fmtflags, 1759
- gcount, 1769
- get, 1769–1772
- getline, 1772, 1773
- getloc, 1774
- good, 1774
- goodbit, 1801
- hex, 1802
- ignore, 1774, 1775
- imbue, 1776
- in, 1802
- init, 1776
- int\_type, 1760
- internal, 1802
- iostate, 1760
- isword, 1777
- left, 1802
- narrow, 1777
- oct, 1802
- off\_type, 1761
- openmode, 1761
- operator void \*, 1778
- operator>>, 1778–1784
- out, 1803
- peek, 1785
- pos\_type, 1761
- precision, 1785, 1786
- putback, 1786
- pwd, 1787
- rdbuf, 1787
- rdstate, 1788
- read, 1788
- readsome, 1789
- register\_callback, 1790
- right, 1803
- scientific, 1803
- seekdir, 1762
- seekg, 1790, 1791
- setf, 1791, 1792
- setstate, 1792
- showbase, 1803
- showpoint, 1803
- showpos, 1803
- skipws, 1803
- str, 1793

- sync, 1793
- sync\_with\_stdio, 1794
- tellg, 1794
- tie, 1795
- traits\_type, 1762
- trunc, 1804
- unget, 1795
- unitbuf, 1804
- unsetf, 1796
- uppercase, 1804
- widen, 1796
- width, 1797
- xalloc, 1797
- std::basic\_ofstream, 1804
- ~basic\_ofstream, 1816
- \_M\_getloc, 1816
- \_M\_write, 1817
- \_\_ctype\_type, 1811
- \_\_num\_get\_type, 1811
- \_\_num\_put\_type, 1811
- adjustfield, 1842
- app, 1842
- ate, 1842
- bad, 1817
- badbit, 1843
- basefield, 1843
- basic\_ofstream, 1815, 1816
- beg, 1843
- binary, 1843
- boolalpha, 1844
- char\_type, 1811
- clear, 1817
- close, 1818
- copyfmt, 1818
- cur, 1844
- dec, 1844
- end, 1844
- eof, 1819
- eofbit, 1844
- event, 1815
- event\_callback, 1811
- exceptions, 1819
- fail, 1820
- failbit, 1845
- fill, 1820, 1821
- fixed, 1845
- flags, 1821
- floatfield, 1845
- flush, 1822
- fmtflags, 1812
- getloc, 1822
- good, 1822
- goodbit, 1845
- hex, 1846
- imbue, 1823
- in, 1846
- init, 1823
- int\_type, 1813
- internal, 1846
- iostate, 1813
- is\_open, 1823
- iword, 1824
- left, 1846
- narrow, 1824
- oct, 1846
- off\_type, 1813
- open, 1825
- openmode, 1814
- operator void \*, 1825
- operator<<, 1826–1832
- out, 1847
- pos\_type, 1814
- precision, 1833
- put, 1833
- pword, 1834
- rdbuf, 1834
- rdstate, 1835
- register\_callback, 1835
- right, 1847
- scientific, 1847
- seekdir, 1814
- seekp, 1836
- setf, 1837
- setstate, 1837
- showbase, 1847
- showpoint, 1847
- showpos, 1847
- skipws, 1848
- sync\_with\_stdio, 1838
- tellp, 1838
- tie, 1839
- traits\_type, 1815

- trunc, 1848
- unitbuf, 1848
- unsetf, 1840
- uppercase, 1848
- widen, 1840
- width, 1840, 1841
- write, 1841
- xalloc, 1842
- std::basic\_ostream, 1848
  - ~basic\_ostream, 1859
  - \_M\_getloc, 1859
  - \_M\_write, 1860
  - \_\_ctype\_type, 1854
  - \_\_num\_get\_type, 1854
  - \_\_num\_put\_type, 1854
- adjustfield, 1884
- app, 1884
- ate, 1885
- bad, 1860
- badbit, 1885
- basefield, 1885
- basic\_ostream, 1859
- beg, 1885
- binary, 1886
- boolalpha, 1886
- char\_type, 1854
- clear, 1860
- copyfmt, 1861
- cur, 1886
- dec, 1886
- end, 1886
- eof, 1861
- eofbit, 1886
- event, 1859
- event\_callback, 1855
- exceptions, 1862
- fail, 1863
- failbit, 1887
- fill, 1863
- fixed, 1887
- flags, 1864
- floatfield, 1887
- flush, 1865
- fmtflags, 1855
- getloc, 1865
- good, 1865
- goodbit, 1887
- hex, 1888
- imbue, 1866
- in, 1888
- init, 1866
- int\_type, 1856
- internal, 1888
- iostate, 1856
- iword, 1866
- left, 1888
- narrow, 1867
- oct, 1889
- off\_type, 1857
- openmode, 1857
- operator void \*, 1867
- operator<<, 1868–1874
- out, 1889
- pos\_type, 1858
- precision, 1874, 1875
- put, 1875
- pword, 1876
- rdbuf, 1876, 1877
- rdstate, 1877
- register\_callback, 1878
- right, 1889
- scientific, 1889
- seekdir, 1858
- seekp, 1878
- setf, 1879
- setstate, 1880
- showbase, 1889
- showpoint, 1889
- showpos, 1890
- skipws, 1890
- sync\_with\_stdio, 1880
- tellp, 1881
- tie, 1881
- traits\_type, 1858
- trunc, 1890
- unitbuf, 1890
- unsetf, 1882
- uppercase, 1890
- widen, 1882
- width, 1883
- write, 1883
- xalloc, 1884

- 
- std::basic\_ostream::sentry, 1891
    - ~sentry, 1891
    - operator bool, 1892
    - sentry, 1891
  - std::basic\_ostringstream, 1892
    - ~basic\_ostringstream, 1904
    - \_M\_getloc, 1904
    - \_M\_write, 1905
    - \_\_ctype\_type, 1899
    - \_\_num\_get\_type, 1899
    - \_\_num\_put\_type, 1899
    - adjustfield, 1930
    - app, 1930
    - ate, 1930
    - bad, 1905
    - badbit, 1930
    - basefield, 1930
    - basic\_ostringstream, 1903
    - beg, 1931
    - binary, 1931
    - boolalpha, 1931
    - char\_type, 1899
    - clear, 1905
    - copyfmt, 1906
    - cur, 1931
    - dec, 1931
    - end, 1932
    - eof, 1906
    - eofbit, 1932
    - event, 1903
    - event\_callback, 1899
    - exceptions, 1907
    - fail, 1908
    - failbit, 1932
    - fill, 1908
    - fixed, 1932
    - flags, 1909
    - floatfield, 1932
    - flush, 1909
    - fmtflags, 1900
    - getloc, 1910
    - good, 1910
    - goodbit, 1933
    - hex, 1933
    - imbue, 1910
    - in, 1933
    - init, 1911
    - int\_type, 1901
    - internal, 1933
    - iostate, 1901
    - iword, 1911
    - left, 1934
    - narrow, 1912
    - oct, 1934
    - off\_type, 1901
    - openmode, 1902
    - operator void \*, 1912
    - operator<<, 1913–1919
    - out, 1934
    - pos\_type, 1902
    - precision, 1919, 1920
    - put, 1920
    - pword, 1920
    - rdbuf, 1921, 1922
    - rdstate, 1922
    - register\_callback, 1922
    - right, 1934
    - scientific, 1934
    - seekdir, 1902
    - seekp, 1922, 1923
    - setf, 1923, 1924
    - setstate, 1924
    - showbase, 1934
    - showpoint, 1935
    - showpos, 1935
    - skipws, 1935
    - str, 1925
    - sync\_with\_stdio, 1925
    - tellp, 1926
    - tie, 1926, 1927
    - traits\_type, 1903
    - trunc, 1935
    - unitbuf, 1935
    - unsetf, 1927
    - uppercase, 1935
    - widen, 1927
    - width, 1928
    - write, 1928
    - xalloc, 1929
  - std::basic\_streambuf, 1936
    - ~basic\_streambuf, 1942
    - \_M\_buf\_locale, 1958
-

---

[\\_M\\_in\\_beg](#), 1958  
[\\_M\\_in\\_cur](#), 1959  
[\\_M\\_in\\_end](#), 1959  
[\\_M\\_out\\_beg](#), 1959  
[\\_M\\_out\\_cur](#), 1960  
[\\_M\\_out\\_end](#), 1960  
[\\_\\_streambuf\\_type](#), 1940  
[basic\\_streambuf](#), 1942  
[char\\_type](#), 1940  
[eback](#), 1942  
[egptr](#), 1942  
[eptr](#), 1943  
[gbump](#), 1943  
[getloc](#), 1944  
[gptr](#), 1944  
[imbue](#), 1944  
[in\\_avail](#), 1945  
[int\\_type](#), 1940  
[off\\_type](#), 1940  
[overflow](#), 1945  
[pbackfail](#), 1946  
[pbase](#), 1946  
[pbump](#), 1947  
[pos\\_type](#), 1941  
[pptr](#), 1947  
[pubimbue](#), 1948  
[pubseekoff](#), 1948  
[pubseekpos](#), 1948  
[pubsetbuf](#), 1949  
[pubsync](#), 1949  
[sbumpc](#), 1949  
[seekoff](#), 1950  
[seekpos](#), 1950  
[setbuf](#), 1951  
[setg](#), 1951  
[setp](#), 1951  
[sgetc](#), 1952  
[sgetn](#), 1952  
[showmanyc](#), 1952  
[snextc](#), 1953  
[sputbackc](#), 1953  
[sputc](#), 1954  
[sputn](#), 1954  
[stossc](#), 1955  
[sungetc](#), 1955  
[sync](#), 1955  
[traits\\_type](#), 1941  
[uflow](#), 1956  
[underflow](#), 1956  
[xsgetn](#), 1957  
[xsputn](#), 1958  
[std::basic\\_string](#), 1961  
[~basic\\_string](#), 1970  
[append](#), 1970–1973  
[assign](#), 1973–1976  
[at](#), 1976, 1977  
[basic\\_string](#), 1966–1969  
[begin](#), 1977, 1978  
[c\\_str](#), 1978  
[capacity](#), 1978  
[cbegin](#), 1978  
[cend](#), 1979  
[clear](#), 1979  
[compare](#), 1979–1982  
[copy](#), 1983  
[cbegin](#), 1983  
[crend](#), 1984  
[data](#), 1984  
[empty](#), 1984  
[end](#), 1984, 1985  
[erase](#), 1985, 1986  
[find](#), 1986–1988  
[find\\_first\\_not\\_of](#), 1988–1990  
[find\\_first\\_of](#), 1990–1992  
[find\\_last\\_not\\_of](#), 1992–1994  
[find\\_last\\_of](#), 1994, 1995  
[get\\_allocator](#), 1996  
[insert](#), 1996–2000  
[length](#), 2001  
[max\\_size](#), 2001  
[npos](#), 2018  
[operator+=](#), 2001, 2002  
[operator=](#), 2003, 2004  
[push\\_back](#), 2005  
[rbegin](#), 2005, 2006  
[rend](#), 2006  
[replace](#), 2006–2013  
[reserve](#), 2013  
[resize](#), 2014  
[rfind](#), 2015, 2016  
[shrink\\_to\\_fit](#), 2017  
[size](#), 2017

---

- substr, 2017
- swap, 2018
- std::basic\_stringbuf, 2019
  - \_M\_buf\_locale, 2042
  - \_M\_in\_beg, 2042
  - \_M\_in\_cur, 2042
  - \_M\_in\_end, 2043
  - \_M\_mode, 2043
  - \_M\_out\_beg, 2043
  - \_M\_out\_cur, 2044
  - \_M\_out\_end, 2044
  - \_\_streambuf\_type, 2022
- basic\_stringbuf, 2024
- char\_type, 2022
- eback, 2025
- egptr, 2025
- epptr, 2025
- gbump, 2026
- getloc, 2026
- gptr, 2026
- imbue, 2027
- in\_avail, 2027
- int\_type, 2023
- off\_type, 2023
- overflow, 2028
- pbackfail, 2028
- pbase, 2029
- pbump, 2029
- pos\_type, 2023
- pptr, 2030
- pubimbue, 2030
- pubseekoff, 2030
- pubseekpos, 2031
- pubsetbuf, 2031
- pubsync, 2031
- sbumpc, 2032
- seekoff, 2032
- seekpos, 2033
- setbuf, 2033
- setg, 2034
- setp, 2034
- sgetc, 2035
- sgetn, 2035
- showmanyc, 2035
- snextc, 2036
- sputbackc, 2036
- sputc, 2037
- sputn, 2037
- stossc, 2038
- str, 2038
- sungetc, 2038
- sync, 2039
- traits\_type, 2023
- uflow, 2039
- underflow, 2040
- xsgetn, 2040
- xspn, 2041
- std::basic\_stringstream, 2045
  - ~basic\_stringstream, 2060
  - \_M\_gcount, 2104
  - \_M\_getloc, 2060
  - \_M\_write, 2061
  - \_\_ctype\_type, 2053
  - \_\_num\_get\_type, 2053
  - \_\_num\_put\_type, 2053
  - adjustfield, 2104
  - app, 2105
  - ate, 2105
  - bad, 2061
  - badbit, 2105
  - basefield, 2105
  - basic\_stringstream, 2059, 2060
  - beg, 2105
  - binary, 2106
  - boolalpha, 2106
  - char\_type, 2054
  - clear, 2061
  - copyfmt, 2062
  - cur, 2106
  - dec, 2106
  - end, 2106
  - eof, 2062
  - eofbit, 2106
  - event, 2059
  - event\_callback, 2054
  - exceptions, 2063
  - fail, 2064
  - failbit, 2107
  - fill, 2064
  - fixed, 2107
  - flags, 2065
  - floatfield, 2107

- flush, 2066
- fmtflags, 2055
- gcount, 2066
- get, 2066–2069
- getline, 2070, 2071
- getloc, 2071
- good, 2071
- goodbit, 2107
- hex, 2108
- ignore, 2072, 2073
- imbue, 2073
- in, 2108
- init, 2074
- int\_type, 2055, 2056
- internal, 2108
- iostate, 2056
- isword, 2074
- left, 2108
- narrow, 2075
- oct, 2109
- off\_type, 2056, 2057
- openmode, 2057
- operator void \*, 2075
- operator<<, 2075–2082
- operator>>, 2082–2088
- out, 2109
- peek, 2089
- pos\_type, 2057, 2058
- precision, 2089
- put, 2090
- putback, 2090
- pword, 2091
- rdbuf, 2091, 2092
- rdstate, 2092
- read, 2093
- readsome, 2093
- register\_callback, 2094
- right, 2109
- scientific, 2109
- seekdir, 2058
- seekg, 2094, 2095
- seekp, 2096
- setf, 2097
- setstate, 2097
- showbase, 2109
- showpoint, 2109
- showpos, 2110
- skipws, 2110
- str, 2098
- sync, 2099
- sync\_with\_stdio, 2099
- tellg, 2099
- tellp, 2100
- tie, 2100, 2101
- traits\_type, 2058, 2059
- trunc, 2110
- unget, 2101
- unitbuf, 2110
- unsetf, 2102
- uppercase, 2110
- widen, 2102
- width, 2102, 2103
- write, 2103
- xalloc, 2104
- std::bernoulli\_distribution, 2111
- bernoulli\_distribution, 2112
- max, 2112
- min, 2112
- operator(), 2112
- p, 2112
- param, 2112, 2113
- reset, 2113
- result\_type, 2111
- std::bernoulli\_distribution::param\_type, 2113
- std::bidirectional\_iterator\_tag, 2114
- std::binary\_function, 2115
- first\_argument\_type, 2116
- result\_type, 2116
- second\_argument\_type, 2116
- std::binary\_negate, 2117
- first\_argument\_type, 2117
- result\_type, 2118
- second\_argument\_type, 2118
- std::binder1st, 2118
- argument\_type, 2119
- result\_type, 2119
- std::binder2nd, 2120
- argument\_type, 2121
- result\_type, 2121
- std::binomial\_distribution, 2121
- max, 2123

- min, 2123
- operator<<, 2125
- operator>>, 2125
- operator(), 2123
- operator==, 2125
- p, 2124
- param, 2124
- reset, 2124
- result\_type, 2123
- t, 2125
- std::binomial\_distribution::param\_type, 2126
- std::bitset, 2127
  - all, 2133
  - any, 2133
  - bitset, 2132, 2133
  - count, 2134
  - flip, 2134
  - none, 2134
  - operator<<, 2135
  - operator<<=, 2135
  - operator>>, 2136
  - operator>>=, 2136
  - operator~, 2138
  - operator^=, 2137
  - operator==, 2135
  - operator&=, 2135
  - reset, 2138
  - set, 2138, 2139
  - size, 2139
  - test, 2139
  - to\_string, 2139
  - to\_ulong, 2140
- std::bitset::reference, 2140
- std::cauchy\_distribution, 2141
  - max, 2142
  - min, 2142
  - operator(), 2143
  - param, 2143
  - reset, 2143
  - result\_type, 2142
- std::cauchy\_distribution::param\_type, 2144
- std::char\_traits, 2145
- std::char\_traits< \_\_gnu\_cxx::character< V, I, S >>, 2146
- std::char\_traits< char >, 2147
- std::char\_traits< wchar\_t >, 2148
- std::chi\_squared\_distribution, 2149
  - max, 2151
  - min, 2151
  - operator<<, 2152
  - operator>>, 2153
  - operator(), 2151
  - operator==, 2152
  - param, 2151, 2152
  - reset, 2152
  - result\_type, 2151
- std::chi\_squared\_distribution::param\_type, 2153
- std::chrono, 679
  - duration\_cast, 683
  - hours, 682
  - microseconds, 682
  - milliseconds, 682
  - minutes, 682
  - nanoseconds, 682
  - seconds, 683
  - time\_point\_cast, 683
- std::chrono::duration, 2154
- std::chrono::duration\_values, 2155
- std::chrono::system\_clock, 2156
- std::chrono::time\_point, 2157
- std::chrono::treat\_as\_floating\_point, 2158
- std::codecvt, 2158
  - do\_out, 2161
  - in, 2161
  - out, 2162
  - unshift, 2163
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2164
  - do\_out, 2167
  - in, 2167
  - out, 2168
  - unshift, 2169
- std::codecvt< char, char, mbstate\_t >, 2170
  - do\_out, 2172
  - in, 2172
  - out, 2173
  - unshift, 2174

- std::codecvt< wchar\_t, char, mbstate\_t >, 2175
  - do\_out, 2177
  - in, 2177
  - out, 2178
  - unshift, 2179
- std::codecvt\_base, 2180
- std::codecvt\_byname, 2181
  - do\_out, 2184
  - in, 2184
  - out, 2185
  - unshift, 2186
- std::collate, 2187
  - ~collate, 2191
  - char\_type, 2190
  - collate, 2190
  - compare, 2191
  - do\_compare, 2191
  - do\_hash, 2192
  - do\_transform, 2192
  - hash, 2193
  - id, 2194
  - string\_type, 2190
  - transform, 2193
- std::collate\_byname, 2194
  - char\_type, 2197
  - compare, 2197
  - do\_compare, 2197
  - do\_hash, 2198
  - do\_transform, 2198
  - hash, 2199
  - id, 2200
  - string\_type, 2197
  - transform, 2199
- std::complex, 2200
  - complex, 2202
  - operator+=", 2202
  - operator-=", 2202
  - value\_type, 2201
- std::condition\_variable, 2202
- std::condition\_variable\_any, 2203
- std::conditional, 2204
- std::const\_mem\_fun1\_ref\_t, 2205
  - first\_argument\_type, 2206
  - result\_type, 2206
  - second\_argument\_type, 2206
- std::const\_mem\_fun1\_t, 2207
  - first\_argument\_type, 2208
  - result\_type, 2208
  - second\_argument\_type, 2208
- std::const\_mem\_fun\_ref\_t, 2208
  - argument\_type, 2210
  - result\_type, 2210
- std::const\_mem\_fun\_t, 2210
  - argument\_type, 2212
  - result\_type, 2212
- std::ctype, 2212
  - char\_type, 2215
  - do\_is, 2216
  - do\_narrow, 2217
  - do\_scan\_is, 2218
  - do\_scan\_not, 2218
  - do\_tolower, 2219
  - do\_toupper, 2220
  - do\_widen, 2220, 2221
  - id, 2227
  - is, 2221, 2222
  - narrow, 2222, 2223
  - scan\_is, 2224
  - scan\_not, 2224
  - tolower, 2224, 2225
  - toupper, 2225, 2226
  - widen, 2226, 2227
- std::ctype< char >, 2228
  - ~ctype, 2231
  - char\_type, 2231
  - classic\_table, 2232
  - ctype, 2231
  - do\_narrow, 2232
  - do\_tolower, 2233
  - do\_toupper, 2234
  - do\_widen, 2234, 2235
  - id, 2241
  - is, 2235, 2236
  - narrow, 2236, 2237
  - scan\_is, 2237
  - scan\_not, 2238
  - table, 2238
  - table\_size, 2241
  - tolower, 2238, 2239
  - toupper, 2239, 2240
  - widen, 2240, 2241

- std::ctype< wchar\_t >, 2242
  - ~ctype, 2246
  - char\_type, 2245
  - ctype, 2246
  - do\_is, 2246, 2247
  - do\_narrow, 2248, 2249
  - do\_scan\_is, 2250
  - do\_scan\_not, 2251
  - do\_tolower, 2252, 2253
  - do\_toupper, 2254, 2255
  - do\_widen, 2255, 2256
  - id, 2262
  - is, 2257
  - narrow, 2258
  - scan\_is, 2259
  - scan\_not, 2259
  - tolower, 2260
  - toupper, 2260, 2261
  - widen, 2261, 2262
- std::ctype\_base, 2263
- std::ctype\_byname, 2264
  - char\_type, 2266
  - do\_is, 2267
  - do\_narrow, 2267, 2268
  - do\_scan\_is, 2269
  - do\_scan\_not, 2269
  - do\_tolower, 2270
  - do\_toupper, 2271
  - do\_widen, 2271, 2272
  - id, 2278
  - is, 2272, 2273
  - narrow, 2273, 2274
  - scan\_is, 2275
  - scan\_not, 2275
  - tolower, 2275, 2276
  - toupper, 2276, 2277
  - widen, 2277, 2278
- std::ctype\_byname< char >, 2279
  - char\_type, 2281
  - classic\_table, 2282
  - do\_narrow, 2282
  - do\_tolower, 2283
  - do\_toupper, 2284
  - do\_widen, 2284, 2285
  - id, 2291
  - is, 2285, 2286
  - narrow, 2286, 2287
  - scan\_is, 2287
  - scan\_not, 2288
  - table, 2288
  - table\_size, 2291
  - tolower, 2288, 2289
  - toupper, 2289, 2290
  - widen, 2290, 2291
- std::decay, 2292
- std::decimal, 683
  - decimal32\_to\_long\_long, 694
- std::decimal::decimal128, 2292
  - decimal128, 2294
- std::decimal::decimal32, 2294
  - decimal32, 2296
- std::decimal::decimal64, 2296
  - decimal64, 2298
- std::default\_delete, 2298
- std::defer\_lock\_t, 2299
- std::deque, 2300
  - ~deque, 2308
  - \_M\_fill\_initialize, 2309
  - \_M\_initialize\_map, 2309
  - \_M\_new\_elements\_at\_back, 2310
  - \_M\_new\_elements\_at\_front, 2310
  - \_M\_pop\_back\_aux, 2310
  - \_M\_pop\_front\_aux, 2310
  - \_M\_push\_back\_aux, 2311
  - \_M\_push\_front\_aux, 2311
  - \_M\_range\_check, 2311
  - \_M\_range\_initialize, 2311, 2312
  - \_M\_reallocate\_map, 2312
  - \_M\_reserve\_elements\_at\_back, 2313
  - \_M\_reserve\_elements\_at\_front, 2313
  - \_M\_reserve\_map\_at\_back, 2313
  - \_M\_reserve\_map\_at\_front, 2313
- assign, 2314, 2315
- at, 2315
- back, 2316
- begin, 2316, 2317
- cbegin, 2317
- cend, 2317
- clear, 2317
- crbegin, 2317

- crend, 2318
- deque, 2306–2308
- emplace, 2318
- empty, 2318
- end, 2319
- erase, 2319
- front, 2320
- get\_allocator, 2320
- insert, 2321, 2322
- max\_size, 2323
- operator=, 2323, 2324
- pop\_back, 2325
- pop\_front, 2325
- push\_back, 2326
- push\_front, 2326
- rbegin, 2326, 2327
- rend, 2327
- resize, 2327
- shrink\_to\_fit, 2328
- size, 2328
- swap, 2328
- std::discard\_block\_engine, 2329
  - base, 2332
  - discard, 2332
  - discard\_block\_engine, 2330, 2331
  - max, 2332
  - min, 2333
  - operator<<, 2334
  - operator>>, 2335
  - operator(), 2333
  - operator==, 2334
  - result\_type, 2330
  - seed, 2333, 2334
- std::discrete\_distribution, 2335
  - max, 2337
  - min, 2337
  - operator<<, 2338
  - operator>>, 2339
  - operator(), 2337
  - param, 2337, 2338
  - probabilities, 2338
  - reset, 2338
  - result\_type, 2337
- std::discrete\_distribution::param\_type, 2339
- std::divides, 2340
  - first\_argument\_type, 2341
  - result\_type, 2341
  - second\_argument\_type, 2341
- std::domain\_error, 2342
  - what, 2342
- std::enable\_if, 2343
- std::enable\_shared\_from\_this, 2343
- std::equal\_to, 2344
  - first\_argument\_type, 2346
  - result\_type, 2346
  - second\_argument\_type, 2346
- std::error\_category, 2346
- std::error\_code, 2347
- std::error\_condition, 2348
- std::exception, 2348
  - what, 2350
- std::exponential\_distribution, 2350
  - exponential\_distribution, 2352
  - lambda, 2352
  - max, 2352
  - min, 2352
  - operator(), 2352
  - param, 2353
  - reset, 2353
  - result\_type, 2351
- std::exponential\_distribution::param\_type, 2354
- std::extreme\_value\_distribution, 2354
  - a, 2356
  - b, 2356
  - max, 2356
  - min, 2356
  - operator(), 2356
  - param, 2357
  - reset, 2357
  - result\_type, 2355
- std::extreme\_value\_distribution::param\_type, 2357
- std::fisher\_f\_distribution, 2358
  - max, 2360
  - min, 2360
  - operator<<, 2361
  - operator>>, 2361
  - operator(), 2360
  - operator==, 2361
  - param, 2360

- reset, 2361
- result\_type, 2359
- std::fisher\_f\_distribution::param\_type, 2362
- std::forward\_iterator\_tag, 2363
- std::forward\_list, 2364
  - ~forward\_list, 2370
  - assign, 2371
  - before\_begin, 2372
  - begin, 2372
  - cbefore\_begin, 2373
  - cbegin, 2373
  - cend, 2373
  - clear, 2373
  - emplace\_after, 2374
  - emplace\_front, 2374
  - empty, 2374
  - end, 2375
  - erase\_after, 2375, 2376
  - forward\_list, 2368–2370
  - front, 2376
  - get\_allocator, 2376
  - insert\_after, 2377, 2378
  - max\_size, 2379
  - merge, 2379
  - operator=, 2380
  - pop\_front, 2381
  - push\_front, 2381
  - remove, 2381
  - remove\_if, 2382
  - resize, 2382
  - reverse, 2383
  - sort, 2383
  - splice\_after, 2383, 2384
  - swap, 2385
  - unique, 2385
- std::fpos, 2386
  - fpos, 2387
  - operator streamoff, 2387
  - operator+, 2387
  - operator+=", 2387
  - operator-, 2388
  - operator-=, 2388
  - state, 2388
- std::front\_insert\_iterator, 2389
  - container\_type, 2390
  - difference\_type, 2390
  - front\_insert\_iterator, 2391
  - iterator\_category, 2390
  - operator\*, 2391
  - operator++, 2391, 2392
  - operator=, 2392
  - pointer, 2390
  - reference, 2391
  - value\_type, 2391
- std::function< \_Res(\_ArgTypes...)>, 2392
  - function, 2395, 2396
  - operator bool, 2396
  - operator(), 2396
  - operator=, 2397–2399
  - swap, 2399
  - target, 2399, 2400
  - target\_type, 2400
- std::future, 2400
  - \_M\_get\_result, 2402
  - future, 2402
  - get, 2402
- std::future< \_Res & >, 2403
  - \_M\_get\_result, 2405
  - future, 2405
  - get, 2405
- std::future< void >, 2405
  - \_M\_get\_result, 2407
  - future, 2407
  - get, 2407
- std::future\_error, 2408
  - what, 2409
- std::gamma\_distribution, 2409
  - alpha, 2411
  - beta, 2411
  - gamma\_distribution, 2411
  - max, 2411
  - min, 2411
  - operator<<, 2413
  - operator>>, 2414
  - operator(), 2412
  - operator==, 2413
  - param, 2412
  - reset, 2413
  - result\_type, 2411

- 
- std::gamma\_distribution::param\_type, 2414
  - std::geometric\_distribution, 2415
    - max, 2416
    - min, 2416
    - operator(), 2417
    - p, 2417
    - param, 2417
    - reset, 2418
    - result\_type, 2416
  - std::geometric\_distribution::param\_type, 2418
  - std::greater, 2419
    - first\_argument\_type, 2420
    - result\_type, 2420
    - second\_argument\_type, 2420
  - std::greater\_equal, 2420
    - first\_argument\_type, 2422
    - result\_type, 2422
    - second\_argument\_type, 2422
  - std::gslice, 2422
  - std::gslice\_array, 2423
  - std::has\_nothrow\_assign, 2425
  - std::has\_nothrow\_copy\_constructor, 2425
  - std::has\_nothrow\_default\_constructor, 2426
  - std::has\_trivial\_assign, 2426
  - std::has\_trivial\_copy\_constructor, 2427
  - std::has\_trivial\_default\_constructor, 2427
  - std::has\_trivial\_destructor, 2427
  - std::hash, 2428
    - argument\_type, 2430
    - result\_type, 2430
  - std::hash< \_\_debug::bitset< \_Nb > >, 2430
    - argument\_type, 2432
    - result\_type, 2432
  - std::hash< \_\_debug::vector< bool, \_- Alloc > >, 2432
    - argument\_type, 2434
    - result\_type, 2434
  - std::hash< \_\_gnu\_cxx::throw\_value\_- limit >, 2434
    - argument\_type, 2436
    - result\_type, 2436
  - std::hash< \_\_gnu\_cxx::throw\_value\_- random >, 2436
    - argument\_type, 2438
    - result\_type, 2438
  - std::hash< \_\_profile::bitset< \_Nb > >, 2438
    - argument\_type, 2440
    - result\_type, 2440
  - std::hash< \_\_profile::vector< bool, \_- Alloc > >, 2440
    - argument\_type, 2442
    - result\_type, 2442
  - std::hash< \_Tp \* >, 2442
    - argument\_type, 2444
    - result\_type, 2444
  - std::hash< error\_code >, 2444
    - argument\_type, 2446
    - result\_type, 2446
  - std::hash< string >, 2446
    - argument\_type, 2448
    - result\_type, 2448
  - std::hash< thread::id >, 2448
    - argument\_type, 2450
    - result\_type, 2450
  - std::hash< u16string >, 2450
    - argument\_type, 2452
    - result\_type, 2452
  - std::hash< u32string >, 2452
    - argument\_type, 2454
    - result\_type, 2454
  - std::hash< wstring >, 2454
    - argument\_type, 2456
    - result\_type, 2456
  - std::hash<::bitset< \_Nb > >, 2456
    - argument\_type, 2458
    - result\_type, 2458
  - std::hash<::vector< bool, \_Alloc > >, 2458
    - argument\_type, 2460
    - result\_type, 2460
  - std::identity, 2460
  - std::independent\_bits\_engine, 2461
    - base, 2464
    - discard, 2464
    - independent\_bits\_engine, 2462, 2463
-

- max, 2464
- min, 2464
- operator>>, 2466
- operator(), 2465
- operator==, 2466
- result\_type, 2462
- seed, 2465
- std::indirect\_array, 2467
  - operator<=, 2469
  - operator>=, 2469
  - operator\*=, 2469
  - operator^=, 2469
  - operator+=, 2469
  - operator-=, 2469
  - operator/=, 2469
  - operator%=, 2468
  - operator&=, 2468
- std::initializer\_list, 2470
- std::input\_iterator\_tag, 2471
- std::insert\_iterator, 2472
  - container\_type, 2473
  - difference\_type, 2473
  - insert\_iterator, 2474
  - iterator\_category, 2473
  - operator\*, 2474
  - operator++, 2474, 2475
  - operator=, 2475
  - pointer, 2473
  - reference, 2474
  - value\_type, 2474
- std::invalid\_argument, 2476
  - what, 2476
- std::ios\_base, 2477
  - ~ios\_base, 2483
  - \_M\_getloc, 2483
  - adjustfield, 2489
  - app, 2489
  - ate, 2489
  - badbit, 2490
  - basefield, 2490
  - beg, 2490
  - binary, 2490
  - boolalpha, 2491
  - cur, 2491
  - dec, 2491
  - end, 2491
  - eofbit, 2491
  - event, 2483
  - event\_callback, 2480
  - failbit, 2492
  - fixed, 2492
  - flags, 2483, 2484
  - floatfield, 2492
  - fmtflags, 2480
  - getloc, 2484
  - goodbit, 2492
  - hex, 2493
  - imbue, 2484
  - in, 2493
  - internal, 2493
  - iostate, 2481
  - iword, 2485
  - left, 2493
  - oct, 2493
  - openmode, 2482
  - out, 2494
  - precision, 2485, 2486
  - pword, 2486
  - register\_callback, 2486
  - right, 2494
  - scientific, 2494
  - seekdir, 2482
  - setf, 2487
  - showbase, 2494
  - showpoint, 2494
  - showpos, 2494
  - skipws, 2495
  - sync\_with\_stdio, 2487
  - trunc, 2495
  - unitbuf, 2495
  - unsetf, 2488
  - uppercase, 2495
  - width, 2488
  - xalloc, 2489
- std::ios\_base::failure, 2495
  - what, 2496
- std::is\_base\_of, 2497
- std::is\_bind\_expression, 2497
- std::is\_bind\_expression< \_Bind< \_-  
Signature > >, 2498
- std::is\_bind\_expression< \_Bind\_result<  
\_Result, \_Signature > >, 2498

- std::is\_constructible, 2499
- std::is\_convertible, 2500
- std::is\_error\_code\_enum, 2500
- std::is\_error\_condition\_enum, 2501
- std::is\_explicitly\_convertible, 2501
- std::is\_lvalue\_reference, 2502
- std::is\_placeholder, 2503
- std::is\_placeholder< \_Placeholder< \_- Num > >, 2503
- std::is\_pod, 2504
- std::is\_reference, 2504
- std::is\_rvalue\_reference, 2505
- std::is\_signed, 2505
- std::is\_standard\_layout, 2506
- std::is\_trivial, 2506
- std::is\_unsigned, 2506
- std::istream\_iterator, 2507
  - difference\_type, 2508
  - istream\_iterator, 2509
  - iterator\_category, 2508
  - pointer, 2508
  - reference, 2509
  - value\_type, 2509
- std::istreambuf\_iterator, 2510
  - char\_type, 2511
  - difference\_type, 2511
  - equal, 2513
  - int\_type, 2511
  - istream\_type, 2511
  - istreambuf\_iterator, 2513
  - iterator\_category, 2512
  - operator\*, 2513
  - operator++, 2514
  - pointer, 2512
  - reference, 2512
  - streambuf\_type, 2512
  - traits\_type, 2512
  - value\_type, 2512
- std::iterator, 2514
  - difference\_type, 2515
  - iterator\_category, 2515
  - pointer, 2516
  - reference, 2516
  - value\_type, 2516
- std::iterator\_traits, 2517
- std::iterator\_traits< \_Tp \* >, 2518
- std::iterator\_traits< const \_Tp \* >, 2518
- std::length\_error, 2519
  - what, 2520
- std::less, 2520
  - first\_argument\_type, 2522
  - result\_type, 2522
  - second\_argument\_type, 2522
- std::less\_equal, 2522
  - first\_argument\_type, 2524
  - result\_type, 2524
  - second\_argument\_type, 2524
- std::linear\_congruential\_engine, 2524
  - discard, 2527
  - increment, 2530
  - linear\_congruential\_engine, 2526
  - max, 2527
  - min, 2527
  - modulus, 2530
  - multiplier, 2530
  - operator<<, 2529
  - operator>>, 2529
  - operator(), 2527
  - operator==, 2529
  - result\_type, 2526
  - seed, 2528
- std::list, 2531
  - \_M\_create\_node, 2538
  - assign, 2538, 2539
  - back, 2539
  - begin, 2539, 2540
  - cbegin, 2540
  - cend, 2540
  - clear, 2540
  - crbegin, 2540
  - crend, 2541
  - emplace, 2541
  - empty, 2541
  - end, 2541, 2542
  - erase, 2542
  - front, 2543
  - get\_allocator, 2543
  - insert, 2544, 2545
  - list, 2535–2537
  - max\_size, 2546
  - merge, 2546
  - operator=, 2547

- pop\_back, 2548
- pop\_front, 2548
- push\_back, 2548
- push\_front, 2549
- rbegin, 2549
- remove, 2549
- remove\_if, 2550
- rend, 2550
- resize, 2551
- reverse, 2551
- size, 2551
- sort, 2551, 2552
- splice, 2552, 2553
- swap, 2553
- unique, 2553, 2554
- std::locale, 2554
  - ~locale, 2558
  - all, 2562
  - category, 2556
  - classic, 2559
  - collate, 2562
  - combine, 2559
  - ctype, 2562
  - global, 2559
  - has\_facet, 2561
  - locale, 2557, 2558
  - messages, 2563
  - monetary, 2563
  - name, 2559
  - none, 2563
  - numeric, 2563
  - operator(), 2560
  - operator=, 2560
  - operator==, 2561
  - time, 2564
  - use\_facet, 2561
- std::locale::facet, 2564
  - ~facet, 2566
  - facet, 2566
- std::locale::id, 2566
  - has\_facet, 2567
  - id, 2567
  - use\_facet, 2568
- std::lock\_guard, 2568
- std::logic\_error, 2569
  - logic\_error, 2570
  - what, 2570
- std::logical\_and, 2570
  - first\_argument\_type, 2572
  - result\_type, 2572
  - second\_argument\_type, 2572
- std::logical\_not, 2572
  - argument\_type, 2574
  - result\_type, 2574
- std::logical\_or, 2574
  - first\_argument\_type, 2576
  - result\_type, 2576
  - second\_argument\_type, 2576
- std::lognormal\_distribution, 2576
  - max, 2578
  - min, 2578
  - operator<<, 2579
  - operator>>, 2580
  - operator(), 2578
  - operator==, 2579
  - param, 2578, 2579
  - reset, 2579
  - result\_type, 2578
- std::lognormal\_distribution::param\_type, 2580
- std::make\_signed, 2581
- std::make\_unsigned, 2581
- std::map, 2582
  - at, 2587
  - begin, 2588
  - cbegin, 2588
  - cend, 2588
  - clear, 2588
  - count, 2589
  - crbegin, 2589
  - crend, 2589
  - empty, 2590
  - end, 2590
  - equal\_range, 2590, 2591
  - erase, 2592
  - find, 2593
  - get\_allocator, 2594
  - insert, 2594–2596
  - key\_comp, 2596
  - lower\_bound, 2596, 2597
  - map, 2585–2587
  - max\_size, 2597

- operator=, 2597, 2598
- rbegin, 2599
- rend, 2599, 2600
- size, 2600
- swap, 2600
- upper\_bound, 2601
- value\_comp, 2601
- std::mask\_array, 2602
  - operator<<=, 2604
  - operator>>=, 2604
  - operator\*=, 2604
  - operator^=, 2604
  - operator+=, 2604
  - operator-=, 2604
  - operator/=, 2604
  - operator%=, 2603
  - operator&=, 2603
- std::mem\_fun1\_ref\_t, 2605
  - first\_argument\_type, 2606
  - result\_type, 2606
  - second\_argument\_type, 2606
- std::mem\_fun1\_t, 2607
  - first\_argument\_type, 2608
  - result\_type, 2608
  - second\_argument\_type, 2608
- std::mem\_fun\_ref\_t, 2608
  - argument\_type, 2610
  - result\_type, 2610
- std::mem\_fun\_t, 2610
  - argument\_type, 2612
  - result\_type, 2612
- std::messages, 2612
  - ~messages, 2616
  - char\_type, 2615
  - id, 2616
  - messages, 2615
  - string\_type, 2615
- std::messages\_base, 2616
- std::messages\_byname, 2617
  - char\_type, 2620
  - id, 2620
  - string\_type, 2620
- std::minus, 2620
  - first\_argument\_type, 2622
  - result\_type, 2622
  - second\_argument\_type, 2622
- std::modulus, 2622
  - first\_argument\_type, 2624
  - result\_type, 2624
  - second\_argument\_type, 2624
- std::money\_base, 2624
- std::money\_get, 2626
  - ~money\_get, 2629
  - char\_type, 2628
  - do\_get, 2629
  - get, 2630, 2631
  - id, 2631
  - iter\_type, 2628
  - money\_get, 2629
  - string\_type, 2628
- std::money\_put, 2632
  - ~money\_put, 2634
  - char\_type, 2634
  - do\_put, 2635
  - id, 2637
  - iter\_type, 2634
  - money\_put, 2634
  - put, 2636, 2637
  - string\_type, 2634
- std::moneypunct, 2638
  - ~moneypunct, 2642
  - char\_type, 2640
  - curr\_symbol, 2642
  - decimal\_point, 2642
  - do\_curr\_symbol, 2642
  - do\_decimal\_point, 2643
  - do\_frac\_digits, 2643
  - do\_grouping, 2644
  - do\_neg\_format, 2644
  - do\_negative\_sign, 2644
  - do\_pos\_format, 2645
  - do\_positive\_sign, 2645
  - do\_thousands\_sep, 2646
  - frac\_digits, 2646
  - grouping, 2646
  - id, 2650
  - intl, 2650
  - moneypunct, 2641
  - neg\_format, 2647
  - negative\_sign, 2648
  - pos\_format, 2648
  - positive\_sign, 2649

- string\_type, 2640
- thousands\_sep, 2649
- std::moneypunct\_byname, 2650
  - char\_type, 2653
  - curr\_symbol, 2654
  - decimal\_point, 2654
  - do\_curr\_symbol, 2654
  - do\_decimal\_point, 2655
  - do\_frac\_digits, 2655
  - do\_grouping, 2655
  - do\_neg\_format, 2656
  - do\_negative\_sign, 2656
  - do\_pos\_format, 2657
  - do\_positive\_sign, 2657
  - do\_thousands\_sep, 2658
  - frac\_digits, 2658
  - grouping, 2658
  - id, 2662
  - intl, 2662
  - neg\_format, 2659
  - negative\_sign, 2660
  - pos\_format, 2660
  - positive\_sign, 2661
  - string\_type, 2653
  - thousands\_sep, 2661
- std::move\_iterator, 2662
- std::multimap, 2663
  - begin, 2669
  - cbegin, 2669
  - cend, 2669
  - clear, 2669
  - count, 2670
  - crbegin, 2670
  - crend, 2670
  - empty, 2670
  - end, 2671
  - equal\_range, 2671, 2672
  - erase, 2672, 2673
  - find, 2674
  - get\_allocator, 2674
  - insert, 2675, 2676
  - key\_comp, 2676
  - lower\_bound, 2677
  - max\_size, 2678
  - multimap, 2666–2668
  - operator=, 2678, 2679
  - rbegin, 2679
  - rend, 2679, 2680
  - size, 2680
  - swap, 2680
  - upper\_bound, 2681
  - value\_comp, 2681
- std::multiplies, 2682
  - first\_argument\_type, 2683
  - result\_type, 2683
  - second\_argument\_type, 2683
- std::multiset, 2683
  - begin, 2688
  - cbegin, 2688
  - cend, 2689
  - clear, 2689
  - count, 2689
  - crbegin, 2689
  - crend, 2690
  - empty, 2690
  - end, 2690
  - equal\_range, 2690, 2691
  - erase, 2691, 2692
  - find, 2693
  - get\_allocator, 2694
  - insert, 2694, 2695
  - key\_comp, 2696
  - lower\_bound, 2696
  - max\_size, 2697
  - multiset, 2686–2688
  - operator<, 2700
  - operator=, 2697, 2698
  - operator==, 2701
  - rbegin, 2698
  - rend, 2698
  - size, 2698
  - swap, 2699
  - upper\_bound, 2699
  - value\_comp, 2700
- std::mutex, 2701
- std::negate, 2702
  - argument\_type, 2703
  - result\_type, 2703
- std::negative\_binomial\_distribution, 2703
  - k, 2705
  - max, 2705
  - min, 2705

- operator<<, 2707
- operator>>, 2707
- operator(), 2705
- operator==, 2707
- p, 2706
- param, 2706
- reset, 2706
- result\_type, 2705
- std::negative\_binomial\_  
distribution::param\_type,  
2708
- std::nested\_exception, 2709
- std::normal\_distribution, 2709
  - max, 2711
  - mean, 2711
  - min, 2712
  - normal\_distribution, 2711
  - operator<<, 2714
  - operator>>, 2714
  - operator(), 2712
  - operator==, 2714
  - param, 2712, 2713
  - reset, 2713
  - result\_type, 2711
  - stddev, 2713
- std::normal\_distribution::param\_type,  
2715
- std::not\_equal\_to, 2715
  - first\_argument\_type, 2717
  - result\_type, 2717
  - second\_argument\_type, 2717
- std::num\_get, 2717
  - ~num\_get, 2721
  - char\_type, 2720
  - do\_get, 2721–2727
  - get, 2728–2736
  - id, 2737
  - iter\_type, 2720
  - num\_get, 2721
- std::num\_put, 2737
  - ~num\_put, 2741
  - char\_type, 2740
  - do\_put, 2741–2744
  - id, 2752
  - iter\_type, 2740
  - num\_put, 2741
  - put, 2745–2751
- std::numeric\_limits, 2752
  - denorm\_min, 2754
  - digits, 2755
  - digits10, 2756
  - epsilon, 2754
  - has\_denorm, 2756
  - has\_denorm\_loss, 2756
  - has\_infinity, 2756
  - has\_quiet\_NaN, 2756
  - has\_signaling\_NaN, 2756
  - infinity, 2754
  - is\_bounded, 2756
  - is\_exact, 2757
  - is\_iec559, 2757
  - is\_integer, 2757
  - is\_modulo, 2757
  - is\_signed, 2757
  - is\_specialized, 2758
  - lowest, 2754
  - max, 2754
  - max\_digits10, 2758
  - max\_exponent, 2758
  - max\_exponent10, 2758
  - min, 2755
  - min\_exponent, 2758
  - min\_exponent10, 2758
  - quiet\_NaN, 2755
  - radix, 2759
  - round\_error, 2755
  - round\_style, 2759
  - signaling\_NaN, 2755
  - tinyness\_before, 2759
  - traps, 2759
- std::numeric\_limits< bool >, 2760
- std::numeric\_limits< char >, 2761
- std::numeric\_limits< char16\_t >, 2762
- std::numeric\_limits< char32\_t >, 2763
- std::numeric\_limits< double >, 2765
- std::numeric\_limits< float >, 2766
- std::numeric\_limits< int >, 2767
- std::numeric\_limits< long >, 2769
- std::numeric\_limits< long double >,  
2770
- std::numeric\_limits< long long >, 2771
- std::numeric\_limits< short >, 2773

- std::numeric\_limits< signed char >, 2774  
 std::numeric\_limits< unsigned char >, 2775  
 std::numeric\_limits< unsigned int >, 2777  
 std::numeric\_limits< unsigned long >, 2778  
 std::numeric\_limits< unsigned long long >, 2779  
 std::numeric\_limits< unsigned short >, 2781  
 std::numeric\_limits< wchar\_t >, 2782  
 std::numpunct, 2783  
   ~numpunct, 2787  
   char\_type, 2786  
   decimal\_point, 2787  
   do\_decimal\_point, 2788  
   do\_falsename, 2788  
   do\_grouping, 2788  
   do\_thousands\_sep, 2789  
   do\_truename, 2789  
   falsename, 2790  
   grouping, 2790  
   id, 2791  
   numpunct, 2786, 2787  
   string\_type, 2786  
   thousands\_sep, 2790  
   truename, 2791  
 std::numpunct\_byname, 2792  
   char\_type, 2794  
   decimal\_point, 2794  
   do\_decimal\_point, 2794  
   do\_falsename, 2795  
   do\_grouping, 2795  
   do\_thousands\_sep, 2795  
   do\_truename, 2796  
   falsename, 2796  
   grouping, 2796  
   id, 2798  
   string\_type, 2794  
   thousands\_sep, 2797  
   truename, 2797  
 std::once\_flag, 2798  
   call\_once, 2799  
 std::ostream\_iterator, 2799  
   char\_type, 2800  
   difference\_type, 2800  
   iterator\_category, 2801  
   operator=, 2803  
   ostream\_iterator, 2802  
   ostream\_type, 2801  
   pointer, 2801  
   reference, 2801  
   traits\_type, 2801  
   value\_type, 2801  
 std::ostreambuf\_iterator, 2803  
   char\_type, 2805  
   difference\_type, 2805  
   failed, 2807  
   iterator\_category, 2805  
   operator\*, 2807  
   operator++, 2807  
   operator=, 2807  
   ostream\_type, 2805  
   ostreambuf\_iterator, 2806  
   pointer, 2805  
   reference, 2805  
   streambuf\_type, 2805  
   traits\_type, 2806  
   value\_type, 2806  
 std::out\_of\_range, 2808  
   what, 2808  
 std::output\_iterator\_tag, 2809  
 std::overflow\_error, 2810  
   what, 2810  
 std::owner\_less< shared\_ptr< \_Tp > >, 2811  
 std::owner\_less< weak\_ptr< \_Tp > >, 2811  
 std::packaged\_task< \_Res(\_- ArgTypes...)>, 2812  
 std::pair, 2813  
   first, 2815  
   first\_type, 2814  
   pair, 2815  
   second, 2815  
   second\_type, 2814  
 std::piecewise\_constant\_distribution, 2816  
   densities, 2817  
   intervals, 2817  
   max, 2818

- min, 2818
- operator<<, 2819
- operator>>, 2819
- operator(), 2818
- param, 2818
- reset, 2819
- result\_type, 2817
- std::piecewise\_constant\_-  
distribution::param\_type,  
2820
- std::piecewise\_linear\_distribution, 2821
  - densities, 2823
  - intervals, 2823
  - max, 2823
  - min, 2823
  - operator<<, 2824
  - operator>>, 2825
  - operator(), 2823
  - param, 2824
  - reset, 2824
  - result\_type, 2822
- std::piecewise\_linear\_-  
distribution::param\_type,  
2825
- std::placeholders, 694
- std::plus, 2826
  - first\_argument\_type, 2828
  - result\_type, 2828
  - second\_argument\_type, 2828
- std::pointer\_to\_binary\_function, 2828
  - first\_argument\_type, 2830
  - result\_type, 2830
  - second\_argument\_type, 2830
- std::pointer\_to\_unary\_function, 2830
  - argument\_type, 2832
  - result\_type, 2832
- std::poisson\_distribution, 2832
  - max, 2834
  - mean, 2834
  - min, 2834
  - operator<<, 2836
  - operator>>, 2836
  - operator(), 2834
  - operator==, 2836
  - param, 2835
  - reset, 2835
  - result\_type, 2833
- std::poisson\_distribution::param\_type,  
2837
- std::priority\_queue, 2837
  - empty, 2840
  - pop, 2840
  - priority\_queue, 2839
  - push, 2841
  - size, 2841
  - top, 2841
- std::promise, 2842
- std::promise< \_Res & >, 2843
- std::promise< void >, 2843
- std::queue, 2844
  - back, 2846
  - c, 2848
  - empty, 2846
  - front, 2846, 2847
  - pop, 2847
  - push, 2847
  - queue, 2846
  - size, 2847
- std::random\_access\_iterator\_tag, 2848
- std::random\_device, 2849
  - result\_type, 2850
- std::range\_error, 2851
  - what, 2851
- std::ratio, 2852
- std::ratio\_add, 2853
- std::ratio\_divide, 2853
- std::ratio\_equal, 2854
- std::ratio\_greater, 2854
- std::ratio\_greater\_equal, 2855
- std::ratio\_less, 2855
- std::ratio\_less\_equal, 2856
- std::ratio\_multiply, 2856
- std::ratio\_not\_equal, 2857
- std::ratio\_subtract, 2857
- std::raw\_storage\_iterator, 2858
  - difference\_type, 2859
  - iterator\_category, 2859
  - pointer, 2859
  - reference, 2859
  - value\_type, 2859
- std::recursive\_mutex, 2860
- std::recursive\_timed\_mutex, 2860

- 
- std::reference\_wrapper, 2861
  - std::rel\_ops, 695
    - operator<=, 696
    - operator>, 696
    - operator>=, 696
  - std::remove\_reference, 2863
  - std::reverse\_iterator, 2863
    - base, 2867
    - difference\_type, 2865
    - iterator\_category, 2865
    - operator\*, 2867
    - operator+, 2868
    - operator++, 2868
    - operator+=, 2868
    - operator-, 2869
    - operator->, 2870
    - operator--, 2869
    - operator-=, 2870
    - pointer, 2865
    - reference, 2866
    - reverse\_iterator, 2866, 2867
    - value\_type, 2866
  - std::runtime\_error, 2871
    - runtime\_error, 2872
    - what, 2872
  - std::seed\_seq, 2872
    - result\_type, 2873
    - seed\_seq, 2873
  - std::set, 2873
    - allocator\_type, 2876
    - begin, 2881
    - cbegin, 2881
    - end, 2882
    - clear, 2882
    - const\_iterator, 2876
    - const\_pointer, 2876
    - const\_reference, 2876
    - const\_reverse\_iterator, 2876
    - count, 2882
    - crbegin, 2882
    - crend, 2883
    - difference\_type, 2877
    - empty, 2883
    - end, 2883
    - equal\_range, 2883, 2884
    - erase, 2884, 2885
    - find, 2886
    - get\_allocator, 2886
    - insert, 2887, 2888
    - iterator, 2877
    - key\_comp, 2888
    - key\_compare, 2877
    - key\_type, 2877
    - lower\_bound, 2889
    - max\_size, 2889
    - operator=, 2890
    - pointer, 2877
    - rbegin, 2891
    - reference, 2878
    - rend, 2891
    - reverse\_iterator, 2878
    - set, 2879–2881
    - size, 2891
    - size\_type, 2878
    - swap, 2891
    - upper\_bound, 2892
    - value\_comp, 2893
    - value\_compare, 2878
    - value\_type, 2878
  - std::shared\_future, 2893
    - \_M\_get\_result, 2895
    - get, 2895
    - shared\_future, 2894, 2895
  - std::shared\_future<\_Res & >, 2895
    - \_M\_get\_result, 2898
    - get, 2898
    - shared\_future, 2897
  - std::shared\_future<void >, 2898
    - \_M\_get\_result, 2901
    - shared\_future, 2900
  - std::shared\_ptr, 2901
    - allocate\_shared, 2906
    - shared\_ptr, 2902–2905
  - std::shuffle\_order\_engine, 2907
    - base, 2910
    - discard, 2910
    - max, 2910
    - min, 2911
    - operator<<, 2912
    - operator>>, 2913
    - operator(), 2911
    - operator==, 2912
-

- result\_type, 2908
- seed, 2911
- shuffle\_order\_engine, 2908, 2909
- std::slice, 2913
- std::slice\_array, 2914
  - operator<<=, 2916
  - operator>>=, 2916
  - operator\*=, 2916
  - operator^=, 2916
  - operator+=, 2916
  - operator-=, 2916
  - operator/=, 2916
  - operator%=, 2916
  - operator&=, 2916
- std::stack, 2917
  - empty, 2919
  - pop, 2919
  - push, 2919
  - size, 2919
  - stack, 2918
  - top, 2919, 2920
- std::student\_t\_distribution, 2920
  - max, 2922
  - min, 2922
  - operator<<, 2923
  - operator>>, 2923
  - operator(), 2922
  - operator==, 2923
  - param, 2922
  - reset, 2923
  - result\_type, 2921
- std::student\_t\_distribution::param\_type, 2924
- std::system\_error, 2925
  - what, 2926
- std::this\_thread, 697
  - get\_id, 697
  - sleep\_for, 697
  - sleep\_until, 698
  - yield, 698
- std::thread, 2926
  - native\_handle, 2927
- std::thread::id, 2927
- std::time\_base, 2928
- std::time\_get, 2930
  - ~time\_get, 2933
- char\_type, 2932
- date\_order, 2933
- do\_date\_order, 2933
- do\_get\_date, 2934
- do\_get\_monthname, 2934
- do\_get\_time, 2935
- do\_get\_weekday, 2936
- do\_get\_year, 2936
- get\_date, 2937
- get\_monthname, 2938
- get\_time, 2938
- get\_weekday, 2939
- get\_year, 2940
- id, 2941
- iter\_type, 2932
- time\_get, 2933
- std::time\_get\_byname, 2941
  - char\_type, 2943
  - date\_order, 2944
  - do\_date\_order, 2944
  - do\_get\_date, 2944
  - do\_get\_monthname, 2945
  - do\_get\_time, 2946
  - do\_get\_weekday, 2946
  - do\_get\_year, 2947
  - get\_date, 2948
  - get\_monthname, 2948
  - get\_time, 2949
  - get\_weekday, 2950
  - get\_year, 2950
  - id, 2951
  - iter\_type, 2943
- std::time\_put, 2952
  - ~time\_put, 2954
  - char\_type, 2954
  - do\_put, 2955
  - id, 2957
  - iter\_type, 2954
  - put, 2955, 2956
  - time\_put, 2954
- std::time\_put\_byname, 2957
  - char\_type, 2959
  - do\_put, 2959
  - id, 2961
  - iter\_type, 2959
  - put, 2960

- std::timed\_mutex, 2961
- std::tr1, 698
- std::tr1::\_\_detail, 706
- std::tr1::\_\_is\_member\_pointer\_helper, 2962
- std::tr1::add\_const, 2963
- std::tr1::add\_cv, 2964
- std::tr1::add\_pointer, 2964
- std::tr1::add\_volatile, 2965
- std::tr1::alignment\_of, 2965
- std::tr1::array, 2967
- std::tr1::bad\_weak\_ptr, 2968
  - what, 2969
- std::tr1::extent, 2970
- std::tr1::has\_virtual\_destructor, 2971
- std::tr1::integral\_constant, 2972
- std::tr1::is\_abstract, 2974
- std::tr1::is\_arithmetic, 2975
- std::tr1::is\_array, 2976
- std::tr1::is\_class, 2977
- std::tr1::is\_compound, 2978
- std::tr1::is\_const, 2979
- std::tr1::is\_empty, 2980
- std::tr1::is\_enum, 2982
- std::tr1::is\_floating\_point, 2983
- std::tr1::is\_function, 2984
- std::tr1::is\_fundamental, 2985
- std::tr1::is\_integral, 2985
- std::tr1::is\_member\_function\_pointer, 2986
- std::tr1::is\_member\_object\_pointer, 2987
- std::tr1::is\_object, 2988
- std::tr1::is\_pointer, 2989
- std::tr1::is\_polymorphic, 2989
- std::tr1::is\_same, 2991
- std::tr1::is\_scalar, 2992
- std::tr1::is\_union, 2992
- std::tr1::is\_void, 2994
- std::tr1::is\_volatile, 2994
- std::tr1::rank, 2995
- std::tr1::remove\_all\_extents, 2997
- std::tr1::remove\_const, 2997
- std::tr1::remove\_cv, 2998
- std::tr1::remove\_extent, 2998
- std::tr1::remove\_pointer, 2999
- std::tr1::remove\_volatile, 2999
- std::try\_to\_lock\_t, 3000
- std::tuple, 3000
- std::tuple< \_T1, \_T2 >, 3001
- std::tuple\_element< 0, tuple< \_Head, \_Tail...> >, 3002
- std::tuple\_element< \_\_i, tuple< \_Head, \_Tail...> >, 3003
- std::tuple\_size< tuple< \_Elements...> >, 3003
- std::type\_info, 3004
  - ~type\_info, 3005
  - name, 3005
- std::unary\_function, 3007
  - argument\_type, 3008
  - result\_type, 3008
- std::unary\_negate, 3008
  - argument\_type, 3010
  - result\_type, 3010
- std::underflow\_error, 3011
  - what, 3011
- std::uniform\_int\_distribution, 3012
  - max, 3013
  - min, 3013
  - operator(), 3014
  - param, 3014
  - reset, 3014
  - result\_type, 3013
  - uniform\_int\_distribution, 3013
- std::uniform\_int\_distribution::param\_type, 3015
- std::uniform\_real\_distribution, 3016
  - max, 3017
  - min, 3017
  - operator(), 3017
  - param, 3018
  - reset, 3018
  - result\_type, 3017
  - uniform\_real\_distribution, 3017
- std::uniform\_real\_distribution::param\_type, 3019
- std::unique\_lock, 3019
- std::unique\_ptr, 3021
- std::unordered\_map, 3023
- std::unordered\_multimap, 3026
- std::unordered\_multiset, 3029
- std::unordered\_set, 3032

- 
- std::valarray, 3034
    - valarray, 3038
  - std::vector, 3038
    - ~vector, 3044
    - \_M\_allocate\_and\_copy, 3045
    - \_M\_range\_check, 3045
    - assign, 3045, 3046
    - at, 3046, 3047
    - back, 3047
    - begin, 3047, 3048
    - capacity, 3048
    - cbegin, 3048
    - cend, 3048
    - clear, 3049
    - crbegin, 3049
    - crend, 3049
    - data, 3049
    - emplace, 3049
    - empty, 3050
    - end, 3050
    - erase, 3051
    - front, 3052
    - insert, 3052–3054
    - max\_size, 3054
    - operator=, 3054, 3055
    - pop\_back, 3056
    - push\_back, 3057
    - rbegin, 3057
    - rend, 3057, 3058
    - reserve, 3058
    - resize, 3058
    - shrink\_to\_fit, 3059
    - size, 3059
    - swap, 3059
    - vector, 3042–3044
  - std::vector< bool, \_Alloc >, 3060
  - std::weak\_ptr, 3064
  - std::weibull\_distribution, 3065
    - a, 3066
    - b, 3066
    - max, 3066
    - min, 3066
    - operator(), 3066
    - param, 3067
    - reset, 3067
    - result\_type, 3066
  - std::weibull\_distribution::param\_type, 3068
  - stdatomic.h, 3379
  - stddev
    - std::normal\_distribution, 2713
  - stdexcept, 3379
  - stdio\_filebuf
    - \_\_gnu\_cxx::stdio\_filebuf, 890, 891
  - stdio\_filebuf.h, 3380
  - stdio\_sync\_filebuf.h, 3380
  - stl\_algo.h, 3381
  - stl\_algobase.h, 3394
  - stl\_bvector.h, 3397
  - stl\_construct.h, 3398
  - stl\_deque.h, 3399
    - \_GLIBCXX\_DEQUE\_BUF\_SIZE, 3402
  - stl\_function.h, 3402
  - stl\_heap.h, 3405
  - stl\_iterator.h, 3407
  - stl\_iterator\_base\_funcs.h, 3411
  - stl\_iterator\_base\_types.h, 3412
  - stl\_list.h, 3413
  - stl\_map.h, 3415
  - stl\_multimap.h, 3416
  - stl\_multiset.h, 3417
  - stl\_numeric.h, 3418
  - stl\_pair.h, 3419
  - stl\_queue.h, 3420
  - stl\_raw\_storage\_iter.h, 3421
  - stl\_relops.h, 3422
  - stl\_set.h, 3423
  - stl\_stack.h, 3424
  - stl\_tempbuf.h, 3425
  - stl\_tree.h, 3425
  - stl\_uninitialized.h, 3427
  - stl\_vector.h, 3429
  - stoss
    - \_\_gnu\_cxx::enc\_filebuf, 836
    - \_\_gnu\_cxx::stdio\_filebuf, 908
    - \_\_gnu\_cxx::stdio\_sync\_filebuf, 936
  - std::basic\_filebuf, 1473
  - std::basic\_streambuf, 1955
  - std::basic\_stringbuf, 2038
  - str
    - std::basic\_istream, 1793
-

- std::basic\_ostringstream, 1925
- std::basic\_stringbuf, 2038
- std::basic\_stringstream, 2098
- stream\_iterator.h, 3430
- streambuf, 3430
  - io, 58
- streambuf.tcc, 3431
- streambuf\_iterator.h, 3432
- streambuf\_type
  - std::istreambuf\_iterator, 2512
  - std::ostreambuf\_iterator, 2805
- streamoff
  - std, 553
- streampos
  - std, 554
- streamsize
  - std, 554
- stride
  - numeric\_arrays, 103
- string, 3433
- strings, 234
- string\_type
  - std::collate, 2190
  - std::collate\_byname, 2197
  - std::messages, 2615
  - std::messages\_byname, 2620
  - std::money\_get, 2628
  - std::money\_put, 2634
  - std::moneypunct, 2640
  - std::moneypunct\_byname, 2653
  - std::numpunct, 2786
  - std::numpunct\_byname, 2794
- stringbuf
  - io, 58
- stringfwd.h, 3436
- Strings, 233
- strings
  - string, 234
  - u16string, 234
  - u32string, 234
  - wstring, 234
- stringstream
  - io, 58
- substr
  - \_\_gnu\_cxx::\_\_versa\_string, 791
  - \_\_gnu\_debug::basic\_string, 1032
- std::basic\_string, 2017
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, 943
- sum
  - numeric\_arrays, 103
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf, 836
  - \_\_gnu\_cxx::stdio\_filebuf, 909
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 936
- std::basic\_filebuf, 1473
- std::basic\_streambuf, 1955
- std::basic\_stringbuf, 2038
- swap
  - \_\_gnu\_cxx, 334
  - \_\_gnu\_cxx::\_\_versa\_string, 792
  - \_\_gnu\_debug::basic\_string, 1032
  - mutating\_algorithms, 141
  - std, 633–635
  - std::basic\_string, 2018
  - std::deque, 2328
  - std::forward\_list, 2385
  - std::function<
    - \_Res(\_-
    - ArgTypes...)>, 2399
  - std::list, 2553
  - std::map, 2600
  - std::multimap, 2680
  - std::multiset, 2699
  - std::set, 2891
  - std::vector, 3059
- swap\_ranges
  - mutating\_algorithms, 141
- sync
  - \_\_gnu\_cxx::enc\_filebuf, 836
  - \_\_gnu\_cxx::stdio\_filebuf, 909
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 936
- std::basic\_filebuf, 1474
- std::basic\_fstream, 1537
- std::basic\_ifstream, 1592
- std::basic\_iostream, 1686
- std::basic\_istream, 1738
- std::basic\_istreamstream, 1793
- std::basic\_streambuf, 1955
- std::basic\_stringbuf, 2039
- std::basic\_stringstream, 2099
- sync\_with\_stdio
  - std::basic\_fstream, 1538

- std::basic\_ifstream, 1592
- std::basic\_ios, 1624
- std::basic\_iostream, 1687
- std::basic\_istream, 1739
- std::basic\_istream, 1794
- std::basic\_ofstream, 1838
- std::basic\_ostream, 1880
- std::basic\_ostringstream, 1925
- std::basic\_stringstream, 2099
- std::ios\_base, 2487
- system\_error, 3437
- t
  - std::binomial\_distribution, 2125
- table
  - std::ctype< char >, 2238
  - std::ctype\_byname< char >, 2288
- table\_size
  - std::ctype< char >, 2241
  - std::ctype\_byname< char >, 2291
- tag\_and\_trait.hpp, 3438
- tags.h, 3440
- tan
  - complex\_numbers, 49
- tanh
  - complex\_numbers, 49
- target
  - std::function< \_Res(\_- ArgTypes...)>, 2399, 2400
- target\_type
  - std::function< \_Res(\_- ArgTypes...)>, 2400
- tellg
  - std::basic\_fstream, 1538
  - std::basic\_ifstream, 1593
  - std::basic\_iostream, 1687
  - std::basic\_istream, 1739
  - std::basic\_istream, 1794
  - std::basic\_stringstream, 2099
- tellp
  - std::basic\_fstream, 1539
  - std::basic\_iostream, 1687
  - std::basic\_ofstream, 1838
  - std::basic\_ostream, 1881
  - std::basic\_ostringstream, 1926
  - std::basic\_stringstream, 2100
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer, 945
- terminate
  - exceptions, 33
- terminate\_handler
  - exceptions, 31
- test
  - std::bitset, 2139
- tgmath.h, 3442
- thousands\_sep
  - std::moneypunct, 2649
  - std::moneypunct\_byname, 2661
  - std::numpunct, 2790
  - std::numpunct\_byname, 2797
- thread, 3442
- Threads, 71
- throw\_allocator.h, 3443
- throw\_with\_nested
  - exceptions, 33
- tie
  - std::basic\_fstream, 1539
  - std::basic\_ifstream, 1593
  - std::basic\_ios, 1625
  - std::basic\_iostream, 1688
  - std::basic\_istream, 1740
  - std::basic\_istream, 1795
  - std::basic\_ofstream, 1839
  - std::basic\_ostream, 1881
  - std::basic\_ostringstream, 1926, 1927
  - std::basic\_stringstream, 2100, 2101
- Time, 34
- time
  - std::locale, 2564
- time\_get
  - std::time\_get, 2933
- time\_members.h, 3446
- time\_point\_cast
  - std::chrono, 683
- time\_put
  - std::time\_put, 2954
- tinyness\_before
  - std::\_\_numeric\_limits\_base, 1342
  - std::numeric\_limits, 2759
- TLB\_size
  - \_\_gnu\_parallel::\_Settings, 1156
- to\_string

- std::bitset, 2139
- to\_ulong
  - std::bitset, 2140
- tolower
  - std, 635
  - std::\_\_ctype\_abstract\_base, 1263, 1264
  - std::ctype, 2224, 2225
  - std::ctype< char >, 2238, 2239
  - std::ctype< wchar\_t >, 2260
  - std::ctype\_byname, 2275, 2276
  - std::ctype\_byname< char >, 2288, 2289
- top
  - std::priority\_queue, 2841
  - std::stack, 2919, 2920
- toupper
  - std, 636
  - std::\_\_ctype\_abstract\_base, 1264, 1265
  - std::ctype, 2225, 2226
  - std::ctype< char >, 2239, 2240
  - std::ctype< wchar\_t >, 2260, 2261
  - std::ctype\_byname, 2276, 2277
  - std::ctype\_byname< char >, 2289, 2290
- tr1\_math\_spec\_func
  - assoc\_laguerre, 107
  - assoc\_legendre, 107
  - beta, 107
  - comp\_ellint\_1, 107
  - comp\_ellint\_2, 108
  - comp\_ellint\_3, 108
  - conf\_hyperg, 108
  - cyl\_bessel\_i, 108
  - cyl\_bessel\_j, 108
  - cyl\_bessel\_k, 108
  - cyl\_neumann, 109
  - ellint\_1, 109
  - ellint\_2, 109
  - ellint\_3, 109
  - expint, 109
  - hermite, 110
  - hyperg, 110
  - laguerre, 110
  - legendre, 110
  - riemann\_zeta, 110
  - sph\_bessel, 110
  - sph\_legendre, 110
  - sph\_neumann, 111
- traits\_type
  - \_\_gnu\_cxx::enc\_filebuf, 822
  - \_\_gnu\_cxx::stdio\_filebuf, 890
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 922
  - std::basic\_filebuf, 1456
  - std::basic\_fstream, 1496, 1497
  - std::basic\_ifstream, 1560
  - std::basic\_ios, 1611
  - std::basic\_iostream, 1647
  - std::basic\_istream, 1708
  - std::basic\_istream::sentry, 1750
  - std::basic\_istreamstream, 1762
  - std::basic\_ofstream, 1815
  - std::basic\_ostream, 1858
  - std::basic\_ostreamstream, 1903
  - std::basic\_streambuf, 1941
  - std::basic\_stringbuf, 2023
  - std::basic\_stringstream, 2058, 2059
  - std::istreambuf\_iterator, 2512
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2806
- transform
  - mutating\_algorithms, 141, 142
  - std::collate, 2193
  - std::collate\_byname, 2199
- transform\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1156
- traps
  - std::\_\_numeric\_limits\_base, 1342
  - std::numeric\_limits, 2759
- tree\_policy.hpp, 3446
- tree\_trace\_base.hpp, 3447
- trie\_policy.hpp, 3447
- true\_type
  - metaprogramming, 117
- truename
  - std::numpunct, 2791
  - std::numpunct\_byname, 2797
- trunc
  - std::basic\_fstream, 1549
  - std::basic\_ifstream, 1602
  - std::basic\_ios, 1633

- std::basic\_iostream, 1697
- std::basic\_istream, 1748
- std::basic\_istringstream, 1804
- std::basic\_ofstream, 1848
- std::basic\_ostream, 1890
- std::basic\_ostringstream, 1935
- std::basic\_stringstream, 2110
- std::ios\_base, 2495
- try\_lock
  - mutexes, 67
- tuple, 3448
- Type Traits, 111
- type\_traits, 3450, 3452
- type\_traits.h, 3456
- type\_utils.hpp, 3456
- typeinfo, 3457
- typelist.h, 3458
- types.h, 3459
- types\_traits.hpp, 3460
- u16streampos
  - std, 554
- u16string
  - strings, 234
- u32streampos
  - std, 554
- u32string
  - strings, 234
- uflow
  - \_\_gnu\_cxx::enc\_filebuf, 837
  - \_\_gnu\_cxx::stdio\_filebuf, 909
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 937
  - std::basic\_filebuf, 1474
  - std::basic\_streambuf, 1956
  - std::basic\_stringbuf, 2039
- uncaught\_exception
  - exceptions, 33
- underflow
  - \_\_gnu\_cxx::enc\_filebuf, 837
  - \_\_gnu\_cxx::stdio\_filebuf, 910
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 937
  - std::basic\_filebuf, 1475
  - std::basic\_streambuf, 1956
  - std::basic\_stringbuf, 2040
- unexpected
  - exceptions, 33
- unexpected\_handler
  - exceptions, 31
- unget
  - std::basic\_fstream, 1540
  - std::basic\_ifstream, 1594
  - std::basic\_iostream, 1689
  - std::basic\_istream, 1740
  - std::basic\_istringstream, 1795
  - std::basic\_stringstream, 2101
- Uniform, 244
- uniform\_int\_distribution
  - std::uniform\_int\_distribution, 3013
- uniform\_real\_distribution
  - std::uniform\_real\_distribution, 3017
- uninitialized\_copy
  - std, 636
- uninitialized\_copy\_n
  - SGIextensions, 21
  - std, 636
- uninitialized\_fill
  - std, 637
- uninitialized\_fill\_n
  - std, 637
- unique
  - mutating\_algorithms, 143
  - std::forward\_list, 2385
  - std::list, 2553, 2554
- unique\_copy
  - mutating\_algorithms, 144
- unique\_copy.h, 3460
- unique\_copy\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1156
- unique\_ptr.h, 3461
- unitbuf
  - std, 637
  - std::basic\_fstream, 1549
  - std::basic\_ifstream, 1602
  - std::basic\_ios, 1633
  - std::basic\_iostream, 1698
  - std::basic\_istream, 1749
  - std::basic\_istringstream, 1804
  - std::basic\_ofstream, 1848
  - std::basic\_ostream, 1890
  - std::basic\_ostringstream, 1935
  - std::basic\_stringstream, 2110
  - std::ios\_base, 2495

- Unordered Associative, 26
- unordered\_map, 3462, 3463
- unordered\_map.h, 3465
- unordered\_set, 3467, 3468
- unordered\_set.h, 3469
- unsetf
  - std::basic\_fstream, 1540
  - std::basic\_ifstream, 1594
  - std::basic\_ios, 1625
  - std::basic\_iostream, 1689
  - std::basic\_istream, 1741
  - std::basic\_istream, 1796
  - std::basic\_ofstream, 1840
  - std::basic\_ostream, 1882
  - std::basic\_ostringstream, 1927
  - std::basic\_stringstream, 2102
  - std::ios\_base, 2488
- unshift
  - std::\_\_codecvt\_abstract\_base, 1250
  - std::codecvt, 2163
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2169
  - std::codecvt< char, char, mbstate\_t >, 2174
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2179
  - std::codecvt\_byname, 2186
- upper\_bound
  - binary\_search\_algorithms, 193, 194
  - std::map, 2601
  - std::multimap, 2681
  - std::multiset, 2699
  - std::set, 2892
- uppercase
  - std, 638
  - std::basic\_fstream, 1549
  - std::basic\_ifstream, 1602
  - std::basic\_ios, 1633
  - std::basic\_iostream, 1698
  - std::basic\_istream, 1749
  - std::basic\_istream, 1804
  - std::basic\_ofstream, 1848
  - std::basic\_ostream, 1890
  - std::basic\_ostringstream, 1935
  - std::basic\_stringstream, 2110
  - std::ios\_base, 2495
- use\_facet
  - std, 638
  - std::locale, 2561
  - std::locale::id, 2568
- Utilities, 72
- utility, 3471
- valarray, 3472
  - numeric\_arrays, 84–86
  - std::valarray, 3038
- valarray\_after.h, 3477
- valarray\_array.h, 3492
- valarray\_array.tcc, 3503
- valarray\_before.h, 3504
- value\_comp
  - std::map, 2601
  - std::multimap, 2681
  - std::multiset, 2700
  - std::set, 2893
- value\_compare
  - std::set, 2878
- value\_type
  - std::back\_insert\_iterator, 1444
  - std::complex, 2201
  - std::front\_insert\_iterator, 2391
  - std::insert\_iterator, 2474
  - std::istream\_iterator, 2509
  - std::istreambuf\_iterator, 2512
  - std::iterator, 2516
  - std::ostream\_iterator, 2801
  - std::ostreambuf\_iterator, 2806
  - std::raw\_storage\_iterator, 2859
  - std::reverse\_iterator, 2866
  - std::set, 2878
- vector, 3504, 3505
  - std::\_\_debug::vector, 1326
  - std::vector, 3042–3044
- vector.tcc, 3507
- vstring.h, 3507
- vstring.tcc, 3511
- vstring\_fwd.h, 3512
- vstring\_util.h, 3513
- wcerr
  - std, 640
- wcin

- std, 640
- wclog
  - std, 640
- wcout
  - std, 640
- wfilebuf
  - io, 58
- wfstream
  - io, 58
- what
  - \_\_gnu\_cxx::forced\_error, 847
  - \_\_gnu\_cxx::recursive\_init\_error, 872
  - std::bad\_alloc, 1446
  - std::bad\_cast, 1448
  - std::bad\_exception, 1449
  - std::bad\_function\_call, 1450
  - std::bad\_typeid, 1451
  - std::domain\_error, 2342
  - std::exception, 2350
  - std::future\_error, 2409
  - std::invalid\_argument, 2476
  - std::ios\_base::failure, 2496
  - std::length\_error, 2520
  - std::logic\_error, 2570
  - std::out\_of\_range, 2808
  - std::overflow\_error, 2810
  - std::range\_error, 2851
  - std::runtime\_error, 2872
  - std::system\_error, 2926
  - std::tr1::bad\_weak\_ptr, 2969
  - std::underflow\_error, 3011
- widen
  - std::\_\_ctype\_abstract\_base, 1265, 1266
  - std::basic\_fstream, 1541
  - std::basic\_ifstream, 1595
  - std::basic\_ios, 1626
  - std::basic\_iostream, 1689
  - std::basic\_istream, 1741
  - std::basic\_istream, 1796
  - std::basic\_ofstream, 1840
  - std::basic\_ostream, 1882
  - std::basic\_ostringstream, 1927
  - std::basic\_stringstream, 2102
  - std::ctype, 2226, 2227
  - std::ctype< char >, 2240, 2241
  - std::ctype< wchar\_t >, 2261, 2262
  - std::ctype\_byname, 2277, 2278
  - std::ctype\_byname< char >, 2290, 2291
- width
  - std::basic\_fstream, 1541
  - std::basic\_ifstream, 1595
  - std::basic\_ios, 1626
  - std::basic\_iostream, 1690
  - std::basic\_istream, 1742
  - std::basic\_istream, 1797
  - std::basic\_ofstream, 1840, 1841
  - std::basic\_ostream, 1883
  - std::basic\_ostringstream, 1928
  - std::basic\_stringstream, 2102, 2103
  - std::ios\_base, 2488
- wfstream
  - io, 59
- wios
  - io, 59
- wiostream
  - io, 59
- wistream
  - io, 59
- wistringstream
  - io, 59
- wofstream
  - io, 59
- workstealing.h, 3513
- wostream
  - io, 59
- wostringstream
  - io, 60
- write
  - std::basic\_fstream, 1542
  - std::basic\_iostream, 1691
  - std::basic\_ofstream, 1841
  - std::basic\_ostream, 1883
  - std::basic\_ostringstream, 1928
  - std::basic\_stringstream, 2103
- ws
  - std, 638
- wstreambuf
  - io, 60
- wstreampos

---

- std, 554
- wstring
  - strings, 234
- wstringbuf
  - io, 60
- wstringstream
  - io, 60
  
- xalloc
  - std::basic\_fstream, 1542
  - std::basic\_ifstream, 1596
  - std::basic\_ios, 1627
  - std::basic\_iostream, 1691
  - std::basic\_istream, 1742
  - std::basic\_istream, 1797
  - std::basic\_ofstream, 1842
  - std::basic\_ostream, 1884
  - std::basic\_ostringstream, 1929
  - std::basic\_stringstream, 2104
  - std::ios\_base, 2489
- xsgetn
  - \_\_gnu\_cxx::enc\_filebuf, 838
  - \_\_gnu\_cxx::stdio\_filebuf, 911
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 938
  - std::basic\_filebuf, 1475
  - std::basic\_streambuf, 1957
  - std::basic\_stringbuf, 2040
- xspn
  - \_\_gnu\_cxx::enc\_filebuf, 838
  - \_\_gnu\_cxx::stdio\_filebuf, 911
  - \_\_gnu\_cxx::stdio\_sync\_filebuf, 938
  - std::basic\_filebuf, 1476
  - std::basic\_streambuf, 1958
  - std::basic\_stringbuf, 2041
  
- yield
  - std::this\_thread, 698