

# ERRATA

---

09-02486

The attached pages update the following:

**CTIX Operating System Manual,**  
Version C, Volume 2,  
2nd Edition  
09-02263-01

**CTIX Operating System Manual,**  
Version C, Volume 3,  
2nd Edition  
09-02264-01

- The man page for *masterupd(1M)* should be removed from Volume 2, and all references to this command in the manuals should be ignored.
- The *route(1M)* man page included with this errata should be substituted for the *route(1M)* man page in Volume 2.
- Insert the *accept(2)* man page, included with this errata, in Section 2 of Volume 3, immediately after the *intro(1M)* man page.

—

—

—

**NAME**

route - manually manipulate the routing tables

**SYNOPSIS**

```
/etc/route [ -f ] [ command destination gateway [ metric ] ]
```

**DESCRIPTION**

*route* is a program used to manually (statically) manipulate the network routing tables. It is normally not needed, since the routing daemon, *routed* manages the system routing table and therefore handles this function. Static routing is generally appropriate in cases where packets are being sent to systems that do not recognize broadcast routing packets.

Only the super-user may modify routing tables.

*route* accepts two commands: *add*, to add a route; and *delete*, to delete a route.

All commands have the following syntax:

```
/etc/route command destination gateway [ metric ]
```

where *destination* is a host or network for which the route is "to," *gateway* is the gateway to which packets should be addressed, and *metric* indicates the "hop count," the number of interfaces a packet must pass through to reach its *destination*. If no metric is specified, *route* assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. All symbolic names specified for a destination or gateway are first looked up in the host name database. If this lookup fails, *route* looks for the name in the network name database.

If the *-f* option is specified, *route* will "flush" (delete) the routing tables of all gateway entries.

**DIAGNOSTICS**

*add* host: *gateway* host *flags* hex-flag

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *iocctl* call.

*delete* host: *gateway* host *flags* hex-flags

As above, but when deleting an entry.

*host host done*

When the **-f** flag is specified, each routing table entry is deleted. When a deletion is done by host name, an error is indicated with a message of this form.

*not in table*

A delete operation was attempted without the **-f** flag for a host name that is not present in the route table.

*routing table overflow*

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

**SEE ALSO**

intro(4), adman(1), routed(1M), hosts(4), networks(4).



**NAME**

accept - accept a connection on a socket

**SYNOPSIS**

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(s, addr, addrlen)
int s;
struct sockaddr *addr;
int *addrlen;
```

**DESCRIPTION**

The *accept* call accepts a connection on a socket. The argument *s* is a socket that has been created with *socket(2)*, bound to an address with *bind(2)*, and is listening for connections after a *listen(2)*. The *accept* extracts the first connection on the queue of pending connections, creates a new socket with the same properties of *s* and allocates a new file descriptor for the socket. If no pending connections are present on the queue, and the socket is not marked as non-blocking, *accept* blocks the caller until a connection is present. If the socket is marked non-blocking and no pending connections are present on the queue, *accept* returns an error as described below. The accepted socket, *ns*, cannot be used to accept more connections. The original socket *s* remains open.

The argument *addr* is a result parameter which is filled in with the address of the connecting entity, as known to the communications layer. The exact format of the *addr* parameter is determined by the "communications domain" [see *protocols(4)*]. The *addrlen* is a value-result parameter; it should initially contain the amount of space pointed to by *addr*; on return it contains the actual length (in bytes) of the address returned. This call is used with connection-based socket types, currently with *SOCK\_STREAM*.

**RETURN VALUE**

The call returns -1 on error. If it succeeds it returns a non-negative integer which is a descriptor for the accepted socket (*ns*, described above).

**ERRORS**

The *accept* will fail if:

**ACCEPT(2)**

**(CTIX Internetworking)**

**ACCEPT(2)**

[EBADF]	The descriptor is invalid.
[ENOTSOCK]	The descriptor references a file, not a socket.
[EOPNOTSUPP]	The referenced socket is not of type SOCK_STREAM.
[EFAULT]	The <i>addr</i> parameter is not in a writable part of the user address space.

**SEE ALSO**

bind(2), connect(2), intro(2), listen(2), socket(2), intro(7).  
*CTIX Network Programmer's Primer*.

The enclosed pages update the following manual:

**CTIX Operating System Manual,  
Version C  
Second Edition  
09-02263-01**

Insert the pages of this Update Notice into the *CTIX Operating System Manual, Version C* according to the instructions below. Each new page is dated so that you know which pages are update pages and which are original pages. The Summary of Changes on the first page of this Update Notice summarizes the changes made.

All material discussed in this Update Notice will be incorporated into the next edition of the *CTIX Operating System Manual, Version C*. To retain a record of this Update Notice, insert this cover page immediately after the manual's title page.

For Volume 1:

<b>Find and Remove:</b>	<b>Replace With:</b>
title page	title page
iii to vii	iii to viii.b
xxix to xciii	xxix to xciii
adb(1)	adb(1)
as(1)	as(1)
cc(1)	cc(1)
cc1sw(1)	cc1sw(1)
config(1M)	config(1M)
cpp(1)	cpp(1)
crash(1M)	crash(1M)
createdev(1M)	createdev(1M)
fsck(1M)	fsck(1M)
fstyp(1M)	fstyp(1M)
hinv(1)	hinv(1)
includes(1)	includes(1)
-----	iopdump(1M)
iv(1)	iv(1)

For Volume 2:

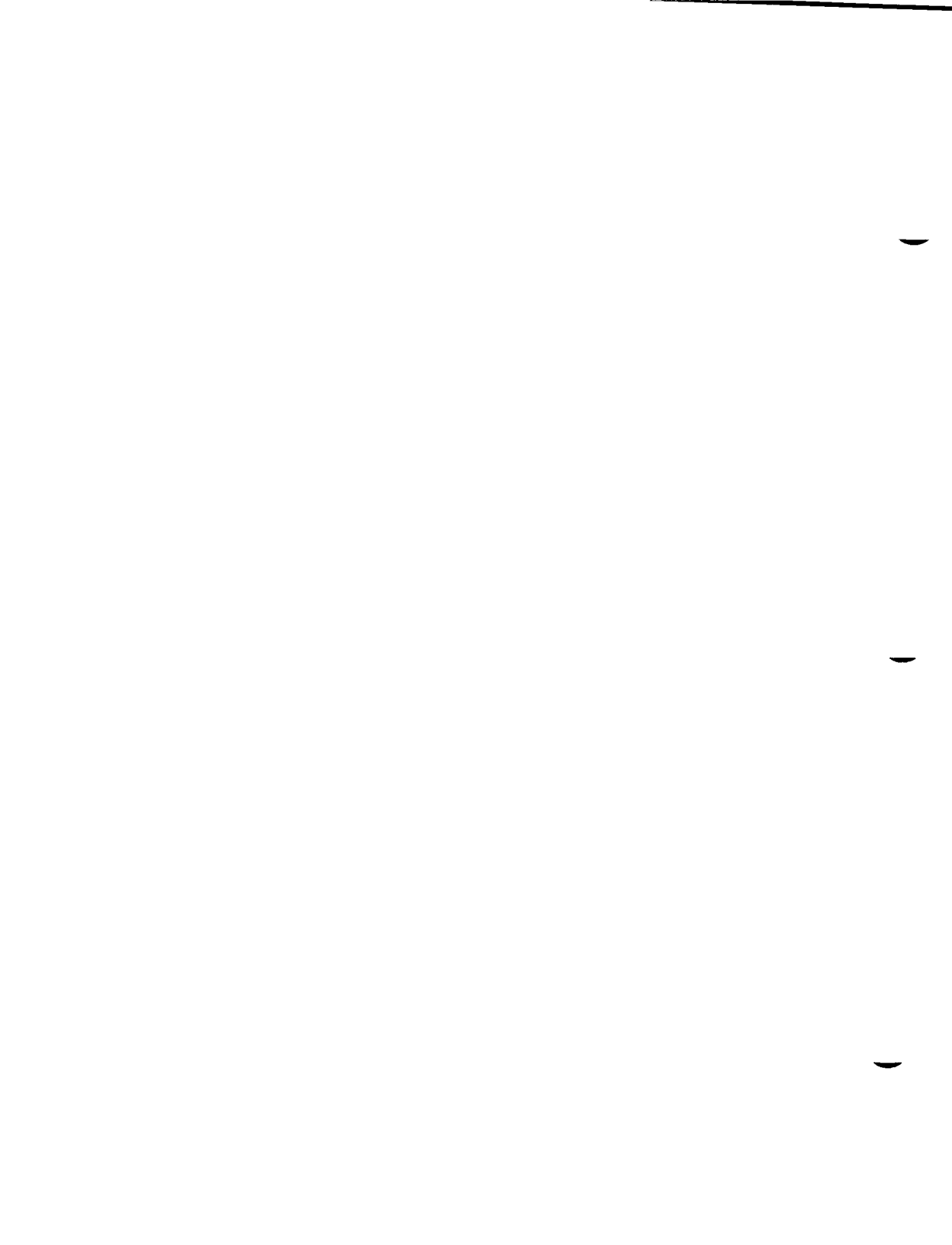
<b>Find and Remove:</b>	<b>Replace With:</b>
title page	title page
iii to vii	iii to vii
xiii to lxxvii	-----
machid(1)	machid(1)
masterupd(1M)	-----
mcs(1)	mcs(1)
mkfs(1M)	mkfs(1M)
mount(1M)	mount(1M)
-----	occ(1)
passmgmt(1M)	passmgmt(1M)
passwd(1)	passwd(1)
profiler(1M)	profiler(1M)
pwconv(1M)	pwconv(1M)
sar(1)	sar(1)
sdb(1)	sdb(1)
serstat(1M)	serstat(1M)
tio(1)	tio(1)

### For Volume 3:

<b>Find and Remove:</b>	<b>Replace With:</b>
title page	title page
iii to vii	iii to vii
xiii to lxxvii	-----
gettimeofday(2)	gettimeofday(2)
notify(2)	notify(2)
shmop(2)	shmop(2)
syslocal(2)	syslocal(2)
fpgetround(3C)	fpgetround(3C)
getspent(3X)	getspent(3X)
monitor(3C)	monitor(3C)
sleep(3C)	sleep(3C)

### For Volume 4:

<b>Find and Remove:</b>	<b>Replace With:</b>
title page	title page
iii to v	iii to v
xi to lxxv	-----
filehdr(4)	filehdr(4)
passwd(4)	passwd(4)
shadow(4)	-----
-----	timezone(4)
disk(7)	disk(7)



**CTIX™ OPERATING SYSTEM MANUAL**

**Version C  
Volume 1**

Convergent Technologies is a registered trademark of  
Convergent Technologies, Inc.

Convergent, CTIX, S/80, S/280, S/480, S/640, and S/4040 are trademarks of  
Convergent Technologies, Inc.

CTIX is derived from UNIX System V by Convergent Technologies under license from  
AT&T. UNIX and RFS are trademarks of AT&T.

Material excerpted from the UNIX System V, Release 3.2 *System Administrator's/User's  
Reference Manual* and *Programmer's Reference Manual* is Copyright 1989 by AT&T  
Technologies. Reprinted by permission.

This software and documentation is based in part on the Fourth Berkeley Software  
Distribution under license from the Regents of the University of California.

This manual was prepared on a Convergent Technologies S/640 Computer System and  
was printed on an Apple LaserWriter II Laser Printer.

Second Edition (November 1989) 09-02262-01  
Update Notice 1 (November 1990) 09-02578

Copyright © 1990 by Convergent Technologies, Inc.,  
San Jose, CA. Printed in USA.

All rights reserved. No part of this document may be reproduced, transmitted, stored in a  
retrieval system, or translated into any language without the prior written consent of  
Convergent Technologies, Inc.

Convergent Technologies makes no representations or warranties with respect to the  
contents hereof and specifically disclaims any implied warranties of merchantability or  
fitness for any particular purpose. Further, Convergent Technologies reserves the right  
to revise this publication and to make changes from time to time in its content without  
being obligated to notify any person of such revision or changes.



# TABLE OF CONTENTS: VOLUME 1

Summary of Changes .....	ix
Guide to Technical Documentation .....	xi
How to Use This Manual .....	xix
How to Get Started .....	xxiii
Permuted Index .....	xxix

## 1. Commands and Application Programs: A-L

Table of Related Entries . . . . .	Section 1
intro . . . . .	introduction to commands and application programs
300 . . . . .	handle special functions of DASI 300 and 300s terminals
4014 . . . . .	paginator for the Tektronix 4014 terminal
450 . . . . .	handle special functions of the DASI 450 terminal
Uutry . . . . .	try to contact a remote system with debugging on
accept . . . . .	allow or prevent LP requests
acct . . . . .	overview of accounting and miscellaneous accounting commands
acctems . . . . .	command summary from per-process accounting records
acctcom . . . . .	search and print process accounting file(s)
acctcon . . . . .	connect-time accounting
acctmrg . . . . .	merge or add total accounting files
acctprc . . . . .	process accounting
acctsh . . . . .	shell procedures for accounting
adb . . . . .	absolute debugger
adman . . . . .	administer a CTIX system
admin . . . . .	create and administer SCCS files
adv . . . . .	advertise a directory for remote access
ar . . . . .	archive and library maintainer for portable archives
arp . . . . .	address resolution display and control
as . . . . .	common assembler
asa . . . . .	interpret ASA carriage control characters
assist . . . . .	assistance using CTIX system commands
astgen . . . . .	generate/modify ASSIST menus and command forms
at . . . . .	execute commands at a later time
awk . . . . .	pattern scanning and processing language
banner . . . . .	make posters
basename . . . . .	deliver portions of path names
bc . . . . .	arbitrary-precision arithmetic language
bcheck . . . . .	print the list of blocks associated with an i-node(s)
bcopy . . . . .	interactive block copy
bdiff . . . . .	big diff
bfs . . . . .	big file scanner
brc . . . . .	system initialization procedures
bs . . . . .	a compiler/interpreter for modest-sized programs
cal . . . . .	print calendar
calendar . . . . .	reminder service
captainfo . . . . .	convert a termcap description into a terminfo description

cat . . . . . concatenate and print files  
cb . . . . . C program beautifier  
cc . . . . . C compiler  
cc1sw . . . . . front-end to the cc command  
cd . . . . . change working directory  
cdc . . . . . change the delta commentary of an SCCS delta  
cflow . . . . . generate C flowgraph  
chkshlib . . . . . compare shared libraries tool  
chmod . . . . . change mode  
chown . . . . . change owner or group  
chroot . . . . . change root directory for a command  
chrtbl . . . . . generate character classification and conversion tables  
ckbupscd . . . . . check file system backup schedule  
clear . . . . . clear terminal screen  
clri . . . . . clear i-node  
cmp . . . . . compare two files  
col . . . . . filter reverse line-feeds  
comb . . . . . combine SCCS deltas  
comm . . . . . select or reject lines common to two sorted files  
config . . . . . configure a CTIX system  
conlocate . . . . . locate a terminal to use as the virtual system console  
conv . . . . . common object file converter  
convert . . . . . convert archive files to common formats  
cp . . . . . copy, link, or move files  
cpio . . . . . copy file archives in and out  
cpp . . . . . the C language preprocessor  
cprs . . . . . compress a common object file  
cpsct . . . . . install object files in binary directories  
crash . . . . . examine system images  
createdev . . . . . create device nodes for assorted device types  
cron . . . . . clock daemon  
crontab . . . . . user crontab file  
crypt . . . . . encode/decode  
csh . . . . . a shell (command interpreter) with C-like syntax  
csplit . . . . . context split  
ct . . . . . spawn getty to a remote terminal  
ctags . . . . . create a tags file  
ctinstall . . . . . install software  
ctrace . . . . . C program debugger  
cu . . . . . call another UNIX system  
cut . . . . . cut out selected fields of each line of a file  
cw . . . . . prepare constant-width text for troff  
cxref . . . . . generate C program cross-reference  
date . . . . . print and set the date  
dbconsole . . . . . change the kernel debugger system console port  
dc . . . . . desk calculator  
dcopy . . . . . copy file systems for optimal access time  
dd . . . . . convert and copy a file  
delta . . . . . make a delta (change) to an SCCS file  
deroff . . . . . remove nroff/troff, tbl, and eqn constructs  
devnm . . . . . device name  
df . . . . . report number of free disk blocks and i-nodes

diff . . . . . differential file comparator  
diff3 . . . . . 3-way differential file comparison  
diffmk . . . . . mark differences between files  
dircmp . . . . . directory comparison  
dis . . . . . object code disassembler  
diskusg . . . . . generate disk accounting data by user ID  
dname . . . . . Print Remote File Sharing domain and network names  
du . . . . . summarize disk usage  
dump . . . . . dump selected parts of an object file  
echo . . . . . echo arguments  
ed . . . . . text editor  
edit . . . . . text editor (variant of ex for casual users)  
efl . . . . . extended FORTRAN language  
egrep . . . . . search a file for a pattern using full regular expressions  
enable . . . . . enable/disable LP printers  
enpstart . . . . . configure Ethernet processor  
env . . . . . set environment for command execution  
eqn . . . . . format mathematical text for nroff or troff  
errdead . . . . . extract error records and status information from dump  
errdemon . . . . . error-logging demon  
errpt . . . . . process a report of logged errors  
errstop . . . . . terminate the error-logging demon  
ex . . . . . text editor  
expand . . . . . expand tabs to spaces, and vice versa  
expr . . . . . evaluate arguments as an expression  
extproc . . . . . turn external processing on or off  
factor . . . . . obtain the prime factors of a number  
ff . . . . . list file names and statistics for a file system  
fgrep . . . . . search a file for a character string  
file . . . . . determine file type  
finc . . . . . fast incremental backup  
find . . . . . find files  
finger . . . . . user information lookup program  
fingerd . . . . . remote user information server  
fold . . . . . fold long lines for finite width output device  
frec . . . . . recover files from a backup tape  
fsck . . . . . check and repair file systems  
fsdb . . . . . file system debugger  
fsize . . . . . report file size  
fsplit . . . . . split FORTRAN, ratfor, or efl files  
fsstat . . . . . report file system status  
fstyp . . . . . determine file system identifier  
ftp . . . . . ARPANET file transfer program  
ftpd . . . . . DARPA Internet File Transfer Protocol server  
fumount . . . . . forced unmount of an advertised resource  
fusage . . . . . disk access profiler  
fuser . . . . . identify processes using a file or file structure  
fwtmp . . . . . manipulate connect accounting records  
gdev . . . . . graphical device routines and filters  
ged . . . . . graphical editor  
gencc . . . . . create a front-end to the cc command  
get . . . . . get a version of an SCCS file

getopt . . . . . parse command options  
getopts . . . . . parse command options  
getservaddr . . . . . get network address of service host  
getty . . . . . set terminal type, modes, speed, and line discipline  
glossary . . . . . definitions of common CTIX system terms and symbols  
graph . . . . . draw a graph  
graphics . . . . . access graphical and numerical commands  
greek . . . . . select terminal filter  
grep . . . . . search a file for a pattern  
gutil . . . . . graphical utilities  
hd . . . . . hexadecimal and ascii file dump  
head . . . . . give first few lines  
help . . . . . CTIX system Help Facility  
helpadm . . . . . make changes to the Help Facility database  
hinv . . . . . hardware inventory  
hostid . . . . . set or print identifier of current host system  
hostname . . . . . set or print the Internet host name of the current system  
hp . . . . . handle special functions of Hewlett-Packard terminals  
hpio . . . . . Hewlett-Packard 2645A terminal tape file archiver  
hyphen . . . . . find hyphenated words  
id . . . . . print user and group IDs and names  
idload . . . . . Remote File Sharing user and group mapping  
ifconfig . . . . . configure network interface parameters  
includes . . . . . determine C language preprocessor include files  
inetd . . . . . internet "super-server"  
infoemp . . . . . compare or print out terminfo descriptions  
init . . . . . process control initialization  
install . . . . . install commands  
iopdump . . . . . upload a front-end I/O processor's RAM  
ipcrm . . . . . remove a message queue, semaphore set or shared memory ID  
ipcs . . . . . report inter-process communication facilities status  
iv . . . . . initialize and maintain volume  
join . . . . . relational database operator  
kill . . . . . terminate a process  
killall . . . . . kill all active processes  
labelit . . . . . provide labels for file systems  
ld . . . . . link editor for common object files  
lddrv . . . . . manage loadable drivers  
ldeeprom . . . . . load EEPROM  
lex . . . . . generate programs for simple lexical tasks  
line . . . . . read one line  
link . . . . . link and unlink files and directories  
lint . . . . . a C program checker  
list . . . . . produce C source listing from a common object file  
locate . . . . . identify a CTIX system command using keywords  
login . . . . . sign on  
logname . . . . . get login name  
lorder . . . . . find ordering relation for an object library  
lp . . . . . send/cancel requests to an LP line printer  
lpadmin . . . . . configure the LP spooling system  
lpr . . . . . line printer spooler  
lpsched . . . . . start/stop the LP scheduler and move requests

lpset . . . . . set parallel line printer options  
lpstat . . . . . print LP status information  
ls . . . . . list contents of directory

—

—

—

## UPDATE NOTICE 1 SUMMARY OF CHANGES

*Update Notice 1* to the *CTIX Operating System Manual, Version C, Second Edition*, documents the new commands and features of the 6.3 release of CTIX, which runs on the new S/4040, a Motorola 68040-based system. Changes to the manual are summarized below.

### Volume 1

- Revised front matter
- New pages:  
*iopdump*(1M).
- Revised pages:  
*adb*(1), *as*(1), *cc*(1), *ccIsw*(1), *config*(1M), *cpp*(1), *crash*(1M), *createdev*(1M), *fsck*(1M), *fstyp*(1M), *hinvt*(1), *includes*(1), *iv*(1).

### Volume 2

- Revised front matter
- New pages:  
*occ*(1).
- Revised pages:  
*machid*(1), *mcs*(1), *mkfs*(1M), *mount*(1M), *passmgmt*(1M), *passwd*(1), *profiler*(1M), *pwconv*(1M), *sar*(1), *sdb*(1), *serstat*(1M), *tio*(1).
- Deleted pages:  
*masterupd*(1M).

### Volume 3

- Revised pages:  
*gettimeofday*(2), *notify*(2), *shmop*(2), *syslocal*(2), *fpggetround*(3C), *getspent*(3X), *monitor*(3C), *sleep*(3C).

## Volume 4

- Revised front matter
- New pages:  
*timezone(4)*.
- Revised pages:  
*filehdr(4), passwd(4), disk(7)*.
- Deleted pages:  
*shadow(4)*.



## PERMUTED INDEX

This index includes entries for all pages of Volumes 1 through 4. The entries themselves are based on the one-line descriptions or titles found in the NAME portion of each manual page; the significant words (keywords) of these descriptions are listed alphabetically down the center of the index.

The index is actually a keyword-in-context (KWIC) index that has three columns. To use the index, read the center column to look up specific commands by name or by subject topics. Note that the entry may begin in the left column or wrap around and continue into the left column. A period (.) marks the end of the entry, and a slash (/) indicates where the entry has been continued or truncated. The right column gives the manual page where the command or subject is described.

hpio: Hewlett-Packard	2645A terminal tape file/	hpio(1)
/special functions of DASI	300 and 300s terminals.	300(1)
for Interphase V/TAPE	3200 half-inch tape/ /interface	ipt(7)
l3tol, ltol3: convert between	3-byte integers and long/	l3tol(3C)
comparison. diff3:	3-way differential file	diff3(1)
paginator for the Tektronix	4014 terminal. 4014:	4014(1)
special functions of the DASI	450 terminal. 450: handle	450(1)
long integer and base-64/	a64l, l64a: convert between	a64l(3C)
	abort: generate a SIGABRT.	abort(3C)
	value. abs: return integer absolute	abs(3C)
	adb: absolute debugger.	adb(1)
abs: return integer	absolute value.	abs(3C)
/floor, ceiling, remainder,	absolute value functions.	floor(3M)
top: terminal	accelerator interface.	tiop(7)
t_accept:	accept a connect request.	t_accept(3n)
prevent LP requests.	accept, reject: allow or	accept(1M)
a directory for remote	access. adv: advertise	adv(1M)
of a file. touch: update	access and modification times	touch(1)
utime: set file	access and modification times.	utime(2)
accessibility of a file.	access: determine	access(2)
commands. graphics:	access graphical and numerical	graphics(1G)
sputl, sgetl:	access long integer data in a/	sputl(3X)
fusage: disk	access profiler.	fusage(1M)
sadp: disk	access profiler.	sadp(1M)
ldfcn: common object file	access routines.	ldfcn(4)
copy file systems for optimal	access time. dcopy:	dcopy(1M)
locking: exclusive	access to regions of a file.	locking(2)
/setutent, endutent, utmpname:	access utmp file entry.	getut(3C)
access: determine	accessibility of a file.	access(2)
enable or disable process	accounting. acct:	acct(2)
acctcon2: connect-time	accounting. acctcon1,	acctcon(1M)
acctprc1, acctprc2: process	accounting.	acctprc(1M)
turnacct: shell procedures for	accounting. /startup,	acctsh(1M)
/accton, acctwtmp: overview of	accounting and miscellaneous/	acct(1M)
accounting and miscellaneous	accounting commands. /of	acct(1M)
diskusg: generate disk	accounting data by user ID.	diskusg(1M)
acct: per-process	accounting file format.	acct(4)

search and print process	accounting file(s).	acctcom:	acctcom(1)
acctmrg: merge or add total	accounting files.		acctmrg(1M)
summary from per-process	accounting records. /command		acctcms(1M)
wtmpfix: manipulate connect	accounting records. fwtmp,		fwtmp(1M)
runacct: run daily	accounting.		runacct(1M)
process accounting.	acct: enable or disable		acct(2)
file format.	acct: per-process accounting		acct(4)
per-process accounting/	acctcms: command summary from		acctcms(1M)
process accounting file(s).	acctcom: search and print		acctcom(1)
connect-time accounting.	acctcon1, acctcon2:		acctcon(1M)
acctwtmp: overview of/	acctdisk, acctdusg, accton,		acct(1M)
accounting files.	acctmrg: merge or add total		acctmrg(1M)
accounting.	acctprc1, acctprc2: process		acctprc(1M)
orderly release/ t_rcvrel:	acknowledge receipt of an		t_rcvrel(3n)
trig: sin, cos, tan, asin,	acos, atan, atan2:/		trig(3M)
killall: kill all	active processes.		killall(1M)
sag: system	activity graph.		sag(1G)
sar: sa1, sa2, sadc: system	activity report package.		sar(1M)
sar: system	activity reporter.		sar(1)
current SCCS file editing	activity. sact: print		sact(1)
report process data and system	activity. /time a command;		timex(1)
Dialers:	ACU/modem calling protocols.		Dialers(5)
random, hopefully interesting,	adage. fortune: print a		fortune(6)
	adb: absolute debugger.		adb(1)
acctmrg: merge or	add total accounting files.		acctmrg(1M)
putenv: change or	add value to environment.		putenv(3C)
/inet_netof: Internet	address manipulation routines.		inet(3)
getservaddr: get network	address of service host.		getservad(1M)
control. arp:	address resolution display and		arp(1M)
arp:	Address Resolution Protocol.		arp(7)
endpoint. t_bind: bind an	address to a transport		t_bind(3n)
allow synchronization of the/	adjtime: correct the time to		adjtime(2)
system.	adman: administer a CTIX		adman(1)
SCCS files.	admin: create and administer		admin(1)
network listener service	administration. nlsadmin:		nlsadmin(1M)
rfadmin: Remote File Sharing	administration.		rfadmin(1M)
uadmin:	administrative control.		uadmin(1M)
uadmin:	administrative control.		uadmin(2)
swap: swap	administrative interface.		swap(1M)
remote access.	adv: advertise a directory for		adv(1M)
	advert: explore Colossal Cave.		advert(6)
remote access. adv:	advertise a directory for		adv(1M)
fumount: forced unmount of an	advertised resource.		fumount(1M)
alarm: set a process	alarm clock.		alarm(2)
aliases:	aliases file for sendmail.		aliases(4)
the data base for the mail	aliases file. /rebuild		newaliases(1)
t_alloc:	allocate a library structure.		t_alloc(3n)
change data segment space	allocation. brk, sbrk:		brk(2)
realloc, calloc: main memory	allocator. malloc, free,		malloc(3C)
mallinfo: fast main memory	allocator. /calloc, mallopt,		malloc(3X)
accept, reject:	allow or prevent LP requests.		accept(1M)
adjtime: correct the time to	allow synchronization of the/		adjtime(2)
process by changing/ renice:	alter priority of running		renice(1)
sort: sort	and/or merge files.		sort(1)
link editor output.	a.out: common assembler and		a.out(4)
introduction to commands and	application programs. intro:		intro(1)
maintainer for portable/	ar: archive and library		ar(1)
format.	ar: common archive file		ar(4)

number: convert	Arabic numerals to English.	number(6)
language. bc:	arbitrary-precision arithmetic	bc(1)
for portable archives. ar:	archive and library maintainer	ar(1)
cpio: format of cpio	archive.	cpio(4)
ar: common	archive file format.	ar(4)
header of a member of an	archive file. /the archive	ldahread(3X)
formats. convert: convert	archive files to common	convert(1)
an archive/ ldahread: read the	archive header of a member of	ldahread(3X)
2645A terminal tape file	archiver. /Hewlett-Packard	hpio(1)
tar: tape file	archiver.	tar(1)
maintainer for portable	archives. /archive and library	ar(1)
cpio: copy file	archives in and out.	cpio(1)
varargs: handle variable	argument list.	varargs(5)
formatted output of a varargs	argument list. /print	vprintf(3S)
command. xargs: construct	argument list(s) and execute	xargs(1)
getopt: get option letter from	argument vector.	getopt(3C)
expr: evaluate	arguments as an expression.	expr(1)
echo: echo	arguments.	echo(1)
bc: arbitrary-precision	arithmetic language.	bc(1)
number facts.	arithmetic: provide drill in	arithmetic(6)
display and control.	arp: address resolution	arp(1M)
Protocol.	arp: Address Resolution	arp(7)
ftp:	ARPANET file transfer program.	ftp(1)
expr: evaluate arguments	as an expression.	expr(1)
/attach and detach serial lines	as: common assembler.	as(1)
/locate a terminal to use	as network interfaces.	slattach(1M)
characters. asa: interpret	as the virtual system console.	conlocate(1M)
and/ /gmtime, asctime, cftime,	ASA carriage control	asa(1)
ascii: map of	asctime, tzset: convert date	ctime(3C)
hd: hexadecimal and	ASCII character set.	ascii(5)
long integer and base-64	ascii file dump.	hd(1)
strings: extract the	ASCII string. /convert between	a64l(3C)
ctime, localtime, gmtime,	ASCII text strings in a file.	strings(1)
trig: sin, cos, tan,	asctime, cftime, asctime/	ctime(3C)
output. a.out: common	asin, acos, atan, atan2:/	trig(3M)
as: common	assembler and link editor	a.out(4)
assertion.	assembler.	as(1)
setbuf, setvbuf:	assert: verify program	assert(3X)
astgen: generate/modify	assign buffering to a stream.	setbuf(3S)
commands. assist:	ASSIST menus and command/	astgen(1)
print the list of blocks	assistance using CTIX system	assist(1)
/create device nodes for	associated with an. bcheck:	bcheck(1M)
menus and command forms.	assorted device types.	createdev(1M)
a later time.	astgen: generate/modify ASSIST	astgen(1)
/sin, cos, tan, asin, acos,	at, batch: execute commands at	at(1)
description file. queuedefs:	atan, atan2: trigonometric/	trig(3M)
double-precision/ strtod,	at/batch/cron queue	queuedefs(4)
integer. strtol, atol,	atof: convert string to	strtod(3C)
integer. strtol,	atoi: convert string to	strtol(3C)
as/ slattach, sldetach:	atol, atoi: convert string to	strtol(3C)
resources. mnnttry:	attach and detach serial lines	slattach(1M)
log of failed login	attempt to mount remote	mnnttry(1M)
wait:	attempts. /usr/adm/loginlog:	loginlog(4)
processing language.	await completion of process.	wait(1)
ungetc: push character	awk: pattern scanning and	awk(1)
back: the game of	back into input stream.	ungetc(3S)
finc: fast incremental	backgammon.	back(6)
	backup.	finc(1M)

ckbupscd: check file system backup schedule. . . . .	ckbupscd(1M)
frec: recover files from a backup tape. . . . .	frec(1M)
banner: make posters. . . . .	banner(1)
newaliases: rebuild the data base for the mail aliases/ . . . . .	newaliases(1)
Sun rpc program number data base. rpc: . . . . .	rpc(4)
terminal capability data base. termcap: . . . . .	termcap(4)
terminal capability data base. terminfo: . . . . .	terminfo(4)
between long integer and base-64 ASCII string. /convert . . . . .	a64l(3C)
(visual) display editor based on ex. /screen-oriented . . . . .	vi(1)
from proto file; set links based on. /out file lists . . . . .	qlist(1)
portions of path names. basename, dimame: deliver . . . . .	basename(1)
later time. at, batch: execute commands at a . . . . .	at(1)
arithmetic language. bc: arbitrary-precision . . . . .	bc(1)
blocks associated with an. bcheck: print the list of . . . . .	bcheck(1M)
system initialization/ brc, bcheckrc, drvload, powerfail: . . . . .	brc(1M)
string operations. bcopy, bcmp, bzero: bit and byte . . . . .	bstring(3)
byte string operations. bcopy, bcmp, bzero: bit and . . . . .	bstring(3)
bcopy: interactive block copy. . . . .	bcopy(1M)
bdiff: big diff. . . . .	bdiff(1)
cb: C program beautifier. . . . .	cb(1)
about the operating system for beginning users. /information . . . . .	starter(1)
j0, j1, jn, y0, y1, yn: Bessel functions. bessell: . . . . .	bessel(3M)
cpset: install object files in binary directories. . . . .	cpset(1M)
fread, fwrite: binary input/output. . . . .	fread(3S)
bsearch: binary search a sorted table. . . . .	bsearch(3C)
tfind, tdelete, twalk: manage binary search trees. tsearch, . . . . .	tsearch(3C)
bind: bind a name to a socket. . . . .	bind(2)
endpoint. t_bind: bind an address to a transport . . . . .	t_bind(3n)
nfsd, biod: NFS daemons. . . . .	nfsd(1M)
bcopy, bcmp, bzero: bit and byte string/ . . . . .	bstring(3)
bj: the game of black jack. . . . .	bj(6)
bcopy: interactive block copy. . . . .	bcopy(1M)
sum: print checksum and block count of a file. . . . .	sum(1)
sync: update the super block. . . . .	sync(1M)
sync: update super block. . . . .	sync(2)
df: report number of free disk blocks and i-nodes. . . . .	df(1M)
bcheck: print the list of blocks associated with an. . . . .	bcheck(1M)
libdev: manipulate Volume Home Blocks (VHB). . . . .	libdev(3X)
powerfail: system/ brc, bcheckrc, drvload, . . . . .	brc(1M)
space allocation. brk, sbrk: change data segment . . . . .	brk(2)
modest-sized programs. bs: a compiler/interpreter for . . . . .	bs(1)
sorted table. bsearch: binary search a . . . . .	bsearch(3C)
stdio: standard buffered input/output package. . . . .	stdio(3S)
setbuf, setvbuf: assign buffering to a stream. . . . .	setbuf(3S)
mknod: build special file. . . . .	mknod(1M)
vme: VME bus interface. . . . .	vme(7)
between host and network byte order. /convert values . . . . .	byteorder(3)
bcopy, bcmp, bzero: bit and byte string operations. . . . .	bstring(3)
size: print section sizes in bytes of common object files. . . . .	size(1)
swab: swap bytes. . . . .	swab(3C)
operations. bcopy, bcmp, bzero: bit and byte string . . . . .	bstring(3)
cc: C compiler. . . . .	cc(1)
cflow: generate C flowgraph. . . . .	cflow(1)
cpp: the C language preprocessor. . . . .	cpp(1)
include/ includes: determine C language preprocessor . . . . .	includes(1)
cb: C program beautifier. . . . .	cb(1)
lint: a C program checker. . . . .	lint(1)

cxref: generate	C program cross-reference.	cxref(1)
ctrace:	C program debugger.	ctrace(1)
extract and share strings in	C programs.	xstr(1)
time. cprofile: setting up a	C shell environment at login	cprofile(4)
object file. list: produce	C source listing from a common	list(1)
	cal: print calendar.	cal(1)
dc: desk	calculator.	dc(1)
cal: print	calendar.	cal(1)
	calendar: reminder service.	calendar(1)
cu:	call another UNIX system.	cu(1C)
data returned by stat system	call. stat:	stat(5)
Dialers: ACU/modem	calling protocols.	Dialers(5)
malloc, free, realloc,	calloc: main memory allocator.	malloc(3C)
fast/ malloc, free, realloc,	calloc, malloc, mallinfo:	malloc(3X)
intro: introduction to system	calls and error numbers.	intro(2)
common shared NFS system	calls. nfssys:	nfssys(2)
request. rumount:	cancel queued remote resource	rumount(1M)
to an LP line printer. lp,	cancel: send/cancel requests	lp(1)
termcap: terminal	capability data base.	termcap(4)
terminfo: terminal	capability data base.	terminfo(4)
description into a terminfo/	captainfo: convert a termcap	captainfo(1M)
asa: interpret ASA	carriage control characters.	asa(1)
text editor (variant of ex for	casual users). edit:	edit(1)
files.	cat: concatenate and print	cat(1)
advent: explore Colossal	Cave.	advent(6)
	cb: C program beautifier.	cb(1)
	cc: C compiler.	cc(1)
cc2sw, cc2fp: front-end to the	cc command. cc1sw,	cc1sw(1)
create a front-end to the	cc command. gcccc:	gcccc(1M)
to the cc command.	cc1sw, cc2sw, cc2fp: front-end	cc1sw(1)
	cd: change working directory.	cd(1)
commentary of an SCCS delta.	cdc: change the delta	cdc(1)
/ceil, fmod, fabs: floor,	ceiling, remainder, absolute/	floor(3M)
	cflow: generate C flowgraph.	cflow(1)
/localtime, gmtime, asctime,	cftime, ascftime, tzset:/	ctime(3C)
strings.	cftime: language specific	cftime(4)
delta: make a delta	(change) to an SCCS file.	delta(1)
priority of running process by	changing nice. renice: alter	renice(1)
pipe: create an interprocess	channel.	pipe(2)
terminal's local RS-232	channels. tp: controlling	tp(7)
stream. ungetc: push	character back into input	ungetc(3S)
conversion/ chrtbl: generate	character classification and	chrtbl(1M)
and neqn. eqnchar: special	character definitions for eqn	eqnchar(5)
_toupper, setchrclass:	character handling. /_tolower,	ctype(3C)
user. cuserid: get	character login name of the	cuserid(3S)
/getchar, fgetc, getw: get	character or word from a/	getc(3S)
/putchar, fputc, putw: put	character or word on a stream.	putc(3S)
ascii: map of ASCII	character set.	ascii(5)
fgrep: search a file for a	character string.	fgrep(1)
interpret ASA carriage control	characters. asa:	asa(1)
_tolower, toascii: translate	characters. /_toupper,	conv(3C)
tr: translate	characters.	tr(1)
lastlogin, monacct, nulladm,/	chargefee, ckpacct, dodisk,	acctsh(1M)
directory.	chdir: change working	chdir(2)
fsock, dfsock:	check and repair file systems.	fsock(1M)
schedule. ckbupscd:	check file system backup	ckbupscd(1M)
permissions file. uuchekc:	check the uucp directories and	uuchekc(1M)
constant-width text for/ cw,	checkcw: prepare	checkcw(1)

text for nroff or/	eqn, neqn,	checkeq: format mathematical	eqn(1)
lint: a C program		checker.	lint(1)
grpck: password/group file		checkers. pwck,	pwck(1M)
systems processed by fsck and/		checklist: list of file	checklist(4)
formatted with the MM/ mm,		checkmm: print/check documents	mm(1)
file. sum: print		checksum and block count of a	sum(1)
chown,		chgrp: change owner or group.	chown(1)
times: get process and		child process times.	times(2)
terminate. wait: wait for		child process to stop or	wait(2)
libraries tool.		chkshlib: compare shared	chkshlib(1)
		chmod: change mode.	chmod(1)
		chmod: change mode of file.	chmod(2)
of a file.		chown: change owner and group	chown(2)
group.		chown, chgrp: change owner or	chown(1)
		chroot: change root directory.	chroot(2)
for a command.		chroot: change root directory	chroot(1M)
classification and conversion/		chrtbl: generate character	chrtbl(1M)
backup schedule.		ckbupscd: check file system	ckbupscd(1M)
monacct, nulladm,/ chargefee,		ckpacct, dodisk, lastlogin,	acctsh(1M)
chrtbl: generate character		classification and conversion/	chrtbl(1M)
strclean: STREAMS error logger		cleanup program.	strclean(1M)
uucp spool directory		clean-up. uucleanup:	uucleanup(1M)
		clri: clear i-node.	clri(1M)
		clear: clear terminal screen.	clear(1)
status/ ferror, feof,		clearerr, fileno: stream	ferror(3S)
the listener. nlsgetcall: get		client's data passed through	nlsgetcall(3n)
(command interpreter) with		C-like syntax. csh: a shell	csh(1)
synchronization of the system		clock. /the time to allow	adjtime(2)
alarm: set a process alarm		clock.	alarm(2)
		clock daemon.	cron(1M)
		clock: report CPU time used.	clock(3C)
on a STREAMS driver.		clone: open any minor device	clone(7)
ldclose, ldaclose:		close a common object file.	ldclose(3X)
close:		close a file descriptor.	close(2)
t_close:		close a transport endpoint.	t_close(3n)
felose, fflush:		close or flush a stream.	felose(3S)
telldir, seekdir, rewinddir,		closedir: directory/ /readdir,	directory(3X)
		clri: clear i-node.	clri(1M)
		cmp: compare two files.	cmp(1)
dis: object		code disassembler.	dis(1)
line-feeds.		col: filter reverse	col(1)
advent: explore		Colossal Cave.	advent(6)
comb:		combine SCCS deltas.	comb(1)
common to two sorted files.		comm: select or reject lines	comm(1)
nice: run a		command at low priority.	nice(1)
cc2fp: front-end to the cc		command. cc1sw, cc2sw,	cc1sw(1)
change root directory for a		command. chroot:	chroot(1M)
examples. usage: retrieve a		command description and usage	usage(1)
env: set environment for		command execution.	env(1)
rcmd: remote shell		command execution.	rcmd(1)
uux: UNIX-to-UNIX system		command execution.	uux(1C)
/ASSIST menus and		command forms.	astgen(1)
create a front-end to the cc		command. gence:	gence(1M)
quits. nohup: run a		command immune to hangups and	nohup(1)
C-like syntax. csh: a shell		(command interpreter) with	csh(1)
getopt: parse		command options.	getopt(1)
getopts, getoptcv: parse		command options.	getopts(1)
locate executable file for		command. path:	path(1)

/shell, the standard/restricted	command programming language.	sh(1)
returning a stream to a remote	command. /routines for	rcmd(3)
and system/ timex: time a	command; report process data	timex(1)
uuxqt: execute remote	command requests.	uuxqt(1M)
return stream to a remote	command. rexec:	rexec(3)
per-process/ acctcms:	command summary from	acctcms(1M)
system: issue a shell	command.	system(3S)
used by the /etc/tapeset	command. /information	tapedrives(4)
test: condition evaluation	command.	test(1)
time: time a	command.	time(1)
locate: identify a CTIX system	command using keywords.	locate(1)
argument list(s) and execute	command. xargs: construct	xargs(1)
and miscellaneous accounting	commands. /of accounting	acct(1M)
intro: introduction to	commands and application/	intro(1)
assistance using CTIX system	commands. assist:	assist(1)
at, batch: execute	commands at a later time.	at(1)
access graphical and numerical	commands. graphics:	graphics(1G)
install: install	commands.	install(1M)
mkhosts: make node name	commands.	mkhosts(1M)
multi-user/ rc2, rc3: run	commands performed for	rc2(1M)
operating system. rc0: run	commands performed to stop the	rc0(1M)
network useful with graphical	commands. stat: statistical	stat(1G)
streamio: STREAMS ioctl	commands.	streamio(7)
manipulate the object file	comment section. mcs:	mcs(1)
cdc: change the delta	commentary of an SCCS delta.	cdc(1)
ar:	common archive file format.	ar(4)
editor output. a.out:	common assembler and link	a.out(4)
as:	common assembler.	as(1)
glossary: definitions of	common CTIX system terms and/	glossary(1)
convert archive files to	common formats. convert:	convert(1)
routines. ldfcn:	common object file access	ldfcn(4)
conv:	common object file converter.	conv(1)
cprs: compress a	common object file.	cprs(1)
ldopen, ldaopen: open a	common object file for/	ldopen(3X)
/line number entries of a	common object file function.	ldread(3X)
ldclose, ldaclose: close a	common object file.	ldclose(3X)
read the file header of a	common object file. ldfhread:	ldfhread(3X)
entries of a section of a	common object file. /number	ldlseek(3X)
the optional file header of a	common object file. /seek to	ldohseek(3X)
/entries of a section of a	common object file.	ldrseek(3X)
/section header of a	common object file.	ldshread(3X)
an indexed/named section of a	common object file. /seek to	ldsseek(3X)
of a symbol table entry of a	common object file. /the index	ldtbindex(3X)
symbol table entry of a	common object file. /indexed	ldtbread(3X)
seek to the symbol table of a	common object file. ldtbseek:	ldtbseek(3X)
line number entries in a	common object file. linenum:	linenum(4)
C source listing from a	common object file. /produce	list(1)
nm: print name list of	common object file.	nm(1)
relocation information for a	common object file. reloc:	reloc(4)
scnhdr: section header for a	common object file.	scnhdr(4)
line number information from a	common object file. /and	strip(1)
/retrieve symbol name for	common object file symbol/	ldgetname(3X)
table format. syms:	common object file symbol	syms(4)
filehdr: file header for	common object files.	filehdr(4)
ld: link editor for	common object files.	ld(1)
section sizes in bytes of	common object files. /print	size(1)
calls. nfssys:	common shared NFS system	nfssys(2)
comm: select or reject lines	common to two sorted files.	comm(1)

ipcs: report inter-process communication facilities/	ipcs(1)
/ftok: standard interprocess communication package.	stdipc(3C)
talkd: remote user communication server.	talkd(1M)
socket: create an endpoint for communication.	socket(2)
/configuration file for uucp communications lines.	Devices(5)
diff: differential file comparator.	diff(1)
descriptions. infocmp: compare or print out terminfo	infocmp(1M)
chkshlib: compare shared libraries tool.	chkshlib(1)
cmp: compare two files.	cmp(1)
SCCS file. sccsdiff: compare two versions of an	sccsdiff(1)
diff3: 3-way differential file comparison.	diff3(1)
dircmp: directory comparison.	dircmp(1)
expression. regcmp, regex: compile and execute regular	regcmp(3X)
regexp: regular expression compile and match routines.	regexp(5)
regcmp: regular expression compile.	regcmp(1)
term: format of compiled term file..	term(4)
cc: C compiler.	cc(1)
tic: terminfo compiler.	tic(1M)
yacc: yet another compiler-compiler.	yacc(1)
modest-sized programs. bs: a compiler/interpreter for	bs(1)
erf, erfc: error function and complementary error function.	erf(3M)
wait: await completion of process.	wait(1)
cprs: compress a common object file.	cprs(1)
pack, pcat, unpack: compress and expand files.	pack(1)
table entry of a/ ldtbodyindex: compute the index of a symbol	ldtbodyindex(3X)
cat: concatenate and print files.	cat(1)
test: condition evaluation command.	test(1)
system. config: configure a CTIX configuration file.	config(1M)
NFS file systems export exports: . exports(4)	exports(4)
(internet/ inetd.conf: configuration file for inetd	inetd.conf(4)
communications/ Devices: configuration file for uucp	Devices(5)
gateways: routed configuration file.	gateways(4)
netcf: Network Configuration File.	netcf(4)
resolv.conf: resolver configuration file.	resolver(4)
STREAMS linker, load socket configuration. /ldsocket:	slink(1)
rtab: Remote I/O Processor configuration table.	rtab(4)
config: configure a CTIX system.	config(1M)
enpstart: configure Ethernet processor.	enpstart(1M)
parameters. ifconfig: configure network interface	ifconfig(1M)
I/O Processor. riopcfg: configure system for Remote	riopcfg(1M)
system. lpadmin: configure the LP spooling	lpadmin(1M)
system. uconf: configure the operating	uconf(1M)
t_rcvconnect: receive the confirmation from a connect/	t_rcvconnect(3)
to use as the virtual system/ conlocate: locate a terminal	conlocate(1M)
fwtmp, wumppfix: manipulate connect accounting records.	fwtmp(1M)
on a socket. connect: initiate a connection	connect(2)
t_accept: accept a connect request.	t_accept(3n)
t_listen: listen for a connect request.	t_listen(3n)
the confirmation from a connect request. /receive	t_rcvconnect(3)
getpeername: get name of connected peer.	getpeername(2)
an out-going terminal line connection. dial: establish	dial(3C)
connect: initiate a connection on a socket.	connect(2)
down part of a full-duplex connection. shutdown: shut	shutdown(2)
or expedited data sent over a connection. /receive data	t_rcv(3n)
data or expedited data over a connection. t_snd: send	t_snd(3n)
t_connect: establish a connection with another/	t_connect(3n)
listen: listen for connections on a socket.	listen(2)
acctcon1, acctcon2: connect-time accounting.	acctcon(1M)



to use as the virtual system console.	/locate a terminal	conlocate(1M)
the kernel debugger system console:	console port. /change	dbconsole(1M)
	console terminal.	console(7)
for implementation-specific constants.	/file header	limits(4)
math: math functions and constants.	unistd:	math(5)
file header for symbolic constants.	unistd:	unistd(4)
cw, checkcw: prepare constant-width text for troff.		cw(1)
mkfs: construct a file system.		mkfs(1M)
execute command. xargs: construct argument list(s) and constructs.	deroff: remove	xargs(1)
nroff/troff, tbl, and eqn debugging on. Uutry: try to contact a remote system with		deroff(1)
ls: list contents of directory.		Uutry(1M)
ttoc, vtoc: graphical table of contents routines.	toc: dtoc,	ls(1)
		toc(1G)
	csplit:	csplit(1)
address resolution display and control.	arp:	arp(1M)
asa: interpret ASA carriage control characters.		asa(1)
	ioctl: control device.	ioctl(2)
	control device.	scsi(7)
Serial Line Internet Protocol control facility.	/switched	slipd(1M)
fcntl: file control.		fcntl(2)
floating point environment control.	/fpsetsticky: IEEE	fpgetround(3)
init, telinit: process control initialization.		init(1M)
icmp: Internet Control Message Protocol.		icmp(7)
msgctl: message control operations.		msgctl(2)
semctl: semaphore control operations.		semctl(2)
shmctl: shared memory control operations.		shmctl(2)
	fcntl: file control options.	fcntl(5)
tcp: Internet Transmission Control Protocol.		tcp(7)
uadmin: administrative control.		uadmin(1M)
uadmin: administrative control.		uadmin(2)
uucp status inquiry and job control.	uustat:	uustat(1C)
vc: version control.		vc(1)
V/TAPE 3200 half-inch tape controller.	/for Interphase	ipt(7)
set drive parameters for tape controllers.	tapeset:	tapeset(1M)
interface. tty: controlling terminal		tty(7)
RS-232 channels. tp: controlling terminal's local		tp(7)
converter. conv: common object file		conv(1)
_toupper, _tolower, toascii: conv: toupper, tolower,		conv(3C)
terminals. term: conventional names for		term(5)
units: conversion program.		units(1)
character classification and conversion tables.	/generate	chrtbl(1M)
into a terminfo/ captainfo: convert a termcap description		captainfo(1M)
dd: convert and copy a file.		dd(1M)
English. number: convert Arabic numerals to		number(6)
common formats. convert: convert archive files to		convert(1)
integers and/ l3tol, ltol3: convert between 3-byte		l3tol(3C)
and base-64 ASCII/ a64l, l64a: convert between long integer		a64l(3C)
to common formats. convert: convert archive files		convert(1)
/ctime, ascftime, tzset: convert date and time to/		ctime(3C)
to string. ecvt, fcvt, gcvt: convert floating-point number		ecvt(3C)
scanf, fscanf, sscanf: convert formatted input.		scanf(3S)
strtod, atof: convert string to/		strtod(3C)
strtol, atol, atoi: convert string to integer.		strtol(3C)
htonl, htons, ntohl, ntohs: convert values between host/		byteorder(3)
conv: common object file converter.		conv(1)
timod: Transport Interface cooperating STREAMS module.		timod(7)
dd: convert and copy a file.		dd(1M)
bcopy: interactive block copy.		bcopy(1M)

	cpio:	copy file archives in and out.	cpio(1)
	access time.	copy file systems for optimal	dcopy(1M)
	cp, ln, mv:	copy, link, or move files.	cp(1)
	volcopy:	make literal copy of file system.	volcopy(1M)
	rep:	remote file copy.	rcp(1)
uname:	UNIX-to-UNIX system	copy.	uucp(1C)
	UNIX-to-UNIX system file	copy.	uuto, uupick: public
	core:	format of core image file.	core(4)
	synchronization of/	adjtime:	correct the time to allow
	atan2:/	trig: sin, cos, tan, asin, acos, atan,	trig(3M)
	functions.	sinh, cosh, tanh: hyperbolic	sinh(3M)
sum:	print checksum and block	count of a file.	sum(1)
	wc:	word count.	wc(1)
	move files.	cp, ln, mv: copy, link, or	cp(1)
	cpio:	format of cpio archive.	cpio(4)
	and out.	cpio: copy file archives in	cpio(1)
	preprocessor.	cpp: the C language	cpp(1)
environment at login time.	file.	cpprofile: setting up a C shell	cpprofile(4)
	binary directories.	cpvs: compress a common object	cpvs(1)
	clock: report	CPU time used.	clock(3C)
	craps:	the game of craps.	craps(6)
	rewrite an existing one.	crash: examine system images.	crash(1M)
	command.	creat: create a new file or	creat(2)
	gencc:	create a front-end to the cc	gencc(1M)
file.	tmpnam, tempnam:	create a name for a temporary	tmpnam(3S)
	an existing one.	creat: create a new file or rewrite	creat(2)
	fork:	create a new process.	fork(2)
	mkshlib:	create a shared library.	mkshlib(1)
	ctags:	create a tags file.	ctags(1)
	tmpfile:	create a temporary file.	tmpfile(3S)
communication.	socket:	create an endpoint for	socket(2)
	channel.	pipe: create an interprocess	pipe(2)
	files.	admin: create and administer SCCS	admin(1)
assorted device/	createdev:	create device nodes for	createdev(1M)
	umask:	set and get file creation mask.	umask(2)
	crontab:	user cron: clock daemon.	cron(1M)
	cxref:	generate C program crontab file.	crontab(1)
	pg:	file perusal filter for CRTs.	pg(1)
	encryption functions.	crypt: encode/decode.	crypt(1)
	generate hashing encryption.	crypt: password and file	crypt(3X)
	interpreter) with C-like/	crypt, setkey, encrypt:	crypt(3C)
	terminal.	csd: a shell (command	csd(1)
	for terminal.	esplit: context split.	esplit(1)
	asctime, cftime, ascftime/	ct: spawn getty to a remote	ct(1C)
	adman: administer a	ctags: create a tags file.	ctags(1)
	config: configure a	ctermid: generate file name	ctermid(3S)
uname: get name of current	/definitions of common	ctime, localtime, gmtime,	ctime(3C)
		ctinstall: install software.	ctinstall(1)
		CTIX system.	adman(1)
		CTIX system.	config(1M)
		CTIX system.	uname(2)
		CTIX system terms and/	glossary(1)
		ctrace: C program debugger.	ctrace(1)
		cu: call another UNIX system.	cu(1C)
		ttt: tic-tac-toe.	ttt(6)
	uname: get name of	current CTIX system.	uname(2)

endpoint.	t_look: look at the	current event on a transport	t_look(3n)
	get/set unique identifier of	current host. /sethostid:	gethostid(2)
sethostname:	get/set name of	current host. gethostname,	gethostname(2)
	set or print identifier of	current host system. hostid:	hostid(1)
	uname: print name of	current CTIX system.	uname(1)
	activity. sact: print	current SCCS file editing	sact(1)
	t_getstate: get the	current state.	t_getstate(3)
the Internet host name of the		current system. /set or print	hostname(1)
slot in the utmp file of the		current user. /find the	ttyslot(3C)
getcwd: get path-name of		current working directory.	getcwd(3C)
scr_dump: format of		curses screen image file..	scr_dump(4)
handling and optimization/		curses: terminal screen	curses(3X)
spline: interpolate smooth		curve.	spline(1G)
name of the user.		cuserid: get character login	cuserid(3S)
each line of a file. cut:		cut out selected fields of	cut(1)
constant-width text for/		cw, checkcw: prepare	cw(1)
cross-reference.		cxref: generate C program	cxref(1)
	cron: clock	daemon.	cron(1M)
rfudaemon: Remote File Sharing		daemon process.	rfudaemon(1M)
routed: network routing		daemon.	routed(1M)
strerr: STREAMS error logger		daemon.	strerr(1M)
nfsd, biod: NFS		daemons.	nfsd(1M)
runacct: run		daily accounting.	runacct(1M)
Protocol server. ftpd:		DARPA Internet File Transfer	ftpd(1M)
number mapper. portmap:		DARPA port to RPC program	portmap(1M)
telnetd:		DARPA TELNET protocol server.	telnetd(1M)
tftp: user interface to the		DARPA TFTP protocol.	tftp(1)
Protocol server. tftpd:		DARPA Trivial File Transfer	tftpd(1M)
/handle special functions of		DASI 300 and 300s terminals.	300(1)
special functions of the		DASI 450 terminal. /handle	450(1)
/time a command; report process		data and system activity.	timex(1)
file. newaliases: rebuild the		data base for the mail aliases	newaliases(1)
rpc: Sun rpc program number		data base.	rpc(4)
termcap: terminal capability		data base.	termcap(4)
terminfo: terminal capability		data base.	terminfo(4)
generate disk accounting		data by user ID. diskusg:	diskusg(1M)
t_rcvuderr: receive a unit		data error indication.	t_rcvuderr(3)
/sgetl: access long integer		data in a machine-independent/	spuul(3X)
plock: lock process, text, or		data in memory.	plock(2)
connection. t_snd: send		data or expedited data over a	t_snd(3n)
over a/ t_rcv: receive		data or expedited data sent	t_rcv(3n)
nlsgetcall: get client's		data passed through the/	nlsgetcall(3n)
prof: display profile		data.	prof(1)
call. stat:		data returned by stat system	stat(5)
I/O Processor for online		data. riopqry: query Remote	riopqry(1M)
brk, sbrk: change		data segment space allocation.	brk(2)
/receive data or expedited		data sent over a connection.	t_rcv(3n)
types: primitive system		data types.	types(5)
t_rcvudata: receive a		data unit.	t_rcvudata(3)
t_sndudata: send a		data unit.	t_sndudata(3)
changes to the Help Facility		database. helpadm: make	helpadm(1M)
join: relational		database operator.	join(1)
using the mkfs(1) proto file		database. /and verify software	qinstall(1)
delete, firstkey, nextkey:		database subroutines. /store,	dbm(3X)
/dbm_error, dbm_clearerr:		database subroutines.	ndbm(3X)
a terminal or query terminfo		database. tput: initialize	tput(1)
udp: Internet User		Datagram Protocol.	udp(7)
settimeofday: get/set		date and time. gettimeofday,	gettimeofday(2)

/ascftime, tzset: convert	date and time to string. . . . .	time(3C)
	date: print and set the date. . . . .	date(1)
debugger system console port.	dbconsole: change the kernel . . . . .	dbconsole(1M)
/dbm_nextkey, dbm_error,	dbm_clearerr: database/ . . . . .	ndbm(3X)
dbm_store,/ dbm_open,	dbm_close, dbm_fetch, . . . . .	ndbm(3X)
/dbm_fetch, dbm_store,	dbm_delete, dbm_firstkey,/ . . . . .	ndbm(3X)
/dbm_firstkey, dbm_nextkey,	dbm_error, dbm_clearerr:/ . . . . .	ndbm(3X)
dbm_open, dbm_close,	dbm_fetch, dbm_store,/ . . . . .	ndbm(3X)
/dbm_store, dbm_delete,	dbm_firstkey, dbm_nextkey,/ . . . . .	ndbm(3X)
firstkey, nextkey: database/	dbm_init, fetch, store, delete, . . . . .	dbm(3X)
/dbm_delete, dbm_firstkey,	dbm_nextkey, dbm_error,/ . . . . .	ndbm(3X)
dbm_fetch, dbm_store,/	dbm_open, dbm_close, . . . . .	ndbm(3X)
/dbm_close, dbm_fetch,	dbm_store, dbm_delete,/ . . . . .	ndbm(3X)
	dc: desk calculator. . . . .	dc(1)
optimal access time.	dcopy: copy file systems for . . . . .	dcopy(1M)
	dd: convert and copy a file. . . . .	dd(1M)
adb: absolute	debugger. . . . .	adb(1)
ctrace: C program	debugger. . . . .	ctrace(1)
fsdb: file system	debugger. . . . .	fsdb(1M)
load symbols in kernel	debugger. mkdbsym: . . . . .	mkdbsym(1M)
sdb: symbolic	debugger. . . . .	sdb(1)
dbconsole: change the kernel	debugger system console port. . . . .	dbconsole(1M)
contact a remote system with	debugging on. Uutry: try to . . . . .	Uutry(1M)
timezone: set	default system time zone. . . . .	timezone(4)
sysdef: output system	definition. . . . .	sysdef(1M)
eqnchar: special character	definitions for eqn and neqn. . . . .	eqnchar(5)
system terms and/ glossary:	definitions of common CTIX . . . . .	glossary(1)
dbm_init, fetch, store,	delete, firstkey, nextkey:/ . . . . .	dbm(3X)
names. basename, dirname:	deliver portions of path . . . . .	basename(1)
file. tail:	deliver the last part of a . . . . .	tail(1)
file. delta: make a	delta (change) to an SCCS . . . . .	delta(1)
delta. cdc: change the	delta commentary of an SCCS . . . . .	cdc(1)
rmdel: remove a	delta from an SCCS file. . . . .	rmdel(1)
to an SCCS file.	delta: make a delta (change) . . . . .	delta(1)
comb: combine SCCS	deltas. . . . .	comb(1)
errdemon: error-logging	demon. . . . .	errdemon(1M)
terminate the error-logging	demon. errstop: . . . . .	errstop(1M)
msg: permit or	deny messages. . . . .	msg(1)
tbl, and eqn constructs.	deroff: remove nroff/troff, . . . . .	deroff(1)
usage: retrieve a command	description and usage/ . . . . .	usage(1)
description into a terminfo	description. /a termcap . . . . .	captoinfo(1M)
queuedefs: at/batch/cron queue	description file. . . . .	queuedefs(4)
system: system	description file. . . . .	system(4)
captoinfo: convert a termcap	description into a terminfo/ . . . . .	captoinfo(1M)
compare or print out terminfo	descriptions. infocmp: . . . . .	infocmp(1M)
close: close a file	descriptor. . . . .	close(2)
dup: duplicate an open file	descriptor. . . . .	dup(2)
dup2: duplicate an open file	descriptor. . . . .	dup2(3C)
getdtablesize: get	descriptor table size. . . . .	getdtablesize(2)
dc:	desk calculator. . . . .	dc(1)
slattach, sldetach: attach and	detach serial lines as network/ . . . . .	slattach(1M)
file. access:	determine accessibility of a . . . . .	access(2)
preprocessor/ includes:	determine C language . . . . .	includes(1)
identifier. fstyp:	determine file system . . . . .	fstyp(1M)
file:	determine file type. . . . .	file(1)
drivers: loadable	device drivers. . . . .	drivers(7)
lines for finite width output	device. fold: fold long . . . . .	fold(1)
master: master	device information table. . . . .	master(4)

	ioctl: control device.	ioctl(2)
	devnm: device name.	devnm(1M)
device/ createdev: create	device nodes for assorted	createdev(1M)
clone: open any minor	device on a STREAMS driver.	clone(7)
/tekset, td: graphical	device routines and filters.	gdev(1G)
scsi: scsi control	device.	scsi(7)
device nodes for assorted	device types. /create	createdev(1M)
for uucp communications/	Devices: configuration file	Devices(5)
scsimap: set mappings for SCSI	devices.	scsimap(1M)
	devnm: device name.	devnm(1M)
blocks and i-nodes.	df: report number of free disk	df(1M)
systems. fsck.	dfsck: check and repair file	fsck(1M)
terminal line connection.	dial: establish an out-going	dial(3C)
ratfor: rational FORTRAN	dialect.	ratfor(1)
protocols.	Dialers: ACU/modem calling	Dialers(5)
bdiff: big	diff.	bdiff(1)
comparison.	diff3: 3-way differential file	diff3(1)
sdiff: side-by-side	difference program.	sdiff(1)
diffmk: mark	differences between files.	diffmk(1)
diff:	differential file comparator.	diff(1)
	dir: format of directories.	dir(4)
	dircmp: directory comparison.	dircmp(1)
file. uucheck: check the uucp	directories and permissions	uucheck(1M)
install object files in binary	directories. cpset:	cpset(1M)
dir: format of	directories.	dir(4)
link and unlink files and	directories. link, unlink:	link(1M)
mkdir, makedirs: make	directories.	mkdir(1)
rm, rmdir: remove files or	directories.	rm(1)
cd: change working	directory.	cd(1)
chdir: change working	directory.	chdir(2)
chroot: change root	directory.	chroot(2)
uucleanup: uucp spool	directory clean-up.	uucleanup(1M)
dircmp:	directory comparison.	dircmp(1)
file. getdents: read	directory entries and put in a	getdents(2)
file system independent	directory entry. dirent:	dirent(4)
unlink: remove	directory entry.	unlink(2)
chroot: change root	directory for a command.	chroot(1M)
/make a lost+found	directory for fsck.	mklostfnd(1M)
adv: advertise a	directory for remote access.	adv(1M)
path-name of current working	directory. getcwd: get	getcwd(3C)
ls: list contents of	directory.	ls(1)
mkdir: make a	directory.	mkdir(2)
mvdire: move a	directory.	mvdire(1M)
pwd: working	directory name.	pwd(1)
/seekdir, rewinddir, closedir:	directory operations.	directory(3X)
ordinary file. mknod: make a	directory, or a special or	mknod(2)
rmdir: remove a	directory.	rmdir(2)
independent directory entry.	dirent: file system	dirent(4)
path names. basename,	dirname: deliver portions of	basename(1)
	dis: object code disassembler.	dis(1)
t_unbind:	disable a transport endpoint.	t_unbind(3n)
printers. enable,	disable: enable/disable LP	enable(1)
acct: enable or	disable process accounting.	acct(2)
dis: object code	disassembler.	dis(1)
type, modes, speed, and line	discipline. /set terminal	getty(1M)
type, modes, speed, and line	discipline. /set terminal	uugetty(1M)
t_snddis: send user-initiated	disconnect request.	t_snddis(3n)
retrieve information from	disconnect. t_rcvdis:	t_rcvdis(3n)

fusage:	disk access profiler.	fusage(1M)
sadb:	disk access profiler.	sadb(1M)
ID. diskusg:	disk accounting data by user	diskusg(1M)
df: report number of free	disk blocks and i-nodes.	df(1M)
disk: general	disk driver.	disk(7)
update: provide	disk synchronization.	update(1M)
du: summarize	disk usage.	du(1M)
accounting data by user ID.	diskusg: generate disk	diskusg(1M)
arp: address resolution	display and control.	arp(1M)
vi: screen-oriented (visual)	display editor based on ex.	vi(1)
information. mntstat:	display mounted resource	mntstat(1M)
prof:	display profile data.	prof(1)
statistics. serstat:	display serial port error	serstat(1M)
local network. ruptime:	display status of nodes on	ruptime(1)
hypot: Euclidean	distance function.	hypot(3M)
/lcong48: generate uniformly	distributed pseudo-random/	drand48(3C)
Sharing domain and network/	dnname: print Remote File	dnname(1M)
routines. /res_send, res_init,	dn_comp, dn_expand: resolver	resolver(3)
/res_send, res_init, dn_comp,	dn_expand: resolver routines.	resolver(3)
MM/ mm, checkmm: print/check	documents formatted with the	mm(1)
macro package for formatting	documents. mm: the MM	mm(5)
slides. mmt, mvt: typeset	documents, view graphs, and	mmt(1)
nulladm,/ chargefee, ckpacct,	dodisk, lastlogin, monacct,	acctsh(1M)
whodo: who is	doing what.	whodo(1M)
/print Remote File Sharing	domain and network names.	dnname(1M)
named: Internet	domain name server.	named(1M)
/atof: convert string to	double-precision number.	strod(3C)
gtdl, ptdl: RS-232 terminal	download. tdl,	tdl(1)
nrand48, mrand48, jrand48,/	drand48, erand48, lrand48,	drand48(3C)
graph:	draw a graph.	graph(1G)
arithmetic: provide	drill in number facts.	arithmetic(6)
controllers. tapeset: set	drive parameters for tape	tapeset(1M)
used by the/ tapedrives: tape	drive specific information	tapedrives(4)
facilitate usage of a tape	drive. tsioctl:	tsioctl(1)
any minor device on a STREAMS	driver. clone: open	clone(7)
disk: general disk	driver.	disk(7)
lddrv: manage loadable	drivers.	lddrv(1M)
drivers.	drivers: loadable device	drivers(7)
initialization/ brc, bcheckrc,	drvload, powerfail: system	brc(1M)
table of contents/ toc:	dtoc, toc, vtoc: graphical	toc(1G)
and status information from	du: summarize disk usage.	du(1M)
hd: hexadecimal and ascii file	dump. /extract error records	errdead(1M)
od: octal	dump.	hd(1)
object file. dump:	dump selected parts of an	od(1)
descriptor.	dump: duplicate an open file	dump(1)
descriptor.	dup2: duplicate an open file	dup(2)
echo:	dup2: duplicate an open file	dup2(3C)
network/ ping: send ICMP	echo arguments.	echo(1)
floating-point number to/	ECHO_REQUEST packets to	ping(1M)
program. end, etext,	ecvt, fcvt, gcvt: convert	ecvt(3C)
ex for casual users).	ed, red: text editor.	ed(1)
sact: print current SCCS file	edata: last locations in	end(3C)
/(visual) display	edit: text editor (variant of	edit(1)
ed, red: text	editing activity.	sact(1)
ex: text	editor based on ex.	vi(1)
files. ld: link	editor.	ed(1)
	editor.	ex(1)
	editor for common object	ld(1)

ged: graphical editor.	ged(1G)
common assembler and linker output. a.out:	a.out(4)
sed: stream editor.	sed(1)
casual users). edit: text editor (variant of ex for	edit(1)
ldeeprom: load EEPROM.	ldeeprom(1M)
/user, real group, and effective group IDs.	getuid(2)
and/ /getegid: get real user, effective user, real group, language.	getuid(2)
split FORTRAN, ratfor, or efl: extended FORTRAN	efl(1)
pattern using full regular/ efl files. fsplit:	fsplit(1)
enable/disable LP printers. egrep: search a file for a	egrep(1)
accounting. acct: Ethernet Processor.	en(7)
real-time priorities. enable, disable:	enable(1)
enable, disable: enable or disable process	acct(2)
crypt: encode/decode.	rtcnable(1M)
encrypt: generate hashing encryption functions.	enable(1)
crypt: password and file encryption functions.	crypt(1)
makekey: generate encryption key.	crypt(3C)
locations in program. end, etext, edata: last	crypt(3X)
/getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group/	makekey(1)
/gethostent, sethostent, endhostent: get network host/	end(3C)
/getnetbyname, setnetent, endnetent: get network entry.	endgrent(3C)
socket: create an endpoint for communication.	gethostbyname(3)
bind an address to a transport endpoint. t_bind:	getnetent(3)
t_close: close a transport endpoint.	socket(2)
current event on a transport endpoint. t_look: look at the	t_bind(3n)
t_open: establish a transport endpoint.	t_close(3n)
manage options for a transport endpoint. t_open:	t_look(3n)
t_unbind: disable a transport endpoint. t_optmgmt:	t_open(3n)
/getprotobyname, setprotoent, endprotoent: get protocol/	t_optmgmt(3n)
/getpwuid, getpwnam, setpwent, endpwent, fgetpwent: get/	t_unbind(3n)
/getservbyname, setservent, endservent: get service entry.	getprotoent(3)
getspent, getspnam, setspent, endspent, fgetspent, lckpwwdf,/	getpwent(3C)
utmp/ /pututline, setutent, endutent, utmpname: access	getservent(3)
convert Arabic numerals to English. number:	getspent(3X)
processor. enpstart: configure Ethernet	getut(3C)
getdents: read directory entries and put in a file.	number(6)
nlist: get entries from name list.	enpstart(1M)
file. linenum: line number entries and put in a file.	getdents(2)
/manipulate line number entries in a common object	nlist(3C)
/ldlseek: seek to line number entries of a common object	linenum(4)
/ldnrseek: seek to relocation entries of a section of a/	ldlread(3X)
system independent directory entry. dirent: file	ldlseek(3X)
utmp, wtmp: utmp and wtmp entry formats.	ldrseek(3X)
fgetgrent: get group file entry. /setgrent, endgrent,	dirent(4)
endhostent: get network host entry. /sethostent,	utmp(4)
endnetent: get network entry. /setnetent,	getgrent(3C)
endprotoent: get protocol entry. /setprotoent,	gethostbyname(3)
fgetpwent: get password file entry. /setpwent, endpwent,	getnetent(3)
getrpcbyname: get rpc entry. /setrpcbyname,	getprotoent(3)
endservent: get service entry. /setservent,	getpwent(3C)
utmpname: access utmp file entry. /setutent, endutent,	getrpcent(3)
object file symbol table entry. /symbol name for common	getservent(3)
/the index of a symbol table entry of a common object file.	getut(3C)
/read an indexed symbol table entry of a common object file.	ldgetname(3X)
putpwent: write password file entry.	ldtbind(3X)
write shadow password file entry. putspent:	ldtbread(3X)
	putpwent(3C)
	putspent(3X)

unlink: remove directory	entry. . . . .	unlink(2)
command execution.	env: set environment for . . . . .	env(1)
	environ: user environment. . . . .	environ(5)
cprofile: setting up a C shell	environment at login time. . . . .	cprofile(4)
profile: setting up an	environment at login time. . . . .	profile(4)
/IEEE floating point	environment control. . . . .	fpgetround(3)
environ: user	environment. . . . .	environ(5)
execution. env: set	environment for command . . . . .	env(1)
getenv: return value for	environment name. . . . .	getenv(3C)
putenv: change or add value to	environment. . . . .	putenv(3C)
performed for multi-user	environment. /run commands . . . . .	rc2(1M)
stop the Remote File Sharing	environment. rfstop: . . . . .	rfstop(1M)
interface, and terminal	environment. /terminal . . . . .	tset(1)
character definitions for	eqn and neqn. /special . . . . .	eqnchar(5)
remove nroff/troff, tbl, and	eqn constructs. deroff: . . . . .	deroff(1)
mathematical text for nroff/	eqn, neqn, checkeq: format . . . . .	eqn(1)
definitions for eqn and neqn.	eqnchar: special character . . . . .	eqnchar(5)
rhosts: remote	equivalent users. . . . .	rhosts(4)
mrnd48, jrnd48, drand48,	erand48, lrnd48, nrnd48, . . . . .	drand48(3C)
graphical device/ gdev: hpd,	erase, hardcopy, tekset, td: . . . . .	gdev(1G)
complementary error function.	erf, erfc: error function and . . . . .	erf(3M)
	err: error-logging interface. . . . .	err(7)
and status information from/	errdead: extract error records . . . . .	errdead(1M)
format.	errdemon: error-logging demon. . . . .	errdemon(1M)
system error/ perror,	errfile: error-log file . . . . .	errfile(4)
function and complementary	errno, sys_errlist, sys_nerr: . . . . .	perror(3C)
receive a unit data	error function. /erfc: error . . . . .	erf(3M)
strclean: STREAMS	error indication. t_rcvuderr: . . . . .	t_rcvuderr(3)
strerr: STREAMS	error logger cleanup program. . . . .	strclean(1M)
log: interface to STREAMS	error logger daemon. . . . .	strerr(1M)
t_error: produce	error logging and event/ . . . . .	log(7)
sys_errlist, sys_nerr: system	error message. . . . .	t_error(3n)
to system calls and	error messages. /errno, . . . . .	perror(3C)
information/ errdead: extract	error numbers. /introduction . . . . .	intro(2)
serstat: display serial port	error records and status . . . . .	errdead(1M)
matherr:	error statistics. . . . .	serstat(1M)
errfile:	error-handling function. . . . .	matherr(3M)
errdemon:	error-log file format. . . . .	errfile(4)
errstop: terminate the	error-logging demon. . . . .	errdemon(1M)
err:	error-logging demon. . . . .	errstop(1M)
process a report of logged	error-logging interface. . . . .	err(7)
hashcheck: find spelling	errors. erprt: . . . . .	erprt(1M)
error-logging demon.	errors. /hashmake, spellin, . . . . .	spell(1)
another transport/ t_connect:	errstop: terminate the . . . . .	errstop(1M)
endpoint. t_open:	establish a connection with . . . . .	t_connect(3n)
terminal line/ dial:	establish a transport . . . . .	t_open(3n)
setmnt:	establish an out-going . . . . .	dial(3C)
with information from	establish mount table. . . . .	setmnt(1M)
with information from	/etc/passwd. /etc/shadow . . . . .	pwconv(1M)
pwconv: install and update	/etc/passwd. /etc/shadow . . . . .	pwunconv(1M)
pwunconv: install and update	/etc/shadow with information/ . . . . .	pwconv(1M)
/information used by the	/etc/shadow with information/ . . . . .	pwunconv(1M)
in program. end,	/etc/tapeset command. . . . .	tapedrives(4)
en:	etext, edata: last locations . . . . .	end(3C)
enpstart: configure	Ethernet Processor. . . . .	en(7)
hypot:	Ethernet processor. . . . .	enpstart(1M)
expression. expr:	Euclidean distance function. . . . .	hypot(3M)
	evaluate arguments as an . . . . .	expr(1)



test: condition	evaluation command.	test(1)
t_look: look at the current	event on a transport endpoint.	t_look(3n)
to STREAMS error logging and	event tracing. log: interface	log(7)
notify, unnotify, evwait,	evnowait: manage/	notify(2)
notify, unnotify,	evwait, evnowait: manage/	notify(2)
edit: text editor (variant of	ex for casual users).	edit(1)
	ex: text editor.	ex(1)
display editor based on	ex. /screen-oriented (visual)	vi(1)
crash:	examine system images.	crash(1M)
a file. locking:	exclusive access to regions of	locking(2)
execve, execlp, execl, execlp:/	exec: execl, execl, execl, execl,	exec(2)
execlp, execlp: execute/ exec:	execl, execl, execl, execl,	exec(2)
execlp:/ exec: execl, execl,	execl, execl, execl, execl,	exec(2)
/execl, execl, execl, execl,	execlp, execlp: execute a/	exec(2)
path: locate	executable file for command.	path(1)
execve, execlp, execlp:	execute a file. /execl,	exec(2)
construct argument list(s) and	execute command. xargs:	xargs(1)
time. at, batch:	execute commands at a later	at(1)
regcmp, regex: compile and	execute regular expression.	regcmp(3X)
requests. uuxqt:	execute remote command	uuxqt(1M)
set environment for command	execution. env:	env(1)
sleep: suspend	execution for an interval.	sleep(1)
sleep: suspend	execution for interval.	sleep(3C)
monitor: prepare	execution profile.	monitor(3C)
rcmd: remote shell command	execution.	rcmd(1)
rexecd: remote	execution server.	rexecd(1M)
profil:	execution time profile.	profil(2)
UNIX-to-UNIX system command	execution. uux:	uux(1C)
execlp: execute/ exec: execl,	execl, execl, execl, execl,	exec(2)
exec: execl, execl, execl,	execl, execl, execlp:/	exec(2)
/execl, execl, execl, execl,	execlp: execute a file.	exec(2)
a new file or rewrite an	existing one. creat: create	creat(2)
exit,	_exit: terminate process.	exit(2)
exponential, logarithm/	exp, log, log10, pow, sqrt:	exp(3M)
pcat, unpack: compress and	expand files. pack,	pack(1)
to spaces, and vice versa.	expand, unexpand: expand tabs	expand(1)
t_snd: send data or	expedited data over a/	t_snd(3n)
t_rcv: receive data or	expedited data sent over a/	t_rcv(3n)
advent:	explore Colossal Cave.	advent(6)
exp, log, log10, pow, sqrt:	exponential, logarithm, power,/	exp(3M)
exports: NFS file systems	export configuration file.	exports(4)
export configuration file.	exports: NFS file systems	exports(4)
expression.	expr: evaluate arguments as an	expr(1)
routines. regexp: regular	expression compile and match	regexp(5)
regcmp: regular	expression compile.	regcmp(1)
expr: evaluate arguments as an	expression.	expr(1)
compile and execute regular	expression. regcmp, regex:	regcmp(3X)
a pattern using full regular	expressions. /a file for	egrep(1)
efl:	extended FORTRAN language.	efl(1)
extproc: tum	external processing on or off.	extproc(1M)
programs. xstr:	extract and share strings in C	xstr(1)
status information/ errdead:	extract error records and	errdead(1M)
in a file. strings:	extract the ASCII text strings	strings(1)
remainder./ floor, ceil, fmod,	fabs: floor, ceiling,	floor(3M)
drive. tsioctl:	facilitate usage of a tape	tsioc(1)
factor: obtain the prime	factors of a number.	factor(1)
/usr/adm/loginlog: log of	failed login attempts.	loginlog(4)
true,	false: provide truth values.	true(1)

data in a machine-independent fashion.	/access long integer	sputl(3X)
	finc: fast incremental backup.	finc(1M)
/calloc, malloc, mallinfo:	fast main memory allocator.	malloc(3X)
a stream.	fclose, fflush: close or flush	fclose(3S)
	fcntl: file control.	fcntl(2)
	fcntl: file control options.	fcntl(5)
floating-point number/	fcvt, gcvt: convert	ecvt(3C)
ecvt,	fdopen: open a stream.	fopen(3S)
fopen, freopen,	feof, clearerr, fileno: stream	feof(3S)
status inquiries.	feof, clearerr,	feof(3S)
error,	fileno: stream status/	error(3S)
firstkey, nextkey:/	fetch, store, delete,	dbm(3X)
dbm: init,	ff: file names and statistics	ff(1M)
for a file system.	fflush: close or flush a	fclose(3S)
stream.	fgetc, getw: get character or	getc(3S)
fclose,	fgetgrent: get group file/	getgrent(3C)
word from a/	fgetpwent: get password file/	getpwent(3C)
getc, getchar,	fgets: get a string from a	gets(3S)
/getgrnam, setgrent, endgrent,	fgetspent, lckpwwdf, ulckpwwdf:/	getspent(3X)
/getpwnam, setpwent, endpwent,	fgrep: search a file for a	fgrep(1)
stream.	gets,	utime(2)
gets,	file access and modification	ldfcn(4)
/getspnam, setspent, endspent,	character string.	file access routines.
character string.	times.	file access:
times.	utime: set	access(2)
ldfcn: common object	ldfcn: common object	file archiver.
determine accessibility of a	file.	hpio(1)
/2645A terminal tape	file archiver.	tar(1)
tar: tape	cpio: copy	file archives in and out.
cpio: copy	pwck, grpck: password/group	file checkers.
pwck, grpck: password/group	chmod: change mode of	file.
chmod: change mode of	change owner and group of a	chown(2)
change owner and group of a	mcs: manipulate the object	file. chown:
mcs: manipulate the object	diff: differential	file comment section.
diff: differential	diff3: 3-way differential	file comparator.
diff3: 3-way differential	fcntl:	file comparison.
fcntl:	fcntl:	file control.
fcntl:	conv: common object	file control options.
conv: common object	rep: remote	file converter.
rep: remote	public UNIX-to-UNIX system	file copy.
public UNIX-to-UNIX system	core: format of core image	file copy. uuto, uupick:
core: format of core image	cprs: compress a common object	file.
cprs: compress a common object	umask: set and get	file.
umask: set and get	crontab: user crontab	file creation mask.
crontab: user crontab	ctags: create a tags	file.
ctags: create a tags	fields of each line of a	file. cut: cut out selected
fields of each line of a	using the mkfs(1) proto	file database. /software
using the mkfs(1) proto	dd: convert and copy a	file.
dd: convert and copy a	a delta (change) to an SCCS	file. delta: make
a delta (change) to an SCCS	close: close a	file descriptor.
close: close a	dup: duplicate an open	file descriptor.
dup: duplicate an open	dup2: duplicate an open	file descriptor.
dup2: duplicate an open	hd: hexadecimal and ascii	file: determine file type.
hd: hexadecimal and ascii	selected parts of an object	file dump.
selected parts of an object	sact: print current SCCS	file. dump: dump
sact: print current SCCS	crypt: password and	file editing activity.
crypt: password and	endgrent, fgetgrent: get group	file encryption functions.
endgrent, fgetgrent: get group	fgetpwent: get password	file entry. /setgrent,
fgetpwent: get password	utmpname: access utmp	file entry. /endpwent,
utmpname: access utmp	putpwent: write password	file entry. /endutent,
putpwent: write password	write shadow password	file entry.
write shadow password		file entry. putspent:

execvp: execute a	file. /execv, execl, execve, . . . . .	execv(2)
systems export configuration	file. exports: NFS file . . . . .	exports(4)
fgrep: search a	file for a character string. . . . .	fgrep(1)
grep: search a	file for a pattern. . . . .	grep(1)
regular/ egrep: search a	file for a pattern using full . . . . .	egrep(1)
path: locate executable	file for command. . . . .	path(1)
inetd.conf: configuration	file for inetd (internet/ . . . . .	inetd.conf(4)
ldlopen: open a common object	file for reading. ldopen, . . . . .	ldopen(3X)
netrc: login	file for remote networks. . . . .	netrc(4)
aliases: aliases	file for sendmail. . . . .	aliases(4)
lines. Devices: configuration	file for uucp communications . . . . .	Devices(5)
acct: per-process accounting	file format. . . . .	acct(4)
ar: common archive	file format. . . . .	ar(4)
errfile: error-log	file format. . . . .	errfile(4)
intro: introduction to	file formats. . . . .	intro(4)
entries of a common object	file function. /line number . . . . .	ldread(3X)
gateways: routed configuration	file. . . . .	gateways(4)
get: get a version of an SCCS	file. . . . .	get(1)
directory entries and put in a	file. getdents: read . . . . .	getdents(2)
group: group	file. . . . .	group(4)
files. filehdr:	file header for common object . . . . .	filehdr(4)
limits:	file header for/ . . . . .	limits(4)
constants. unistd:	file header for symbolic . . . . .	unistd(4)
file. ldfhread: read the	file header of a common object . . . . .	ldfhread(3X)
ldohseek: seek to the optional	file header of a common object/ . . . . .	ldohseek(3X)
split: split a	file into pieces. . . . .	split(1)
issue: issue identification	file. . . . .	issue(4)
of a member of an archive	file. /read the archive header . . . . .	ldahread(3X)
close a common object	file. ldclose, ldaclose: . . . . .	ldclose(3X)
file header of a common object	file. ldfhread: read the . . . . .	ldfhread(3X)
a section of a common object	file. /line number entries of . . . . .	ldlseek(3X)
file header of a common object	file. /seek to the optional . . . . .	ldohseek(3X)
a section of a common object	file. /relocation entries of . . . . .	ldrseek(3X)
header of a common object	file. /indexed/named section . . . . .	ldshread(3X)
section of a common object	file. /to an indexed/named . . . . .	ldsseek(3X)
table entry of a common object	file. /the index of a symbol . . . . .	ldtbindex(3X)
table entry of a common object	file. /read an indexed symbol . . . . .	ldtbread(3X)
table of a common object	file. /seek to the symbol . . . . .	ldtbseek(3X)
entries in a common object	file. linenum: line number . . . . .	linenum(4)
link: link to a	file. . . . .	link(2)
listing from a common object	file. list: produce C source . . . . .	list(1)
set links/ qlist: print out	file lists from proto file; . . . . .	qlist(1)
access to regions of a	file. locking: exclusive . . . . .	locking(2)
make an ifile from an object	file. mkifile: . . . . .	mkifile(1M)
mknod: build special	file. . . . .	mknod(1M)
or a special or ordinary	file. /make a directory, . . . . .	mknod(2)
ctermid: generate	file name for terminal. . . . .	ctermid(3S)
mktemp: make a unique	file name. . . . .	mktemp(3C)
for a file system	file names and statistics . . . . .	ff(1M)
netcf: Network Configuration	File. . . . .	netcf(4)
data base for the mail aliases	file. newaliases: rebuild the . . . . .	newaliases(1)
change the format of a text	file. newform: . . . . .	newform(1)
name list of common object	file. nm: print . . . . .	nm(1)
null: the null	file. . . . .	null(7)
/find the slot in the utmp	file of the current user. . . . .	ttyslot(3C)
/identify processes using a	file or file structure. . . . .	fuser(1M)
one. creat: create a new	file or rewrite an existing . . . . .	creat(2)
passwd: passwd	file. . . . .	passwd(4)

or subsequent lines of one	file. /lines of several files	paste(1)
pg:	file perusal filter for CRTs.	pg(1)
/rewind, ftell: reposition a	file pointer in a stream.	fseek(3S)
lseek: move read/write	file pointer.	lseek(2)
prs: print an SCCS	file.	prs(1)
queue description	file. /at/batch/cron	queuedefs(4)
read: read from	file.	read(2)
for a common object	file. /relocation information	reloc(4)
resolver configuration	file. resolv.conf:	resolver(4)
Sharing name server master	file. rfmaster: Remote File	rfmaster(4)
remove a delta from an SCCS	file. rmdel:	rmdel(1)
bfs: big	file scanner.	bfs(1)
two versions of an SCCS	file. sccsdiff: compare	sccsdiff(1)
sccsfile: format of SCCS	file.	sccsfile(4)
header for a common object	file. scnhdr: section	scnhdr(4)
format of curses screen image	file. scr_dump:	scr_dump(4)
/out file lists from proto	file; set links based on.	qlist(1)
rfadmin: Remote	File Sharing administration.	rfadmin(1M)
rfdaemon: Remote	File Sharing daemon process.	rfdaemon(1M)
network/ dname: print Remote	File Sharing domain and	dname(1M)
rfstop: stop the Remote	File Sharing environment.	rfstop(1M)
rfpasswd: change Remote	File Sharing host password.	rfpasswd(1M)
master file. rfmaster: Remote	File Sharing name server	rfmaster(4)
query. nsquery: Remote	File Sharing name server	nsquery(1M)
shell/ rfadmin: Remote	File Sharing notification	rfadmin(1M)
unadv: unadvertise a Remote	File Sharing resource.	unadv(1M)
/mount, unmount Remote	File Sharing (RFS) resources.	mountall(1M)
rfstart: start Remote	File Sharing.	rfstart(1M)
mapping. idload: Remote	File Sharing user and group	idload(1M)
fsize: report	file size.	fsize(1)
stat, fstat: get	file status.	stat(2)
the ASCII text strings in a	file. strings: extract	strings(1)
from a common object	file. /line number information	strip(1)
processes using a file or	file structure. /identify	fuser(1M)
checksum and block count of a	file. sum: print	sum(1)
swrite: synchronous write on a	file.	swrite(2)
/symbol name for common object	file symbol table entry.	ldgetname(3X)
syms: common object	file symbol table format.	syms(4)
ckbupscd: check	file system backup schedule.	ckbupscd(1M)
fsdb:	file system debugger.	fsdb(1M)
volume. fs:	file system: format of system	fs(4)
fstyp: determine	file system identifier.	fstyp(1M)
directory entry. dirent:	file system independent	dirent(4)
statfs, fstatfs: get	file system information.	statfs(2)
mkfs: construct a	file system.	mkfs(1M)
mount: mount a	file system.	mount(2)
/mount, unmount Network	File System resources.	nmountall(1M)
nfsstat: Network	File System statistics.	nfsstat(1M)
ustat: get	file system statistics.	ustat(2)
fsstat: report	file system status.	fsstat(1M)
mnttab: mounted	file system table.	mnttab(4)
rmtab: remotely mounted	file system table.	rmtab(4)
sysfs: get	file system type information.	sysfs(2)
umount: unmount a	file system.	umount(2)
volcopy: make literal copy of	file system.	volcopy(1M)
system: system description	file.	system(4)
/umount: mount and unmount	file systems and remote/	mount(1M)
configuration/ exports: NFS	file systems export	exports(4)

access time.	dcopy: copy	file systems for optimal	dcopy(1M)
fsck, dfscck: check and repair	file systems.	file systems.	fsck(1M)
labelit: provide labels for	mount, unmount multiple	file systems. /umountall:	labelit(1M)
and/ checklist: list of	deliver the last part of a	file systems processed by fsck	mountall(1M)
term: format of compiled term	file. tail:	file..	checklist(4)
tmpfile: create a temporary	file.	file.	tail(1)
create a name for a temporary	file. tmpnam, tempnam:	file. tmpfile	term(4)
and modification times of a	file. touch: update access	file. tmpnam(3S)	tmpfile(3S)
ftp: ARPANET	file transfer program.	file. tmpnam(3S)	tmpnam(3S)
ftpd: DARPA Internet	File Transfer Protocol server.	file. touch	touch(1)
tftpd: DARPA Trivial	File Transfer Protocol server.	file transfer program.	ftp(1)
uucp system. uucico:	file transport program for the	File Transfer Protocol server.	ftpd(1M)
ftw: walk a	file tree.	File Transfer Protocol server.	tftpd(1M)
file: determine	file type.	file transport program for the	uucico(1M)
undo a previous get of an SCCS	file. unget:	file tree.	ftw(3C)
report repeated lines in a	file. uniq:	file type.	file(1)
directories and permissions	file. uueck: check the uucp	file. unget:	unget(1)
val: validate SCCS	file.	file. uniq:	uniq(1)
write: write on a	file.	file. uueck: check the uucp	uueck(1M)
umask: set	file-creation mode mask.	file.	val(1)
common object files.	filehdr: file header for	file.	write(2)
error, feof, clearerr,	fileno: stream status/	file.	umask(1)
and print process accounting	file(s). acctcom: search	filehdr: file header for	filehdr(4)
merge or add total accounting	files. acctmerg:	fileno: stream status/	ferror(3S)
create and administer SCCS	files. admin:	file(s). acctcom: search	acctcom(1)
link, unlink: link and unlink	files and directories.	files. acctmerg:	acctmerg(1M)
cat: concatenate and print	files.	files. admin:	admin(1)
cmp: compare two	files.	link, unlink: link and unlink	link(1M)
lines common to two sorted	files. comm: select or reject	cat: concatenate and print	cat(1)
ln, mv: copy, link, or move	files. cp, . . . . .	cmp: compare two	cmp(1)
mark differences between	files. diffmk:	lines common to two sorted	comm(1)
file header for common object	files. filehdr:	ln, mv: copy, link, or move	cp(1)
find: find	files.	mark differences between	diffmk(1)
frec: recover	files from a backup tape.	file header for common object	filehdr(4)
format specification in text	files. fspec:	find: find	find(1)
FORTTRAN, ratfor, or efl	files. fsplit: split	frec: recover	frec(1M)
string, format of graphical	files. /graphical primitive	format specification in text	fspec(4)
cpset: install object	files in binary directories.	FORTTRAN, ratfor, or efl	fsplit(1)
language preprocessor include	files. includes: determine C	string, format of graphical	gps(4)
intro: introduction to special	files.	cpset: install object	cpset(1M)
link editor for common object	files. ld:	language preprocessor include	includes(1)
lockf: record locking on	files.	intro: introduction to special	intro(7)
passmgmt: password	files management.	link editor for common object	ld(1)
rm, rmdir: remove	files or directories.	lockf: record locking on	lockf(3C)
/merge same lines of several	files or subsequent lines of/	passmgmt: password	passmgmt(1M)
unpack: compress and expand	files. pack, pcat,	rm, rmdir: remove	rm(1)
pr: print	files.	/merge same lines of several	paste(1)
in bytes of common object	files. /print section sizes	unpack: compress and expand	pack(1)
sort: sort and/or merge	files.	pr: print	pr(1)
convert: convert archive	files to common formats.	in bytes of common object	size(1)
what: identify SCCS	files.	sort: sort and/or merge	sort(1)
fstab:	filter for CRTs.	convert: convert archive	convert(1)
pg: file perusal	filter.	what: identify SCCS	what(1)
greek: select terminal	filter.	fstab:	fstab(4)
nl: line numbering	filter reverse line-feeds.	pg: file perusal	pg(1)
col:		greek: select terminal	greek(1)
		nl: line numbering	nl(1)
		col:	col(1)

tio: tape io	filter. . . . .	tio(1)
graphical device routines and	filters. /tekset, td: . . . . .	gdev(1G)
tplot: graphics	filters. . . . .	tpplot(1G)
find:	finc: fast incremental backup. . . . .	finc(1M)
find files. . . . .	find files. . . . .	find(1)
hyphen:	find hyphenated words. . . . .	hyphen(1)
ttyname, isatty:	find name of a terminal. . . . .	ttyname(3C)
object library. lorder:	find ordering relation for an	lorder(1)
hashmake, spellin, hashcheck:	find spelling errors. spell. . . . .	spell(1)
of the current user. ttyslot:	find the slot in the utmp file	ttyslot(3C)
lookup program.	finger: user information . . . . .	finger(1)
information server.	fingerd: remote user . . . . .	fingerd(1M)
fold: fold long lines for	finite width output device. . . . .	fold(1)
dbminit, fetch, store, delete,	firstkey, nextkey: database/	dbm(3X)
fish: play "Go	Fish". . . . .	fish(6)
tee: pipe	fitting. . . . .	tee(1)
/fpgetsticky, fpsetsticky: IEEE	floating point environment/	fpgetround(3)
isnand, isnanf: test for	floating point NaN/ isnan: . . . . .	isnan(3C)
ecvt, fcvt, gcvt: convert	floating-point number to/	ecvt(3C)
/modf: manipulate parts of	floating-point numbers. . . . .	frexp(3C)
floor, ceil, fmod, fabs:	floor, ceiling, remainder./	floor(3M)
cflow: generate C	flowgraph. . . . .	cflow(1)
fclose, fflush: close or	flush a stream. . . . .	fclose(3S)
remainder./ floor, ceil,	fmod, fabs: floor, ceiling,	floor(3M)
width output device. fold:	fold long lines for finite	fold(1)
stream.	fopen, freopen, fdopen: open a	fopen(3S)
advertised resource. fumount:	forced unmount of an	fumount(1M)
per-process accounting file	fork: create a new process. . . . .	fork(2)
service request/ nlsrequest:	format. acct: . . . . .	acct(4)
ar: common archive file	format and send listener . . . . .	nlsrequest(3n)
errfile: error-log file	format. . . . .	ar(4)
nroff or/ eqn, neqn, checkeq:	format. . . . .	errfile(4)
newform: change the	format mathematical text for	eqn(1)
inode:	format of a text file. . . . .	newform(1)
term:	format of an i-node. . . . .	inode(4)
core:	format of compiled term file.	term(4)
cpio:	format of core image file.	core(4)
file. scr_dump:	cpio: format of cpio archive. . . . .	cpio(4)
dir:	format of curses screen image	scr_dump(4)
/graphical primitive string,	format of directories. . . . .	dir(4)
scsfile:	format of graphical files. . . . .	gps(4)
fs: file system:	format of SCCS file. . . . .	scsfile(4)
files. fspec:	format of system volume. . . . .	fs(4)
object file symbol table	format specification in text	fspec(4)
troff. tbl:	format. syms: common . . . . .	syms(4)
nroff:	format tables for nroff or	tbl(1)
archive files to common	format text. . . . .	nroff(1)
intro: introduction to file	formats. convert: convert . . . . .	convert(1)
wtmp: utmp and wtmp entry	formats. . . . .	intro(4)
scanf, fscanf, sscanf: convert	formats. utmp. . . . .	utmp(4)
/vprintf, vsprintf: print	formatted input. . . . .	scanf(3S)
fprintf, sprintf: print	formatted output of a varargs/	vprintf(3S)
/checkmm: print/check documents	formatted output. printf, . . . . .	printf(3S)
mptx: the macro package for	formatted with the MM macros.	mm(1)
mm: the MM macro package for	formatting a permuted index. . . . .	mptx(5)
ms: text	formatting documents. . . . .	mm(5)
man: macros for	formatting macros. . . . .	ms(5)
	formatting manual pages. . . . .	man(5)

me: macros for	formatting papers.	me(5)
ASSIST menus and command	forms. /generate/modify	astgen(1)
ratfor: rational	FORTTRAN dialect.	ratfor(1)
efl: extended	FORTTRAN language.	efl(1)
files. fsplit: split	FORTTRAN, ratfor, or efl	fsplit(1)
hopefully interesting, adage.	fortune: print a random,	fortune(6)
fpgetround, fpsetround,	fpgetmask, fpsetmask/	fpgetround(3)
fpgetmask, fpsetmask/	fpgetround, fpsetround,	fpgetround(3)
/fpgetmask, fpsetmask,	fpgetsticky, fpsetsticky: IEEE/	fpgetround(3)
formatted output. printf,	fprintf, sprintf: print	printf(3S)
/fpsetround, fpgetmask,	fpsetmask, fpgetsticky/	fpgetround(3)
fpsetmask/ fpgetround,	fpsetround, fpgetmask,	fpgetround(3)
point/ /fpsetmask, fpgetsticky,	fpsetsticky: IEEE floating	fpgetround(3)
word on a/ putc, putchar,	fputc, putw: put character or	putc(3S)
stream. puts,	fputs: put a string on a	puts(3S)
input/output.	fread, fwrite: binary	fread(3S)
backup tape.	frec: recover files from a	frec(1M)
t_free:	free a library structure.	t_free(3n)
df: report number of	free disk blocks and i-nodes.	df(1M)
memory allocator. malloc,	free, realloc, calloc: main	malloc(3C)
mallopt, mallinfo:/ malloc,	free, realloc, calloc,	malloc(3X)
stream. fopen,	freopen, fdopen: open a	fopen(3S)
parts of floating-point/	frexp, ldexp, modf: manipulate	frexp(3C)
frec: recover files	from a backup tape.	frec(1M)
list: produce C source listing	from a common object file.	list(1)
/and line number information	from a common object file.	strip(1)
/receive the confirmation	from a connect request.	t_rcvconnect(3)
recvfrom: receive a message	from a socket. recv,	recv(2)
getw: get character or word	from a stream. /fgetc,	getc(3S)
gets, fgets: get a string	from a stream.	gets(3S)
mkifile: make an ifile	from an object file.	mkifile(1M)
rmdel: remove a delta	from an SCCS file.	rmdel(1)
getopt: get option letter	from argument vector.	getopt(3C)
t_rcvdis: retrieve information	from disconnect.	t_rcvdis(3n)
records and status information	from dump. /extract error	errdead(1M)
/etc/shadow with information	from /etc/passwd. /and update	pwconv(1M)
/etc/shadow with information	from /etc/passwd. /and update	pwunconv(1M)
read: read	from file.	read(2)
ncheck: generate path names	from i-numbers.	ncheck(1M)
nlist: get entries	from name list.	nlist(3C)
acctcms: command summary	from per-process accounting/	acctcms(1M)
qlist: print out file lists	from proto file; set links/	qlist(1)
getpw: get name	from UID.	getpw(3C)
cc1sw, cc2sw, cc2fp:	front-end to the cc command.	cc1sw(1)
genccc: create a	front-end to the cc command.	genccc(1M)
system volume.	fs: file system: format of	fs(4)
formatted input. scanf,	fscanf, sscanf: convert	scanf(3S)
of file systems processed by	fsck and ncheck. /list	checklist(4)
file systems.	fsck, dfscck: check and repair	fsck(1M)
a lost+found directory for	fsck. mklost+found: make	mklostfnd(1M)
reposition a file pointer in/	fsdb: file system debugger.	fsdb(1M)
	fseek, rewind, ftell:	fseek(3S)
	fsize: report file size.	fsize(1)
text files.	fspec: format specification in	fspec(4)
or efl files.	fsplit: split FORTRAN, ratfor,	fsplit(1)
status.	fsstat: report file system	fsstat(1M)
	fstab: file-system-table.	fstab(4)
stat,	fstat: get file status.	stat(2)

information.	statsf:	get file system	statsf(2)
identifier.	fstyp:	determine file system	fstyp(1M)
pointer in a/ fseek, rewind,	ftell:	reposition a file	fseek(3S)
communication/ stdipc,	ftok:	standard interprocess	stdipc(3C)
program.	ftp:	ARPANET file transfer	ftp(1)
Transfer Protocol server.	ftpd:	DARPA Internet File	ftpd(1M)
	ftw:	walk a file tree.	ftw(3C)
/a file for a pattern using	full:	regular expressions.	egrep(1)
shutdown: shut down part of a	full-duplex:	connection.	shutdown(2)
advertised resource.	fumount:	forced unmount of an	fumount(1M)
error/ erf, erfc: error	function and complementary		erf(3M)
gamma: log gamma	function.		gamma(3M)
hypot: Euclidean distance	function.		hypot(3M)
of a common object file	function. /line number entries		ldread(3X)
matherr: error-handling	function.		matherr(3M)
prof: profile within a	function.		prof(5)
math: math	functions and constants.		math(5)
intro: introduction to	functions and libraries.		intro(3)
j0, j1, jn, y0, y1, yn: Bessel	functions. bessel:		bessel(3M)
password and file encryption	functions. crypt:		crypt(3X)
logarithm, power, square root	functions. /sqrt: exponential,		exp(3M)
remainder, absolute value	functions. /floor, ceiling,		floor(3M)
ocurse: optimized screen	functions.		ocurse(3X)
300, 300s: handle special	functions of DASI 300 and 300s/		300(1)
terminals. hp: handle special	functions of Hewlett-Packard		hp(1)
terminal. 450: handle special	functions of the DASI 450		450(1)
sinh, cosh, tanh: hyperbolic	functions.		sinh(3M)
atan, atan2: trigonometric	functions. /tan, asin, acos,		trig(3M)
	fusage: disk access profiler.		fusage(1M)
using a file or file/	fuser: identify processes		fuser(1M)
fread,	fwrite: binary input/output.		fread(3S)
connect accounting records.	fwtmp, wtmpfix: manipulate		fwtmp(1M)
moo: guessing	game.		moo(6)
back: the	game of backgammon.		back(6)
bj: the	game of black jack.		bj(6)
craps: the	game of craps.		craps(6)
wump: the	game of hunt-the-wumpus.		wump(6)
trk: trekkie	game.		trk(6)
intro: introduction to	games.		intro(6)
gamma: log	gamma function.		gamma(3M)
file.	gateways: routed configuration		gateways(4)
number to string. ecvt, fcvt,	gcvt: convert floating-point		ecvt(3C)
tekset, td: graphical device/	gdev: hpd, erase, hardcopy,		gdev(1G)
	ged: graphical editor.		ged(1G)
the cc command.	gencc: create a front-end to		gencc(1M)
maze:	generate a maze.		maze(6)
abort:	generate a SIGABRT.		abort(3C)
cflow:	generate C flowgraph.		cflow(1)
cross-reference. cxref:	generate C program		cxref(1)
classification and/ chrtbl:	generate character		chrtbl(1M)
by user ID. diskusg:	generate disk accounting data		diskusg(1M)
makekey:	generate encryption key.		makekey(1)
terminal. ctermid:	generate file name for		ctermid(3S)
crypt, setkey, encrypt:	generate hashing encryption.		crypt(3C)
i-numbers. ncheck:	generate path names from		ncheck(1M)
lexical tasks. lex:	generate programs for simple		lex(1)
/srand48, seed48, lcong48:	generate uniformly distributed/		drand48(3C)
and command forms. astgen:	generate/modify ASSIST menus		astgen(1)



rand: simple random-number	generator.	rand,	rand(3C)
	gets, fgets:	get a string from a stream.	gets(3S)
	get:	get a version of an SCCS file.	get(1)
getsockopt, setsockopt:	get and set options on/	getsockopt	(2)
	ulimit:	get and set user limits.	ulimit(2)
	the user. cuserid:	get character login name of	cuserid(3S)
getc, getchar, fgetc, getw:	get character or word from a/	getc	(3S)
through the/ nlsgetcall:	get client's data passed	nlsgetcall	(3n)
	getdtablesize:	get descriptor table size.	getdtablesize(2)
	nlist:	get entries from name list.	nlist(3C)
umask: set and	get file creation mask.	umask	(2)
	stat, fstat:	get file status.	stat(2)
	statfs, fstatfs:	get file system information.	statfs(2)
	ustat:	get file system statistics.	ustat(2)
information. sysfs:	get file system type	sysfs	(2)
	file:	get: get a version of an SCCS	get(1)
/setgrent, endgrent, fgetgrent:	get group file entry.	getgrent	(3C)
	getlogin:	get login name.	getlogin(3C)
	logname:	get login name.	logname(1)
	msgget:	get message queue.	msgget(2)
	getpw:	get name from UID.	getpw(3C)
	getpeername:	get name of connected peer.	getpeername(2)
	system. uname:	get name of current CTIX	uname(2)
provider. nlsprovider:	get name of transport	nlsprovider	(3n)
host. getservaddr:	get network address of service	getserv	(1M)
/setnetent, endnetent:	get network entry.	getnetent	(3)
/sethostent, endhostent:	get network host entry.	gethostbyname	(3)
	getmsg:	get next message off a stream.	getmsg(2)
unset: undo a previous	get of an SCCS file.	unset	(1)
argument vector. getopt:	get option letter from	getopt	(3C)
/setpwent, endpwent, fgetpwent:	get password file entry.	getpwent	(3C)
working directory. getcwd:	get path-name of current	getcwd	(3C)
	times. times:	get process and child process	times(2)
and/ getpid, getpgid, getppid:	get process, process group,	getpid	(2)
/setprotoent, endprotoent:	get protocol entry.	getprotoent	(3)
information. l_getinfo:	get protocol-specific service	l_getinfo	(3n)
/getuid, getgid, getegid:	get real user, effective user/	getuid	(2)
getrpcbyname, getrpcbynumber:	get rpc entry. getrpcent,	getrpcent	(3)
	getrpcport:	get RPC port number.	getrpcport(3)
/setservent, endservent:	get service entry.	setservent	(3)
	semget:	get set of semaphores.	semget(2)
fgetspent, lckpwwdf, ulckpwwdf:	get shadow. /endspent,	getspent	(3X)
identifier. shmget:	get shared memory segment	shmget	(2)
	getsockname:	get socket name.	getsockname(2)
	t_getstate:	get the current state.	t_getstate(3)
	tty:	get the name of the terminal.	tty(1)
	time:	get time.	time(2)
get character or word from a/	getc, getchar, fgetc, getw:	getc	(3S)
current working directory.	getcwd:	get path-name of	getcwd(3C)
entries and put in a file.	getdents:	read directory	getdents(2)
	table size. getdtablesize:	get descriptor	getdtablesize(2)
	getuid, geteuid, getgid,	getgid: get real user/	getuid(2)
	environment name.	getenv: return value for	getenv(3C)
real user, effective/ getuid,	geteuid, getgid, getegid:	get	getuid(2)
user/ getuid, geteuid,	getgid, getegid:	get real	getuid(2)
setgrent, endgrent/	getgrent, getrgid, getgiam,	getgrent	(3C)
gethostent, sethostent,/	gethostbyname, gethostbyaddr,	gethostbyname	(3)
unique identifier of current/	gethostid, sethostid: get/set	gethostid	(2)

get/set name of current host.	gethostname, sethostname:	gethostname(2)
	getlogin: get login name.	getlogin(3C)
stream.	getmsg: get next message off a	getmsg(2)
getnetbyname, setnetent./	getnetent, getnetbyaddr,	getnetent(3)
argument vector.	getopt: get option letter from	getopt(3C)
	getopt: parse command options.	getopt(1)
options. getopt, getoptcv:	getoptcv: parse command	getopts(1)
command options.	getopts, getoptcv: parse	getopts(1)
	getpass: read a password.	getpass(3C)
connected peer.	getpeername: get name of	getpeername(2)
process group, and/ getpid,	getpgid, getpid: get process,	getpid(2)
process, process group, and/	getpid, getpgid, getpid: get	getpid(2)
group, and/ getpid, getpgid,	getpid: get process, process	getpid(2)
getprotobyname, setprotoent./	getprotoent, getprotobyname,	getprotoent(3)
	getpw: get name from UID.	getpw(3C)
setpwent, endpwent./	getpwent, getpwuid, getpwnam,	getpwent(3C)
getpwent, getpwuid,	getpwnam, setpwent, endpwent./	getpwent(3C)
endpwent./ getpwent,	getpwuid, getpwnam, setpwent,	getpwent(3C)
get rpc entry. getrpcent,	getrpcbyname, getrpcbynumber:	getrpcent(3)
getrpcbynumber: get rpc/	getrpcent, getrpcbyname,	getrpcent(3)
number.	getrpcport: get RPC port	getrpcport(3)
a stream.	gets, fgets: get a string from	gets(3S)
address of service host.	getservaddr: get network	getserv(1M)
getservbyname, setservent./	getservent, getservbyport,	getservent(3)
gettimeofday, settimeofday:	get/set date and time.	gettimeofday(2)
gethostname, sethostname:	get/set name of current host.	gethostname(2)
current/ gethostid, sethostid:	get/set unique identifier of	gethostid(2)
	getsockname: get socket name.	getsockname(2)
and set options on sockets.	getsockopt, setsockopt: get	getsockopt(2)
endspent, fgetsent, lckpwwdf./	getspent, getspnam, setspent,	getspent(3X)
get/set date and time.	gettimeofday, settimeofday:	gettimeofday(2)
and terminal settings used by	getty. gettydefs: speed	gettydefs(4)
modes, speed, and line/	getty: set terminal type,	getty(1M)
ct: spawn	getty to a remote terminal.	ct(1C)
settings used by getty.	gettydefs: speed and terminal	gettydefs(4)
getcgid: get real user./	getuid, geteuid, getgid,	getuid(2)
pututline, setutent./ getut:	getutent, getutid, getutline,	getut(3C)
setutent./ getut: getutent,	getutid, getutline, pututline,	getut(3C)
getut: getutent, getutid,	getutline, pututline./	getut(3C)
from a/ getc, getchar, fgets,	getw: get character or word	getc(3S)
common CTIX system terms and/	glossary: definitions of	glossary(1)
asctime/ ctime, localtime,	grmtime, asctime, ctime,	ctime(3C)
fish: play	"Go Fish".	fish(6)
setjmp, longjmp: non-local	goto.	setjmp(3C)
string, format of graphical/	gps: graphical primitive	gps(4)
graph: draw a	graph.	graph(1G)
sag: system activity	graph.	sag(1G)
commands. graphics: access	graphical and numerical	graphics(1G)
/network useful with	graphical commands.	stat(1G)
/erase, hardcopy, tekset, id:	graphical device routines and/	gdev(1G)
ged:	graphical editor.	ged(1G)
primitive string, format of	graphical files. /graphical	gps(4)
toc: dtoc, ttoc, vtoc:	graphical table of contents/	toc(1G)
gutil:	graphical utilities.	gutil(1G)
numerical commands.	graphics: access graphical and	graphics(1G)
tplot:	graphics filters.	tplot(1G)
plot:	graphics interface.	plot(4)
subroutines. plot:	graphics interface	plot(3X)

mvt: typeset documents, view package for typesetting view	graphs, and slides. mmt,	mmt(1)
pattern.	graphs and slides. /macro	mv(5)
/user, effective user, real	greek: select terminal filter.	greek(1)
/getppid: get process, process	grep: search a file for a	grep(1)
chown, chgrp: change owner or	group, and effective group/	getuid(2)
endgrent, fgetgrent: get	group, and parent process IDs.	getpid(2)
group:	group.	chown(1)
setpgrp: set process	group file entry. /setgrent,	getgrent(3C)
id: print user and	group file.	group(4)
real group, and effective	group ID.	setpgrp(2)
setuid, setgid: set user and	group IDs and names.	id(1M)
Remote File Sharing user and	group IDs. /effective user,	getuid(2)
newgrp: log in to a new	group IDs.	setuid(2)
chown: change owner and	group mapping. idload:	idload(1M)
a signal to a process or a	group.	newgrp(1M)
update, and regenerate	group of a file.	chown(2)
checkers. pwck,	group of processes. /send	kill(2)
ssignal,	groups of programs. /maintain,	make(1)
install or relocate a PT or	grpck: password/group file	pwck(1M)
download. tdl,	gsignal: software signals.	ssignal(3C)
hangman:	GT local printer. /mvtpty:	mkpty(1)
moo:	gtdl, ptdl: RS-232 terminal	tdl(1)
/for Interphase V/TAPE 3200	guess the word.	hangman(6)
stape: SCSI quarter-inch and	guessing game.	moo(6)
system state. shutdown,	gutil: graphical utilities.	gutil(1G)
DASI 300 and 300s/ 300, 300s:	half-inch tape controller.	ipt(7)
Hewlett-Packard/ hp:	half-inch tape.	stape(7)
the DASI 450 terminal. 450:	halt: shut down system, change	shutdown(1M)
varargs:	handle special functions of	300(1)
curses: terminal screen	handle special functions of	hp(1)
setchrclass: character	handle special functions of	450(1)
nohup: run a command immune to	handle variable argument list.	varargs(5)
graphical/ gdev: hpd, erase,	handling and optimization/	curses(3X)
hinvt:	handling. /_tolower, _toupper,	ctype(3C)
hcreate, hdestroy: manage	hangman: guess the word.	hangman(6)
spell, hashmake, spellin,	hangups and quits.	nohup(1)
setkey, encrypt: generate	hardcopy, tekset, td:	gdev(1G)
find spelling errors. spell,	hardware inventory.	hinvt(1M)
search tables. hsearch,	hash search tables. hsearch,	hsearch(3C)
dump.	hashcheck: find spelling/	spell(1)
tables. hsearch, hcreate,	hashing encryption. crypt,	crypt(3C)
file. scnhdr: section	hashmake, spellin, hashcheck:	spell(1)
files. filehdr: file	hcreate, hdestroy: manage hash	hsearch(3C)
limits: file	hd: hexadecimal and ascii file	hd(1)
unistd: file	hdestroy: manage hash search	hsearch(3C)
file. ldfhread: read the file	header for a common object	scnhdr(4)
/seek to the optional file	header for common object	filehdr(4)
/read an indexed/named section	header for/	limits(4)
ldahread: read the archive	header for symbolic constants.	unistd(4)
helpadm: make changes to the	header of a common object	ldfhread(3X)
help: CTIX system	header of a common object/	ldohseek(3X)
Help Facility database.	header of a common object/	ldshread(3X)
tape file archiver. hpio:	header of a member of an/	ldahread(3X)
/handle special functions of	Help Facility database.	helpadm(1M)
	Help Facility.	help(1)
	helpadm: make changes to the	helpadm(1M)
	Hewlett-Packard 2645A terminal	hpio(1)
	Hewlett-Packard terminals.	hp(1)

dump. hd:	hexadecimal and ascii file	hd(1)
libdev: manipulate Volume	hin v: hardware inventory.	hin v(1M)
fortune: print a random,	Home Blocks (VHB).	libdev(3X)
/ntohs: convert values between	hopefully interesting, adage.	fortune(6)
endhostent: get network	host and network byte order.	byteorder(3)
unique identifier of current	host entry. /sethostent,	gethostbyname(3)
get/set name of current	host. /sethostid: get/set	gethostid(2)
get network address of service	host. /sethostname:	gethostname(2)
/set or print the Internet	host. getservaddr:	getservad(1M)
change Remote File Sharing	host name of the current/	hostname(1)
rwhod:	host password. rfpaswd:	rfpaswd(1M)
or print identifier of current	host status server.	rwhod(1M)
identifier of current host/	host system. hostid: set	hostid(1)
Internet host name of the/	hostid: set or print	hostid(1)
packets to network	hostname: set or print the	hostname(1)
of Hewlett-Packard terminals.	hosts. /send ICMP ECHO_REQUEST	ping(1M)
id: graphical device/ gdev:	hp: handle special functions	hp(1)
terminal tape file archiver.	hpd, erase, hardcopy, tekset,	gdev(1G)
manage hash search tables.	hpio: Hewlett-Packard 2645A	hpio(1)
convert values between host/	hsearch, hcreate, hdestroy:	hsearch(3C)
values between host/ htonl,	htonl, htons, ntohl, ntohs:	byteorder(3)
wump: the game of	htons, ntohl, ntohs: convert	byteorder(3)
sinh, cosh, tanh:	hunt-the-wumpus.	wump(6)
hyphen: find	hyperbolic functions.	sinh(3M)
function.	hyphenated words.	hyphen(1)
network hosts. ping: send	hypot: Euclidean distance	hypot(3M)
Protocol.	icmp: Internet Control Message	ping(1M)
disk accounting data by user	ID. diskusg: generate	diskusg(1M)
semaphore set or shared memory	ID. /remove a message queue,	ipcrm(1)
and names.	id: print user and group IDs	id(1M)
setpgpr: set process group	ID.	setpgpr(2)
issue: issue	identification file.	issue(4)
fstyp: determine file system	identi fier.	fstyp(1M)
/sethostid: get/set unique	identi fier of current host.	gethostid(2)
system. hostid: set or print	identi fier of current host	hostid(1)
get shared memory segment	identi fier. shmget:	shmget(2)
using keywords. locate:	identify a CTIX system command	locate(1)
file or file/ fuser:	identify processes using a	fuser(1M)
what:	identify SCCS files.	what(1)
user and group mapping.	idload: Remote File Sharing	idload(1M)
id: print user and group	IDs and names.	id(1M)
group, and parent process	IDs. /get process, process	getpid(2)
group, and effective group	IDs. /effective user, real	getuid(2)
setgid: set user and group	IDs. setuid,	setuid(2)
/fpgetsticky, fpsetsticky:	IEEE floating point/	fpgetround(3)
interface parameters.	ifconfig: configure network	ifconfig(1M)
mkfile: make an	ifile from an object file.	mkfile(1M)
core: format of core	image file.	core(4)
format of curses screen	image file. scr_dump:	scr_dump(4)
crash: examine system	images.	crash(1M)
nohup: run a command	immune to hangups and quits.	nohup(1)
limits: file header for	implementation-specific/	limits(4)
C language preprocessor	include files. /determine	includes(1)
finc: fast	incremental backup.	finc(1M)
dirent: file system	independent directory entry.	dirent(4)
/tgoto, tputs: terminal	independent operations.	otermcap(3X)
for formatting a permuted	index. /the macro package	mptx(5)

of a/ ldtbindex: compute the	index of a symbol table entry	ldtbindex(3X)
ptx: permuted	index.	ptx(1)
a common/ ldtbread: read an	indexed symbol table entry of	ldtbread(3X)
ldshread, ldnsbread: read an	indexed/named section header/	ldshread(3X)
ldsseek, ldnsseek: seek to an	indexed/named section of a/	ldsseek(3X)
receive of an orderly release	indication. /acknowledge	t_rcvrel(3n)
receive a unit data error	indication. t_rcvuderr:	t_rcvuderr(3)
family.	inet: Internet protocol	inet(7)
inet_ntoa, inet_makeaddr,/	inet_addr, inet_network,	inet(3)
"super-server".	inetd: internet	inetd(1M)
configuration file for	inetd (internet/ inetd.conf:	inetd.conf(4)
for inetd (internet/	inetd.conf: configuration file	inetd.conf(4)
/inet_ntoa, inet_makeaddr,	inet_lnaof, inet_netof:/	inet(3)
/inet_network, inet_ntoa,	inet_makeaddr, inet_lnaof/	inet(3)
/inet_makeaddr, inet_lnaof/	inet_netof: Internet address/	inet(3)
/inet_makeaddr/ inet_addr,	inet_network, inet_ntoa,	inet(3)
inet_addr, inet_network,	inet_ntoa, inet_makeaddr/	inet(3)
terminfo descriptions.	infocmp: compare or print out	infocmp(1M)
inittab: script for the	init process.	inittab(4)
initialization.	init, telinit: process control	init(1M)
init, telinit: process control	initialization.	init(1M)
/drvload, powerfail: system	initialization procedures.	brc(1M)
terminfo database. tput:	initialize a terminal or query	tput(1)
volume. iv:	initialize and maintain	iv(1)
socket. connect:	initiate a connection on a	connect(2)
t_sndrel:	initiate an orderly release.	t_sndrel(3n)
process. popen, pclose:	initiate pipe to/from a	popen(3S)
process.	inittab: script for the init	inittab(4)
cli: clear	i-node.	cli(1M)
inode: format of an	i-node.	inode(4)
number of free disk blocks and	i-nodes. df: report	df(1M)
start and stop terminal	input and output. /manually	rsterm(1M)
sscanf: convert formatted	input. scanf, fscanf,	scanf(3S)
push character back into	input stream. ungetc:	ungetc(3S)
fread, fwrite: binary	input/output.	fread(3S)
poll: STREAMS	input/output multiplexing.	poll(2)
stdio: standard buffered	input/output package.	stdio(3S)
fileno: stream status	inquiries. /feof, clearerr,	ferror(3S)
uustat: uucp status	inquiry and job control.	uustat(1C)
with information from/ pwconv:	install and update /etc/shadow	pwconv(1M)
with information/ pwunconv:	install and update /etc/shadow	pwunconv(1M)
using the mkfs(1)/ qinstall:	install and verify software	qinstall(1)
install:	install commands.	install(1M)
directories. cpset:	install object files in binary	cpset(1M)
local printer. mktpy, mvtpy:	install or relocate a PT or GT	mktpy(1)
ctinstall:	install software.	ctinstall(1)
abs: return	integer absolute value.	abs(3C)
/l64a: convert between long	integer and base-64 ASCII/	a64l(3C)
sputl, sgetl: access long	integer data in a/	sputl(3X)
atol, atoi: convert string to	integer. strtol,	strtol(3C)
3-byte integers and long	integers. /convert between	l3tol(3C)
bcopy:	interactive block copy.	bcopy(1M)
system. mailx:	interactive message processing	mailx(1)
print a random, hopefully	interesting, adage: fortune:	fortune(6)
tset: set terminal, terminal	interface, and terminal/	tset(1)
module. timod: Transport	Interface cooperating STREAMS	timod(7)
err: error-logging	interface.	err(7)
V/TAPE 3200 half-inch/ ipt:	interface for Interphase	ipt(7)

	qic: interface for QIC tape. . . . .	qic(7)
lo: software loopback network	interface. . . . .	lo(7)
lp: parallel printer	interface. . . . .	lp(7)
mem, kmem: system memory	interface. . . . .	mem(7)
ifconfig: configure network	interface parameters. . . . .	ifconfig(1M)
plot: graphics	interface. . . . .	plot(4)
STREAMS/ tirdwr: Transport	Interface read/write interface . . . . .	tirdwr(7)
/Transport Interface read/write	interface STREAMS module. . . . .	tirdwr(7)
plot: graphics	interface subroutines. . . . .	plot(3X)
swap: swap administrative	interface. . . . .	swap(1M)
termio: general terminal	interface. . . . .	termio(7)
tiop: terminal accelerator	interface. . . . .	tiop(7)
logging and event/ log:	interface to STREAMS error . . . . .	log(7)
telnet: user	interface to TELNET protocol. . . . .	telnet(1)
protocol. tftp: user	interface to the DARPA TFTP . . . . .	tftp(1)
tty: controlling terminal	interface. . . . .	tty(7)
vme: VME bus	interface. . . . .	vme(7)
detach serial lines as network	interfaces. /attach and . . . . .	slattach(1M)
/inet_1naof, inet_netof:	Internet address manipulation/ . . . . .	inet(3)
Protocol. icmp:	Internet Control Message . . . . .	icmp(7)
named:	Internet domain name server. . . . .	named(1M)
Protocol server. ftpd: DARPA	Internet File Transfer . . . . .	ftpd(1M)
hostname: set or print the	Internet host name of the/ . . . . .	hostname(1)
names and numbers for the	internet. networks: . . . . .	networks(4)
slipd: switched Serial Line	Internet Protocol control/ . . . . .	slipd(1M)
inet:	Internet protocol family. . . . .	inet(7)
ip:	Internet Protocol. . . . .	ip(7)
protocols: list of	Internet protocols. . . . .	protocols(4)
services: list of	Internet services. . . . .	services(4)
inetd:	internet "super-server". . . . .	inetd(1M)
/configuration file for inetd	(internet "super-server"). . . . .	inetd.conf(4)
Protocol. tcp:	Internet Transmission Control . . . . .	tcp(7)
Protocol. udp:	Internet User Datagram . . . . .	udp(7)
half-inch/ ipt: interface for	Interphase V/TAPE 3200 . . . . .	ipt(7)
spline:	interpolate smooth curve. . . . .	spline(1G)
characters. asa:	interpret ASA carriage control . . . . .	asa(1)
sno: SNOBOL	interpreter. . . . .	sno(1)
syntax. csh: a shell (command	interpreter) with C-like . . . . .	csh(1)
pipe: create an	interprocess channel. . . . .	pipe(2)
facilities/ ipcs: report	inter-process communication . . . . .	ipcs(1)
stdipc, ftok: standard	interprocess communication/ . . . . .	stdipc(3C)
suspend execution for an	interval. sleep: . . . . .	sleep(1)
sleep: suspend execution for	interval. . . . .	sleep(3C)
application programs. intro:	introduction to commands and . . . . .	intro(1)
intro:	introduction to file formats. . . . .	intro(4)
libraries. intro:	introduction to functions and . . . . .	intro(3)
intro:	introduction to games. . . . .	intro(6)
intro:	introduction to miscellany. . . . .	intro(5)
intro:	introduction to special files. . . . .	intro(7)
and error numbers. intro:	introduction to system calls . . . . .	intro(2)
generate path names from	i-numbers. ncheck: . . . . .	ncheck(1M)
hiniv: hardware	inventory. . . . .	hiniv(1M)
tio: tape	io filter. . . . .	tio(1)
select: synchronous	I/O multiplexing. . . . .	select(2)
table. rtab: Remote	I/O Processor configuration . . . . .	rtab(4)
riopqry: query Remote	I/O Processor for online data. . . . .	riopqry(1M)
configure system for Remote	I/O Processor. riopcf: . . . . .	riopcfg(1M)
streamio: STREAMS	ioctl commands. . . . .	streamio(7)

	ioctl: control device. . . . .	ioctl(2)
I/O Processor's RAM	iopdump: upload a Front-end . . . . .	iopdump(1M)
	ip: Internet Protocol. . . . .	ip(7)
semaphore set or shared/	ipcrm: remove a message queue, . . . . .	ipcrm(1)
communication facilities/	ipcs: report inter-process . . . . .	ipcs(1)
V/TAPE 3200 half-inch tape/	ipt: interface for Interphase . . . . .	ipt(7)
/islower, isupper, isalpha,	isalnum, isspace, iscntrl,/ . . . . .	ctype(3C)
/isxdigit, islower, isupper,	isalpha, isalnum, isspace,/ . . . . .	ctype(3C)
/ispunct, isprint, isgraph,	isascii, tolower, toupper,/ . . . . .	ctype(3C)
terminal. ttyname,	isatty: find name of a . . . . .	ttyname(3C)
/isalpha, isalnum, isspace,	iscntrl, ispunct, isprint,/ . . . . .	ctype(3C)
isupper, isalpha, isalnum/	isdigit, isxdigit, islower, . . . . .	ctype(3C)
/iscntrl, ispunct, isprint,	isgraph, isascii, tolower,/ . . . . .	ctype(3C)
isalnum/ isdigit, isxdigit,	islower, isupper, isalpha, . . . . .	ctype(3C)
for floating point NaN/	isnan: isnand, isnanf: test . . . . .	isnan(3C)
floating point NaN/ isnan:	isnanand, isnanf: test for . . . . .	isnan(3C)
port NaN/ isnan: isnand,	isnananf: test for floating . . . . .	isnan(3C)
/isspace, iscntrl, ispunct,	isprint, isgraph, isascii,/ . . . . .	ctype(3C)
/isalnum, isspace, iscntrl,	ispunct, isprint, isgraph/ . . . . .	ctype(3C)
/isupper, isalpha, isalnum,	isspace, iscntrl, ispunct,/ . . . . .	ctype(3C)
system:	issue a shell command. . . . .	system(3S)
	issue: issue identification file. . . . .	issue(4)
isdigit, isxdigit, islower,	isupper, isalpha, isalnum/ . . . . .	ctype(3C)
isalpha, isalnum/ isdigit,	isxdigit, islower, isupper, . . . . .	ctype(3C)
news: print news	items. . . . .	news(1)
volume.	iv: initialize and maintain . . . . .	iv(1)
functions. bessel:	j0, j1, jn, y0, y1, yn: Bessel . . . . .	bessel(3M)
functions. bessel: j0,	j1, jn, y0, y1, yn: Bessel . . . . .	bessel(3M)
bj: the game of black	jack. . . . .	bj(6)
functions. bessel: j0, j1,	jn, y0, y1, yn: Bessel . . . . .	bessel(3M)
operator.	join: relational database . . . . .	join(1)
/rand48, nrand48, mrand48,	jrand48, srand48, seed48/ . . . . .	drand48(3C)
mkdbsym: load symbols in	kernel debugger. . . . .	mkdbsym(1M)
port. dbconsole: change the	kernel debugger system console . . . . .	dbconsole(1M)
makekey: generate encryption	key. . . . .	makekey(1)
a CTIX system command using	keywords. locate: identify . . . . .	locate(1)
killall:	kill all active processes. . . . .	killall(1M)
process or a group of/	kill: send a signal to a . . . . .	kill(2)
	kill: terminate a process. . . . .	kill(1)
processes.	killall: kill all active . . . . .	killall(1M)
mem,	kmem: system memory interface. . . . .	mem(7)
quiz: test your	knowledge. . . . .	quiz(6)
3-byte integers and long/	l3tol, ltol3: convert between . . . . .	l3tol(3C)
integer and base-64/ a64l,	l64a: convert between long . . . . .	a64l(3C)
labelit: provide	labels for file systems. . . . .	labelit(1M)
scanning and processing	language. awk: pattern . . . . .	awk(1)
arbitrary-precision arithmetic	language. bc: . . . . .	bc(1)
efl: extended FORTRAN	language. . . . .	efl(1)
scanning and processing	language. nawk: pattern . . . . .	nawk(1)
cpp: the C	language preprocessor. . . . .	cpp(1)
files. includes: determine C	language preprocessor include . . . . .	includes(1)
command programming	language. /standard/restricted . . . . .	sh(1)
cftime:	language specific strings. . . . .	cftime(4)
chargefee, ckpacct, dodisk,	lastlogin, monacct, nulladm,/ . . . . .	acctsh(1M)
shl: shell	layer manager. . . . .	shl(1)
/setspent, endspent, fgetspent,	lckpwwf, ulckpwwf: get shadow. . . . .	getspent(3X)
/jrand48, srand48, seed48,	lcong48: generate uniformly/ . . . . .	drand48(3C)
object files.	ld: link editor for common . . . . .	ld(1)

object file.	ldclose,	ldaclose: close a common	ldclose(3X)
header of a member of an/ file for reading.	ldopen,	ldahread: read the archive	ldahread(3X)
common object file.	ldclose, ldaclose:	open a common object	ldopen(3X)
drivers.	lddrv:	manage loadable	lddrv(1M)
		lddeeprom: load EEPROM.	lddeeprom(1M)
of floating-point/ frexp, access routines.	ldexp, modf:	manipulate parts	frexp(3C)
of a common object file.	ldfcn:	common object file	ldfcn(4)
name for common object file/ line number entries/	ldhread:	read the file header	ldhread(3X)
number/	ldlread, ldlimit,	retrieve symbol	ldgetname(3X)
number/	ldlread, ldlimit,	ldlitem: manipulate	ldlread(3X)
manipulate line number/	ldlread, ldlimit,	ldlitem: manipulate line	ldlread(3X)
line number entries of a/ entries of a section/	ldlseek, ldnlseek:	seek to	ldlseek(3X)
entries of a section/	ldnrseek:	seek to line number	ldlseek(3X)
indexed/named/	ldshread, ldnsread:	seek to relocation	ldrseek(3X)
indexed/named/	ldsseek, ldnsseek:	read an	ldshread(3X)
file header of a common/ object file for reading.	ldohseek:	seek to an	ldsseek(3X)
relocation entries of a/ indexed/named section header/	ldopen, ldaopen:	seek to the optional	ldohseek(3X)
socket configuration.	ldrseek, ldnrseek:	open a common	ldopen(3X)
indexed/named section of a/ of a symbol table entry of a/ symbol table entry of a/ table of a common object/ getopt:	ldshread, ldnsread:	seek to	ldrseek(3X)
generate programs for simple update.	ldsocket:	ldshread: read an	ldshread(3X)
update.	ldsocket:	STREAMS linker, load	slink(1)
Blocks (VIIB).	ldsseek, ldnsseek:	seek to an	ldsseek(3X)
introduction to functions and chkshlib: compare shared relation for an object portable/	ldtindex:	compute the index	ldtindex(3X)
mkshlib: create a shared t_alloc: allocate a t_free: free a t_sync: synchronize transport implementation-speci fic/ ulimit: get and set user an out-going terminal type, modes, speed, and type, modes, speed, and slipd: switched Serial line: read one common object file.	ldtindex:	ldtindex: read an indexed	ldtindex(3X)
/ldlimit, ldlimit: manipulate ldlseek, ldnlseek: seek to strip: strip symbol and nl: out selected fields of each send/cancel requests to an LP lpset: set parallel lpr: lsearch, lfind: col: filter reverse	ldtbread:	seek to the symbol	ldtbread(3X)
	letter from argument vector.		getopt(3C)
	lexical tasks.	lex:	lex(1)
	lfind: linear search and		lsearch(3C)
	libdev: manipulate Volume Home		libdev(3X)
	libraries.	intro:	intro(3)
	libraries tool.		chkshlib(1)
	library. /find ordering		lorder(1)
	library maintainer for		ar(1)
	library.		mkshlib(1)
	library structure.		t_alloc(3n)
	library structure.		t_free(3n)
	library.		t_sync(3n)
	limits: file header for		limits(4)
	limits.		ulimit(2)
	line connection. /establish		dial(3C)
	line discipline. /set terminal		getty(1M)
	line discipline. /set terminal		uugetty(1M)
	Line Internet Protocol control/ line.		slipd(1M)
	line.		line(1)
	line number entries in a		linenum(4)
	line number entries of a/ line number entries of a/ line number information from a/ strip: strip symbol and nl: out selected fields of each send/cancel requests to an LP lpset: set parallel lpr: lsearch, lfind: col: filter reverse		ldlread(3X)
			ldlseek(3X)
			strip(1)
			nl(1)
			cut(1)
			lp(1)
			lpset(1M)
			lpr(1)
			line(1)
			lsearch(3C)
			col(1)



in a common object file.	linenum: line number entries . . . . .	linenum(4)
/attach and detach serial files.	comm: select or reject	slattach(1M)
file for uucp communications	lines common to two sorted	comm(1)
device. fold: fold long	lines. Devices: configuration	Devices(5)
head: give first few	lines for finite width output	fold(1)
uniq: report repeated	lines. . . . .	head(1)
subsequent/ paste: merge same	lines in a file. . . . .	uniq(1)
directories. link, unlink:	lines of several files or	paste(1)
files. ld:	link and unlink files and	link(1M)
a.out: common assembler and	link editor for common object	ld(1)
	link editor output. . . . .	a.out(4)
	link: link to a file. . . . .	link(2)
cp, ln, mv: copy,	link, or move files. . . . .	cp(1)
link:	link to a file. . . . .	link(2)
slink, ldsocket: STREAMS	linker, load socket/	slink(1)
lists from proto file; set	links based on. /out file	qlist(1)
	lint: a C program checker. . . . .	lint(1)
ls:	list contents of directory. . . . .	ls(1)
nlist: get entries from name	list. . . . .	nlist(3C)
and statistics for file system	list file names . . . . .	ff(1M)
an. bcheck: print the	list of blocks associated with	bcheck(1M)
nm: print name	list of common object file. . . . .	nm(1)
by fsck and/ checklist:	list of file systems processed	checklist(4)
hosts:	list of hosts on network. . . . .	hosts(4)
protocols:	list of Internet protocols. . . . .	protocols(4)
services:	list of Internet services. . . . .	services(4)
terminal number. ttytype:	list of terminal types by	ttytype(4)
from a common object file.	list: produce C source listing	list(1)
handle variable argument	list. varargs: . . . . .	varargs(5)
output of a varargs argument	list. /printf formatted	vprintf(3S)
t_listen:	listen for a connect request. . . . .	t_listen(3n)
socket. listen:	listen for connections on a	listen(2)
data passed through the	listener. /get client's	nlsgetcall(3n)
nlsadmin: network	listener service/ . . . . .	nlsadmin(1M)
nlsrequest: format and send	listener service request/	nlsrequest(3n)
file. list: produce C source	listing from a common object	list(1)
xargs: construct argument	list(s) and execute command. . . . .	xargs(1)
links/ qlist: print out file	lists from proto file; set	qlist(1)
volcopy: make	literal copy of file system. . . . .	volcopy(1M)
files. cp,	ln, mv: copy, link, or move	cp(1)
interface.	lo: software loopback network	lo(7)
ldeeprom:	load EEPROM. . . . .	ldeeprom(1M)
/ldsocket: STREAMS linker,	load socket configuration. . . . .	slink(1)
debugger. mkdbsym:	load symbols in kernel . . . . .	mkdbsym(1M)
drivers:	loadable device drivers. . . . .	drivers(7)
lddrv: manage	loadable drivers. . . . .	lddrv(1M)
cftime, asctime/ ctime,	localtime, gmtime, asctime,	ctime(3C)
the virtual system/ conlocate:	locate a terminal to use as	conlocate(1M)
command. path:	locate executable file for	path(1)
command using keywords.	locate: identify a CTIX system	locate(1)
end, etext, edata: last	locations in program. . . . .	end(3C)
memory. plock:	lock process, text, or data in	plock(2)
files.	lockf: record locking on	lockf(3C)
regions of a file.	locking: exclusive access to	locking(2)
lockf: record	locking on files. . . . .	lockf(3C)
gamma:	log gamma function. . . . .	gamma(3M)
newgrp:	log in to a new group. . . . .	newgrp(1M)
error logging and event/	log: interface to STREAMS	log(7)

exponential, logarithm/ exp,	log, log10, pow, sqrt:	exp(3M)
/usr/adm/loginlog:	log of failed login attempts.	loginlog(4)
logarithm, power./ exp, log,	log10, pow, sqrt: exponential,	exp(3M)
/log10, pow, sqrt: exponential,	logarithm, power, square root/	exp(3M)
errpt: process a report of	logged errors.	errpt(1M)
rwho: who is	logged in on local network.	rwho(1)
strclean: STREAMS error	logger cleanup program.	strclean(1M)
strerr: STREAMS error	logger daemon.	strerr(1M)
/interface to STREAMS error	logging and event tracing.	log(7)
/log of failed	login attempts.	loginlog(4)
networks. netrc:	login file for remote	netrc(4)
getlogin: get	login name.	getlogin(3C)
logname: get	login name.	logname(1)
cuserid: get character	login name of the user.	cuserid(3S)
logname: return	login name of user.	logname(3X)
passwd: change	login password.	passwd(1)
rlogin: remote	login.	rlogin(1)
rlogind: remote	login server.	rlogind(1M)
	login: sign on.	login(1)
up a C shell environment at	login time. cprofile: setting	cprofile(4)
setting up an environment at	login time. profile:	profile(4)
	logname: get login name.	logname(1)
user.	logname: return login name of	logname(3X)
a64l, l64a: convert between	long integer and base-64 ASCII/	a64l(3C)
sputl, sgetl: access	long integer data in a/	sputl(3X)
between 3-byte integers and	long integers. /l3tol3: convert	l3tol(3C)
output device. fold: fold	long lines for finite width	fold(1)
setjmp,	longjmp: non-local goto.	setjmp(3C)
finger: user information	lookup program.	finger(1)
lo: software	loopback network interface.	lo(7)
for an object library.	lorder: find ordering relation	lorder(1)
mklost+found: make a	lost+found directory for fsck.	mklostfnd(1M)
nice: run a command at	low priority.	nice(1)
send/cancel requests to an	LP line printer. lp, cancel:	lp(1)
interface.	lp: parallel printer	lp(7)
disable: enable/disable	LP printers. enable,	enable(1)
reject: allow or prevent	LP requests. accept,	accept(1M)
/lpshut, lpmove: start/stop the	LP scheduler and move/	lpsched(1M)
lpadmin: configure the	LP spooling system.	lpadmin(1M)
lpstat: print	LP status information.	lpstat(1)
spooling system.	lpadmin: configure the LP	lpadmin(1M)
scheduler/ lpsched, lpshut,	lpmove: start/stop the LP	lpsched(1M)
	lpr: line printer spooler.	lpr(1)
start/stop the LP scheduler/	lpsched, lpshut, lpmove:	lpsched(1M)
printer options.	lpset: set parallel line	lpset(1M)
LP scheduler and/ lpsched,	lpshut, lpmove: start/stop the	lpsched(1M)
information.	lpstat: print LP status	lpstat(1)
jranda48/ dranda48, eranda48,	lranda48, nranda48, mrand48,	dranda48(3C)
directory.	ls: list contents of	ls(1)
and update.	lsearch, lfind: linear search	lsearch(3C)
pointer.	lseek: move read/write file	lseek(2)
integers and long/ l3tol,	l3tol3: convert between 3-byte	l3tol(3C)
	m4: macro processor.	m4(1)
mega, unixpc,	machid: mc68k, miti, mini,	machid(1)
values:	machine-dependent values.	values(5)
/access long integer data in a	machine-independent fashion.	sputl(3X)
permuted index. mptx: the	macro package for formatting a	mptx(5)
documents. mm: the MM	macro package for formatting	mm(5)

view graphs and/	mv: a troff macro package for typesetting	mv(5)
	m4: macro processor	m4(1)
pages. man:	macros for formatting manual	man(5)
	me: macros for formatting papers	me(5)
formatted with the MM	macros. /print/check documents	mm(1)
ms: text formatting	macros	ms(5)
/rebuild the data base for the	mail aliases file	newaliases(1)
users or read mail.	mail, rmail: send mail to	mail(1)
	sendmail: mail routing program	sendmail(1M)
processing system.	mailx: interactive message	mailx(1)
malloc, free, realloc, calloc:	main memory allocator	malloc(3C)
/malloc, mallinfo: fast	main memory allocator	malloc(3X)
regenerate groups of/	make: maintain, update, and	make(1)
iv: initialize and	maintain volume	iv(1)
ar: archive and library	maintainer for portable/	ar(1)
SCCS file. delta:	make a delta (change) to an	delta(1)
	mkdir: make a directory	mkdir(2)
or ordinary file. mknod:	make a directory, or a special	mknod(2)
for fsck. mklost+found:	make a lost+found directory	mklostfnd(1M)
	mktemp: make a unique file name	mktemp(3C)
file. mkifile:	make an ifile from an object	mkifile(1M)
Facility database. helpadm:	make changes to the Help	helpadm(1M)
	mkdir, mkdirs: make directories	mkdir(1)
system. volcopy:	make literal copy of file	volcopy(1M)
regenerate groups of/	make: maintain, update, and	make(1)
mkhosts:	make node name commands	mkhosts(1M)
	banner: make posters	banner(1)
session. script:	make typescript of terminal	script(1)
key.	makekey: generate encryption	makekey(1)
/realloc, calloc, malloc,	mallinfo: fast main memory/	malloc(3X)
main memory allocator.	malloc, free, realloc, calloc:	malloc(3C)
malloc, mallinfo: fast main/	malloc, free, realloc, calloc,	malloc(3X)
malloc, free, realloc, calloc,	malloc, mallinfo: fast main/	malloc(3X)
manual pages.	man: macros for formatting	man(5)
/find, tdelete, twalk:	manage binary search trees	tsearch(3C)
hsearch, hcreate, hdestroy:	manage hash search tables	hsearch(3C)
	lddrv: manage loadable drivers	lddrv(1M)
unnotify, evwait, evnowait:	manage notifications. notify,	notify(2)
endpoint. t_optmgmt:	manage options for a transport	t_optmgmt(3n)
passmgmt: password files	management	passmgmt(1M)
window: window	management primitives	window(7)
sigignore, sigpause: signal	management. /sigrelse,	sigset(2)
wm: window	management	wm(1)
shl: shell layer	manager	shl(1)
records. fwtmp, wtmpfix:	manipulate connect accounting	fwtmp(1M)
of/ ldread, ldlimit, ldlimit:	manipulate line number entries	ldread(3X)
frexp, ldexp, modf:	manipulate parts of/	frexp(3C)
comment section. mcs:	manipulate the object file	mcs(1)
route: manually	manipulate the routing tables	route(1M)
(VHB). libdev:	manipulate Volume Home Blocks	libdev(3X)
/inet_netof: Internet address	manipulation routines	inet(3)
man: macros for formatting	manual pages	man(5)
routing tables. route:	manually manipulate the	route(1M)
terminal input and/ rsterm:	manually start and stop	rsterm(1M)
	ascii: map of ASCII character set	ascii(5)
port to RPC program number	mapper. portmap: DARPA	portmap(1M)
File Sharing user and group	mapping. idload: Remote	idload(1M)
scsimap: set	mappings for SCSI devices	scsimap(1M)

files. diffmk:	mark differences between	diffmk(1)
umask: set file-creation mode	mask.	umask(1)
set and get file creation	mask. umask:	umask(2)
table. master:	master device information	master(4)
File Sharing name server	master file. rfmaster: Remote	rfmaster(4)
information table.	master: master device	master(4)
regular expression compile and	match routines. regexp:	regexp(5)
math:	math functions and constants.	math(5)
constants.	math: math functions and	math(5)
eqn, neqn, checkeq: format	mathematical text for nroff or/	eqn(1)
function.	matherr: error-handling	matherr(3M)
maze: generate a	maze.	maze(6)
unixpc., machid:	mc68k, miti, mini, mega,	machid(1)
file comment section.	mcs: manipulate the object	mcs(1)
machid: mc68k, miti, mini,	mega, unixpc.	machid(1)
interface.	mem, kmem: system memory	mem(7)
memcpy, memset:/ memory:	memcpy, memchr, memcmp,	memory(3C)
memset:/ memory: memcpy,	memchr, memcmp, memcpy,	memory(3C)
memory: memcpy, memchr,	memcmp, memcpy, memset: memory/	memory(3C)
/memcpy, memchr, memcmp,	memcpy, memset: memory/	memory(3C)
free, realloc, calloc: main	memory allocator. malloc,	malloc(3C)
malloc, mallinfo: fast main	memory allocator. /calloc,	malloc(3X)
shmctl: shared	memory control operations.	shmctl(2)
queue, semaphore set or shared	memory ID. /remove a message	ipcrm(1)
mem, kmem: system	memory interface.	mem(7)
memcpy, memcpy, memset:/	memory: memcpy, memchr,	memory(3C)
memcpy, memcpy, memset:	memory operations. /memchr,	memory(3C)
shmop: shared	memory operations.	shmop(2)
lock process, text, or data in	memory. plock:	plock(2)
shmget: get shared	memory segment identifier.	shmget(2)
/memchr, memcmp, memcpy,	memset: memory operations.	memory(3C)
astgen: generate/modify ASSIST	menus and command forms.	astgen(1)
sort: sort and/or	merge files.	sort(1)
files. acctmrg:	merge or add total accounting	acctmrg(1M)
files or subsequent/ paste:	merge same lines of several	paste(1)
	msg: permit or deny messages.	msg(1)
msgctl:	message control operations.	msgctl(2)
recv, recvfrom: receive a	message from a socket.	recv(2)
send listener service request	message. /format and	nlrequest(3n)
getmsg: get next	message off a stream.	getmsg(2)
putmsg: send a	message on a stream.	putmsg(2)
msgop:	message operations.	msgop(2)
mailx: interactive	message processing system.	mailx(1)
icmp: Internet Control	Message Protocol.	icmp(7)
msgget: get	message queue.	msgget(2)
or shared/ ipcrm: remove a	message queue, semaphore set	ipcrm(1)
t_error: produce error	message.	t_error(3n)
send, sendto: send a	message to a socket.	send(2)
msg: permit or deny	messages.	msg(1)
sys_err: system error	messages. /errno, sys_errlist,	perror(3C)
strace: print STREAMS trace	messages.	strace(1M)
machid: mc68k, miti,	mini, mega, unixpc.,	machid(1)
driver. clone: open any	minor device on a STREAMS	clone(7)
machid: mc68k,	miti, mini, mega, unixpc.,	machid(1)
kernel debugger.	mkdbsym: load symbols in	mkdbsym(1M)
	mkdir: make a directory.	mkdir(2)
directories.	mkdir, makedirs: make	mkdir(1)
	mkfs: construct a file system.	mkfs(1M)

/and verify software using the	mkfs(1) proto file database.	ginstall(1)
commands.	mkhosts: make node name	mkhosts(1M)
object file.	mkifile: make an ifile from an	mkifile(1M)
lost+found directory for/	mklost+found: make a	mklostfnd(1M)
special or ordinary file.	mknod: build special file.	mknod(1M)
library.	mkshlib: create a shared	mkshlib(1)
name.	mktemp: make a unique file	mktemp(3C)
relocate a PT or GT local/	mktpy, mvtpy: install or	mktpy(1)
documents formatted with the/	mm, checkmm: print/check	mm(1)
formatting documents. mm: the	MM macro package for	mm(5)
documents formatted with the	MM macros. /print/check	mm(1)
view graphs, and slides.	mmt, mvt: typeset documents,	mmt(1)
table.	mnttab: mounted file system	mnttab(4)
chmod: change	mode.	chmod(1)
umask: set file-creation	mode mask.	umask(1)
chmod: change	mode of file.	chmod(2)
getty: set terminal type,	modes, speed, and line/	getty(1M)
uugetty: set terminal type,	modes, speed, and line/	uugetty(1M)
bs: a compiler/interpreter for	modest-sized programs.	bs(1)
floating-point/ frexp, ldexp,	modf: manipulate parts of	frexp(3C)
touch: update access and	modification times of a file.	touch(1)
utime: set file access and	modification times.	utime(2)
Interface cooperating STREAMS	module. timod: Transport	timod(7)
read/write interface STREAMS	module. /Transport Interface	tirdwr(7)
/ckpacct, dodisk, lastlogin,	monacct, nulladm, prctmp,/	acctsh(1M)
profile.	monitor: prepare execution	monitor(3C)
mount:	moo: guessing game.	moo(6)
and remote/ mount, umount:	more, page: text perusal.	more(1)
mnttry: attempt to	mount a file system.	mount(2)
mountd: NFS	mount and unmount file systems	mount(1M)
setmnt: establish	mount remote resources.	mnttry(1M)
systems. mountall, umountall:	mount request server.	mountd(1M)
System/ nmountall, numountall:	mount table.	setmnt(1M)
mountall, rumountall:	mount, unmount multiple file	mountall(1M)
unmount multiple file/	mount, unmount Network File	nmountall(1M)
server.	mount, unmount Remote File/	rmountall(1M)
mnttab:	mountall, umountall: mount,	mountall(1M)
rmntstat: remotely	mountd: NFS mount request	mountd(1M)
rmntstat: display	mounted file system table.	mnttab(4)
mount: queue remote resource	mounted file system table.	mnttab(4)
showmount: show all remote	mounted resource information.	rmntstat(1M)
mvdire	mounts.	rmount(1M)
cp, ln, mv: copy, link, or	mounts.	showmount(1M)
lseek:	move a directory.	mvdire(1M)
the LP scheduler and	move files.	cp(1)
formatting a permuted index.	move read/write file pointer.	lseek(2)
/erand48, lrand48, nrand48,	move requests. /start/stop	lpsched(1M)
operations.	mptx: the macro package for	mptx(5)
/umountall: mount, unmount	mrand48, jrand48, srand48,/	drand48(3C)
poll: STREAMS input/output	ms: text formatting macros.	ms(5)
select: synchronous I/O	msgctl: message control	msgctl(2)
sxt: STREAMS	msgget: get message queue.	msgget(2)
	msgop: message operations.	msgop(2)
	multiple file systems.	mountall(1M)
	multiplexing.	poll(2)
	multiplexing.	select(2)
	multiplexor.	sxt(7)

run commands performed for	multi-user environment. /rc3:	rc2(1M)
typesetting view graphs and/	mv: a troff macro package for	mv(5)
cp, ln,	mv: copy, link, or move files.	cp(1)
	mvdir: move a directory.	mvdir(1M)
graphs, and slides. mmt,	mvt: typeset documents, view	mmt(1)
PT or GT local/ mktpy,	mvtpy: install or relocate a	mktpy(1)
server.	named: Internet domain name	named(1M)
test for floating point	NaN (Not-A-Number). /isnanf:	isnan(3C)
processing language.	nawk: pattern scanning and	nawk(1)
systems processed by fsck and	ncheck. /list of file	checklist(4)
from i-numbers.	ncheck: generate path names	ncheck(1M)
mathematical text for/ eqn,	neqn, checkeq: format	eqn(1)
definitions for eqn and	neqn. /special character	eqnchar(5)
File.	netcf: Network Configuration	netcf(4)
networks.	netrc: login file for remote	netrc(4)
	netstat: show network status.	netstat(1)
host. getservaddr: get	network address of service	getservad(1M)
values between host and	network byte order. /convert	byteorder(3)
netcf:	Network Configuration File.	netcf(4)
setnetent, endnetent: get	network entry. /getnetbyname,	getnetent(3)
/numountall: mount, unmount	Network File System resources.	nmountall(1M)
statistics. nfsstat:	Network File System	nfsstat(1M)
/sethostent, endhostent: get	network host entry.	gethostbyname(3)
ICMP ECHO_REQUEST packets to	network hosts. ping: send	ping(1M)
hosts: list of hosts on	network.	hosts(4)
lo: software loopback	network interface.	lo(7)
ifconfig: configure	network interface parameters.	ifconfig(1M)
and detach serial lines as	network interfaces. /attach	slattach(1M)
administration. nlsadmin:	network listener service	nlsadmin(1M)
Remote File Sharing domain and	network names. dname: print	dname(1M)
routed:	network routing daemon.	routed(1M)
status of nodes on local	network. ruptime: display	ruptime(1)
who is logged in on local	network. rwho:	rwho(1)
netstat: show	network status.	netstat(1)
commands. stat: statistical	network useful with graphical	stat(1G)
uucpd, ouucpd:	network uucp servers.	uucpd(1M)
for the internet.	networks: names and numbers	networks(4)
netrc: login file for remote	networks.	netrc(4)
base for the mail aliases/	newaliases: rebuild the data	newaliases(1)
a text file.	newform: change the format of	newform(1)
	newgrp: log in to a new group.	newgrp(1M)
news: print	news items.	news(1)
/store, delete, firstkey,	nextkey: database subroutines.	dbm(3X)
nfsd, biod:	NFS daemons.	nfsd(1M)
configuration file. exports:	NFS file systems export	exports(4)
mountd:	NFS mount request server.	mountd(1M)
nfsysys: common shared	NFS system calls.	nfsysys(2)
	nfsd, biod: NFS daemons.	nfsd(1M)
statistics.	nfsstat: Network File System	nfsstat(1M)
system calls.	nfsysys: common shared NFS	nfsysys(2)
process.	nice: change priority of a	nice(2)
of running process by changing	nice. renice: alter priority	renice(1)
priority.	nice: run a command at low	nice(1)
	nl: line numbering filter.	nl(1)
list.	nlist: get entries from name	nlist(3C)
service administration.	nlsadmin: network listener	nlsadmin(1M)
passed through the listener.	nlsgetcall: get client's data	nlsgetcall(3n)
transport provider.	nlsprovider: get name of	nlsprovider(3n)

listener service request/ object file.	nlsrequest: format and send . . . . .	nlsrequest(3n)
umount Network File System/ mkhosts: make createdev: create device runtime: display status of hangups and quits. setjmp, longjmp: test for floating point NaN	nm: print name list of common . . . . . nmountall, numountall: mount, node name commands. . . . . nodes for assorted device/ . . . . . nodes on local network. . . . . nohup: run a command immune to non-local goto. . . . . (Not-A-Number). /isnanf: . . . . .	nm(1) nmountall(1M) mkhosts(1M) createdev(1M) runtime(1) nohup(1) setjmp(3C) isnan(3C)
rfsuadmin: Remote File Sharing evwait, evnowait: manage evnowait: manage/ drand48, erand48, lrand48,	notification shell script. . . . . notifications. /unnotify, . . . . . notify, unnotify, evwait, . . . . . nrand48, mrand48, jrand48./ nroff: format text. . . . .	rfsuadmin(1M) notify(2) notify(2) drand48(3C) nroff(1)
format mathematical text for tbl: format tables for constructs. deroff: remove name server query. between host/ htonl, htons, host and/ htonl, htons, ntohl, null: the /dodisk, lastlogin, monacct, nl: line number: convert Arabic graphics: access graphical and Network File/ nmountall, dis: ldfcn: common mcs: manipulate the conv: common cprs: compress a common dump selected parts of an ldopen, ldaopen: open a common number entries of a common ldaclose: close a common the file header of a common of a section of a common file header of a common of a section of a common section header of a common section of a common symbol table entry of a common symbol table entry of a common the symbol table of a common number entries in a common C source listing from a common mkfile: make an ifile from an nm: print name list of common information for a common section header for a common information from a common entry. /symbol name for common format. syms: common file header for common directories. cpset: install ld: link editor for common sizes in bytes of common find ordering relation for an	nroff or troff. /checked: . . . . . nroff or troff. . . . . nroff/troff, tbl, and eqn . . . . . nsquery: Remote File Sharing . . . . . ntohl, ntohs: convert values . . . . . ntohs: convert values between . . . . . null file. . . . . nulladm, prctmp, prdaily./ numbering filter. . . . . numerals to English. . . . . numerical commands. . . . . numountall: mount, unmount object code disassembler. . . . . object file access routines. . . . . object file comment section. . . . . object file converter. . . . . object file. . . . . object file. dump: . . . . . object file for reading. . . . . object file function. /line . . . . . object file. ldclose, . . . . . object file. ldfhread: read . . . . . object file. /number entries . . . . . object file. /to the optional . . . . . object file. /entries . . . . . object file. /indexed/named . . . . . object file. /indexed/named . . . . . object file. /the index of a . . . . . object file. /read an indexed . . . . . object file. /seek to . . . . . object file. linenum: line . . . . . object file. list: produce . . . . . object file. . . . . object file. /relocation . . . . . object file. scnhdr: . . . . . object file. /and line number . . . . . object file symbol table . . . . . object file symbol table . . . . . object files. filehdr: . . . . . object files in binary . . . . . object files. . . . . object files. /print section . . . . . object library. lorder: . . . . .	deroff(1) nsquery(1M) byteorder(3) byteorder(3) null(7) acctsh(1M) nl(1) number(6) graphics(1G) nmountall(1M) dis(1) ldfcn(4) mcs(1) conv(1) cprs(1) dump(1) ldopen(3X) ldlread(3X) ldclose(3X) ldfhread(3X) ldlseek(3X) ldohseek(3X) ldrseek(3X) ldshread(3X) ldsseek(3X) ldtbindex(3X) ldtbread(3X) ldtbseek(3X) linenum(4) list(1) mkfile(1M) nm(1) reloc(4) scnhdr(4) strip(1) ldgetname(3X) syms(4) filehdr(4) cpset(1M) ld(1) size(1) lorder(1)

number.	factor:	obtain the prime factors of a	factor(1)
	occ:	occ command	occ(1)
	od:	octal dump.	od(1)
functions.	ocurse:	optimized screen	ocurse(3X)
	od:	octal dump.	od(1)
	occ:	old C compiler.	occ(1)
query Remote I/O Processor for	riopqry:	online data.	riopqry(1M)
reading.	ldopen, ldaopen:	open a common object file for	ldopen(3X)
	fopen, freopen, fdopen:	open a stream.	fopen(3S)
STREAMS driver.	clone:	open any minor device on a	clone(7)
	dup:	duplicate an	dup(2)
	dup2:	duplicate an	dup2(3C)
	open:	open for reading or writing.	open(2)
	seekdir,/	directory:	opendir, readdir, telldir, directory(3X)
starter: information about the		operating system for beginning/	starter(1)
	prf:	operating system profiler.	prf(7)
/prfdc, prfsnap, prfpr:		operating system profiler.	profiler(1M)
commands performed to stop the		operating system.	rc0: run rc0(1M)
	uconf:	configure the	uconf(1M)
bzero: bit and byte string	operations.	bcopy, bcmp,	bstring(3)
rewinddir, closedir: directory	operations.	helldir, seekdir,	directory(3X)
memcmp, memcpy, memset: memory	operations.	memccpy, memchr,	memory(3C)
msgctl: message control	operations.		msgctl(2)
	msgop:	message	msgop(2)
tputs: terminal independent	operations.	agetstr, tgoto,	otermcap(3X)
semctl: semaphore control	operations.		semctl(2)
	semop:	semaphore	semop(2)
shmctl: shared memory control	operations.		shmctl(2)
	shmop:	shared memory	shmop(2)
	strespn, strtok:	string	string(3C)
join: relational database	operator.		join(1)
dcopy: copy file systems for		optimal access time.	dcopy(1M)
terminal screen handling and		optimization package.	curses: curses(3X)
	ocurse:	optimized screen functions.	ocurse(3X)
	vector.	getopt: get	option letter from argument
common/ ldohseek: seek to the		optional file header of a	ldohseek(3X)
	fcntl:	file control	options. fcntl(5)
	stty:	set the	options for a terminal. stty(1)
endpoint.	t_optmgmt:	manage	options for a transport t_optmgmt(3n)
	getopt:	parse command	options. getopt(1)
	getoptcv:	parse command	options. getopt(1)
	set parallel line printer	options.	lpset(1M)
/setsockopt: get and set		options on sockets.	getsockopt(2)
object library.	lorder:	find	ordering relation for an lorder(1)
/acknowledge receipt of an		orderly release indication.	t_rcvrel(3n)
	t_sndrel:	initiate an	orderly release. t_sndrel(3n)
a directory, or a special or		ordinary file.	mknod: make mknod(2)
keywords.	locate:	identify a	CTIX system command using locate(1)
	assist:	assistance using	CTIX system commands. assist(1)
	help:	CTIX system Help Facility.	help(1)
uname: print name of current		CTIX system.	uname(1)
	dial:	establish an	out-going terminal line/ dial(3C)
assembler and link editor		output.	a.out: common a.out(4)
long lines for finite width		output device.	fold: fold fold(1)
/vsprintf: print formatted		output of a varargs argument/	vprintf(3S)
sprintf: print formatted		output.	printf, fprintf printf(3S)
and stop terminal input and		output.	/manually start rsterm(1M)
	sysdef:	output system definition.	sysdef(1M)



	uucpd,	ouucpd: network uucp servers. . . . .	uucpd(1M)
/acctdusg, accton, acctwtmp:	overview of accounting and/	. . . . .	acct(1M)
chown, chgrp: change	owner and group of a file. . . . .		chown(2)
chown, chgrp: change	owner or group. . . . .		chown(1)
and expand files.	pack, pcat, unpack: compress	. . . . .	pack(1)
handling and optimization	package. /terminal screen . . . . .		curses(3X)
permuted/ mptx: the macro	package for formatting a	. . . . .	mptx(5)
documents. mm: the MM macro	package for formatting	. . . . .	mm(5)
graphs and/ mv: a troff macro	package for typesetting view	. . . . .	mv(5)
sadc: system activity report	package. sar: sa1, sa2, . . . . .		sar(1M)
standard buffered input/output	package. stdio: . . . . .		stdio(3S)
interprocess communication	package. /flok: standard	. . . . .	stdipc(3C)
ping: send ICMP ECHO_REQUEST	packets to network hosts. . . . .		ping(1M)
more,	page: text perusal. . . . .		more(1)
macros for formatting manual	pages. man: . . . . .		man(5)
4014 terminal. 4014:	paginator for the Tektronix	. . . . .	4014(1)
me: macros for formatting	papers. . . . .		me(5)
lpset: set	parallel line printer options.	. . . . .	lpset(1M)
lp:	parallel printer interface. . . . .		lp(7)
tapeset: set drive	parameters for tape/ . . . . .		tapeset(1M)
configure network interface	parameters. ifconfig: . . . . .		ifconfig(1M)
process, process group, and	parent process IDs. /get . . . . .		getpid(2)
getopt:	parse command options. . . . .		getopt(1)
getopts, getoptcv:	parse command options. . . . .		getopts(1)
nlsgetcall: get client's data	passed through the listener. . . . .		nlsgetcall(3n)
management.	passmgmt: password files . . . . .		passmgmt(1M)
	passwd: change login password. . . . .		passwd(1)
	passwd: password file. . . . .		passwd(4)
functions. crypt:	password and file encryption . . . . .		crypt(3X)
/endpwent, fgetpwent: get	password file entry. . . . .		getpwent(3C)
putpwent: write	password file entry. . . . .		putpwent(3C)
putspent: write shadow	password file entry. . . . .		putspent(3X)
passwd:	password file. . . . .		passwd(4)
passmgmt:	password files management. . . . .		passmgmt(1M)
getpass: read a	password. . . . .		getpass(3C)
passwd: change login	password. . . . .		passwd(1)
Remote File Sharing host	password. rfpasswd: change . . . . .		rfpasswd(1M)
pwck, grpck:	password/group file checkers. . . . .		pwck(1M)
several files or subsequent/	paste: merge same lines of . . . . .		paste(1)
for command.	path: locate executable file . . . . .		path(1)
dimame: deliver portions of	path names. basename, . . . . .		basename(1)
ncheck: generate	path names from i-numbers. . . . .		ncheck(1M)
directory. getcwd: get	path-name of current working . . . . .		getcwd(3C)
grep: search a file for a	pattern. . . . .		grep(1)
processing language. awk:	pattern scanning and . . . . .		awk(1)
processing language. nawk:	pattern scanning and . . . . .		nawk(1)
egrep: search a file for a	pattern using full regular/ . . . . .		egrep(1)
signal.	pause: suspend process until . . . . .		pause(2)
expand files. pack,	pcat, unpack: compress and . . . . .		pack(1)
a process. popen,	pclose: initiate pipe to/from . . . . .		popen(3S)
get name of connected	peer. getpeername: . . . . .		getpeername(2)
rc2, rc3: run commands	performed for multi-user/ . . . . .		rc2(1M)
operating/ rc0: run commands	performed to stop the . . . . .		rc0(1M)
check the uucp directories and	permissions file. uucheck: . . . . .		uucheck(1M)
mesg:	permit or deny messages. . . . .		mesg(1)
macro package for formatting a	permuted index. mptx: the . . . . .		mptx(5)
ptx:	permuted index. . . . .		ptx(1)
format. acct:	per-process accounting file . . . . .		acct(4)

acctcms: command summary from	per-process accounting/	acctcms(1M)
sys_errr: system error/	peror, ermo, sys_errlist,	perorr(3C)
pg: file	perusal filter for CRTs.	pg(1)
more, page: text	perusal.	more(1)
CRTs.	pg: file perusal filter for	pg(1)
split: split a file into	pieces.	split(1)
packets to network hosts.	ping: send ICMP ECHO_REQUEST	ping(1M)
channel.	pipe: create an interprocess	pipe(2)
tee:	pipe fitting.	tee(1)
popen, pclose: initiate	pipe to/from a process.	popen(3S)
fish:	play "Go Fish".	fish(6)
data in memory.	plock: lock process, text, or	plock(2)
subroutines.	plot: graphics interface.	plot(4)
ftell: reposition a file	plot: graphics interface	plot(3X)
lseek: move read/write file	pointer in a stream. /rewind,	fseek(3S)
multiplexing.	pointer.	lseek(2)
to/from a process.	poll: STREAMS input/output	poll(2)
kernel debugger system console	popen, pclose: initiate pipe	popen(3S)
serstat: display serial	port. dbconsole: change the	dbconsole(1M)
getrpcport: get RPC	port error statistics.	serstat(1M)
mapper. portmap: DARPA	port number.	getrpcport(3)
and library maintainer for	port to RPC program number	portmap(1M)
basename, dimame: deliver	portable archives. /archive	ar(1)
program number mapper.	portions of path names.	basename(1)
banner: make	portmap: DARPA port to RPC	portmap(1M)
logarithm/ exp, log, log10,	posters.	banner(1)
/sqrt: exponential, logarithm,	pow, sqrt: exponential,	exp(3M)
brc, bcheckrc, drvload,	power, square root functions.	exp(3M)
	powerfail: system/	brc(1M)
	pr: print files.	pr(1)
/lastlogin, monacct, nulladm,	prctmp, prdaily, prtacct/	acctsh(1M)
/monacct, nulladm, prctmp,	prdaily, prtacct, runacct/	acctsh(1M)
for troff. cw, checkcw:	prepare constant-width text	cw(1)
monitor:	prepare execution profile.	monitor(3C)
cpp: the C language	preprocessor.	cpp(1)
includes: determine C language	preprocessor include files.	includes(1)
accept, reject: allow or	prevent LP requests.	accept(1M)
unget: undo a	previous get of an SCCS file.	unget(1)
profiler.	prf: operating system	prf(7)
profiler: prfld, prfstat,	prfdc, prfsnap, prfpr:/	profiler(1M)
prfstat, prfpr:/ profiler:	prfld, prfstat, prfdc,	profiler(1M)
/prfstat, prfdc, prfsnap,	prfpr: operating system/	profiler(1M)
system/ /prfld, prfstat, prfdc,	prfsnap, prfpr: operating	profiler(1M)
prfpr:/ profiler: prfld,	prfstat, prfdc, prfsnap,	profiler(1M)
factor: obtain the	prime factors of a number.	factor(1)
graphical/ gps: graphical	primitive string, format of	gps(4)
types:	primitive system data types.	types(5)
window: window management	primitives.	window(7)
interesting, adage. fortune:	print a random, hopefully	fortune(6)
prs:	print an SCCS file.	prs(1)
date:	print and set the date.	date(1)
cal:	print calendar.	cal(1)
of a file. sum:	print checksum and block count	sum(1)
editing activity. sact:	print current SCCS file	sact(1)
cat: concatenate and	print files.	cat(1)
pr:	print files.	pr(1)
vprintf, vfprintf, vsprintf:	print formatted output of a/	vprintf(3S)
printf, fprintf, sprintf:	print formatted output.	printf(3S)

host system. hostid:	set or print identifier of current	hostid(1)
lpstat:	print LP status information.	lpstat(1)
object file. nm:	print name list of common	nm(1)
system. uname:	print name of current CTIX	uname(1)
news:	print news items.	news(1)
proto file; set links/ qlist:	print out file lists from	qlist(1)
infocmp:	compare or print out terminfo/	infocmp(1M)
file(s). acctcom:	search and print process accounting	acctcom(1)
domain and network/ dname:	print Remote File Sharing	dname(1M)
of common object files. size:	print section sizes in bytes	size(1)
strace:	print STREAMS trace messages.	strace(1M)
of the/ hostname:	set or print the Internet host name	hostname(1)
associated with an. bcheck:	print the list of blocks	bcheck(1M)
names. id:	print user and group IDs and	id(1M)
formatted with/ mm, checkmm:	print/check documents	mm(1)
lp:	parallel printer interface.	lp(7)
requests to an LP line	printer. /cancel: send/cancel	lp(1)
or relocate a PT or GT local	printer. /mvtpy: install	mktpy(1)
lpset: set parallel line	printer options.	lpset(1M)
lpr: line	printer spooler.	lpr(1)
disable: enable/disable LP	printers. enable,	enable(1)
print formatted output.	printf, fprintf, sprintf:	printf(3S)
rtenable: real-time	priorities enabled/disabled.	rtenable(1M)
nice: run a command at low	priority.	nice(1)
nice: change	priority of a process.	nice(2)
changing nice. renice: alter	priority of running process by	renice(1)
errors. errpt:	process a report of logged	errpt(1M)
acct: enable or disable	process accounting.	acct(2)
acctprc1, acctprc2:	process accounting.	acctprc(1M)
acctcom: search and print	process accounting file(s).	acctcom(1)
alarm: set a	process alarm clock.	alarm(2)
times. times: get	process and child process	times(2)
/alter priority of running	process by changing nice.	renice(1)
init, telinit:	process control/	init(1M)
timex: time a command; report	process data and system/	timex(1)
exit, _exit: terminate	process.	exit(2)
fork: create a new	process.	fork(2)
/getpgrp, getppid: get process,	process group, and parent/	getpid(2)
setpgrp: set	process group ID.	setpgrp(2)
process group, and parent	process IDs. /get process,	getpid(2)
inittab: script for the init	process.	inittab(4)
kill: terminate a	process.	kill(1)
nice: change priority of a	process.	nice(2)
kill: send a signal to a	process or a group of/	kill(2)
initiate pipe to/from a	process. popen, pclose:	popen(3S)
getpid, getpgrp, getppid: get	process, process group, and/	getpid(2)
Remote File Sharing daemon	process. rfudaemon:	rfudaemon(1M)
ps: report	process status.	ps(1)
memory. plock: lock	process, text, or data in	plock(2)
times: get process and child	process times.	times(2)
wait: wait for child	process to stop or terminate.	wait(2)
ptrace:	process trace.	ptrace(2)
pause: suspend	process until signal.	pause(2)
wait: await completion of	process.	wait(1)
/list of file systems	processed by fsck and ncheck.	checklist(4)
to a process or a group of	processes. /send a signal	kill(2)
killall: kill all active	processes.	killall(1M)
structure. fuser: identify	processes using a file or file	fuser(1M)

awk: pattern scanning and processing language.	awk(1)
nawk: pattern scanning and processing language.	nawk(1)
extproc: turn external processing on or off.	extproc(1M)
mailx: interactive message processing system.	mailx(1)
rtab: Remote I/O Processor configuration table.	rtab(4)
en: Ethernet Processor.	en(7)
enpstart: configure Ethernet processor.	enpstart(1M)
riopqry: query Remote I/O Processor for online data.	riopqry(1M)
m4: macro processor.	m4(1)
system for Remote I/O Processor. riopcfg: configure	riopcfg(1M)
a common object file. list: produce C source listing from	list(1)
t_error: produce error message.	t_error(3n)
function. prof: profile within a	prof(5)
profile. profil: execution time	profil(2)
prof: display profile data.	prof(1)
monitor: prepare execution profile.	monitor(3C)
profil: execution time profile.	profil(2)
environment at login time. profile: setting up an	profile(4)
prof: profile within a function.	prof(5)
fusage: disk access profiler.	fusage(1M)
prf: operating system profiler.	prf(7)
prfdc, prfsnap, prfpr:/ profiler: prfld, prfstat,	profiler(1M)
sadp: disk access profiler.	sadp(1M)
standard/restricted command programming language. /the	sh(1)
software using the mkfs(1) proto file database. /verify	qinstall(1)
on. /print out file lists from proto file; set links based	qlist(1)
arp: Address Resolution Protocol.	arp(7)
/switched Serial Line Internet Protocol control facility.	slipd(1M)
/setprotoent, endprotoent: get protocol entry.	getprotoent(3)
inet: Internet protocol family.	inet(7)
icmp: Internet Control Message Protocol.	icmp(7)
ip: Internet Protocol.	ip(7)
DARPA Internet File Transfer Protocol server. ftpd:	ftpd(1M)
telnetd: DARPA TELNET protocol server.	telnetd(1M)
DARPA Trivial File Transfer Protocol server. tftpd:	tftpd(1M)
Internet Transmission Control Protocol. tcp:	tcp(7)
user interface to TELNET protocol. telnet:	telnet(1)
interface to the DARPA TFTP protocol. tftp: user	tftp(1)
udp: Internet User Datagram Protocol.	udp(7)
Dialers: ACU/modem calling protocols.	Dialers(5)
protocols: list of Internet protocols	protocols(4)
information. t_getinfo: get protocol-specific service	t_getinfo(3n)
update: provide disk synchronization.	update(1M)
arithmetic: provide drill in number facts.	arithmetic(6)
systems. labelit: provide labels for file	labelit(1M)
true, false: provide truth values.	true(1)
get name of transport provider. nlsprovider:	nlsprovider(3n)
prs: print an SCCS file.	prs(1)
/nulladm, prctmp, prdaily, prtacct, runacct, shutacct,/	acctsh(1M)
ps: report process status.	ps(1)
/generate uniformly distributed pseudo-random numbers.	drand48(3C)
/mvtpy: install or relocate a PT or GT local printer.	mktpy(1)
download. tdl, gtdl, ptdl: RS-232 terminal	tdl(1)
ptrace: process trace.	ptrace(2)
ptx: permuted index.	ptx(1)
stream. ungetc: push character back into input	ungetc(3S)
put character or word on a/ putc, putchar, fputc, putw:	putc(3S)
environment. putenv: change or add value to	putenv(3C)

stream.	putmsg: send a message on a	putmsg(2)
entry.	putpwent: write password file	putpwent(3C)
stream.	puts, fputs: put a string on a	puts(3S)
password file entry.	putspent: write shadow	putspent(3X)
/getutent, getutid, getutline,	pututline, setutent, endutent,/	getut(3C)
a/ putc, putchar, fputc,	putw: put character or word on	putc(3S)
file checkers.	pwck, grpck: password/group	pwck(1M)
/etc/shadow with information/	pwconv: install and update	pwconv(1M)
	pwd: working directory name.	pwd(1)
/etc/shadow with information/	pwunconv: install and update	pwunconv(1M)
qic: interface for	QIC tape.	qic(7)
software using the mkfs(1)/	qinstall: install and verify	qinstall(1)
from proto file; set links/	qlist: print out file lists	qlist(1)
	qsort: quicker sort.	qsort(3C)
tape. stape: SCSI	quarter-inch and half-inch	stape(7)
File Sharing name server	query. nsquery: Remote	nsquery(1M)
online data. riopqry:	query Remote I/O Processor for	riopqry(1M)
tput: initialize a terminal or	query terminfo database.	tput(1)
queuedefs: at/batch/cron	queue description file.	queuedefs(4)
msgget: get message	queue.	msgget(2)
rmount:	queue remote resource mounts.	rmount(1M)
ipcm: remove a message	queue, semaphore set or shared/	ipcm(1)
request. rmount: cancel	queued remote resource	rmount(1M)
description file.	queuedefs: at/batch/cron queue	queuedefs(4)
qsort:	quicker sort.	qsort(3C)
command immune to hangups and	quits. nohup: run a	nohup(1)
	quiz: test your knowledge.	quiz(6)
random-number generator.	rand, srand: simple	rand(3C)
adage. fortune: print a	random, hopefully interesting,	fortune(6)
rand, srand: simple	random-number generator.	rand(3C)
fsplit: split FORTRAN,	ratfor, or efl files.	fsplit(1)
dialect.	ratfor: rational FORTRAN	ratfor(1)
ratfor:	rational FORTRAN dialect.	ratfor(1)
stop the operating system.	rc0: run commands performed to	rc0(1M)
performed for multi-user/	rc2, rc3: run commands	rc2(1M)
execution.	rcmd: remote shell command	rcmd(1)
routines for returning a/	rcmd, rresvport, ruserok:	rcmd(3)
	rcp: remote file copy.	rcp(1)
getpass:	read a password.	getpass(3C)
entry of a common/ ldtbread:	read an indexed symbol table	ldtbread(3X)
header/ ldshread, ldnsbread:	read an indexed/named section	ldshread(3X)
in a file. getdents:	read directory entries and put	getdents(2)
read:	read from file.	read(2)
rmail: send mail to users or	read mail. mail,	mail(1)
line:	read one line.	line(1)
	read: read from file.	read(2)
member of an/ ldahread:	read the archive header of a	ldahread(3X)
common object file. ldfhread:	read the file header of a	ldfhread(3X)
directory: opendir,	readdir, telldir, seekdir,/	directory(3X)
open a common object file for	reading. ldopen, ldaopen:	ldopen(3X)
open: open for	reading or writing.	open(2)
lseek: move	read/write file pointer.	lseek(2)
tirdwr: Transport Interface	read/write interface STREAMS/	tirdwr(7)
allocator. malloc, free,	realloc, calloc: main memory	malloc(3C)
mallinfo: fast/ malloc, free,	realloc, calloc, mallopt,	malloc(3X)
enabled/disabled. rtpenable:	real-time priorities	rtpenable(1M)
reboot:	reboot the system.	reboot(1M)
mail aliases/ newaliases:	rebuild the data base for the	newaliases(1)

specify what to do upon	receipt of a signal.	signal:	signal(2)
t_rcvrel: acknowledge	receipt of an orderly release/		t_rcvrel(3n)
t_rcvdata:	receive a data unit.		t_rcvdata(3)
socket. rcv, rcvfrom:	receive a message from a		rcv(2)
indication. t_rcvuderr:	receive a unit data error		t_rcvuderr(3)
sent over a/ t_rcv:	receive data or expedited data		t_rcv(3n)
a connect/ t_rcvconnect:	receive the confirmation from		t_rcvconnect(3)
lockf:	record locking on files.		lockf(3C)
from per-process accounting	records. /command summary		acctcms(1M)
from/ errdead: extract error	records and status information		errdead(1M)
manipulate connect accounting	records. fwtmp, wtmpfix:		fwtmp(1M)
tape. freq:	recover files from a backup		freq(1M)
message from a socket.	rcv, rcvfrom: receive a		rcv(2)
ed,	red: text editor.		ed(1)
execute regular expression.	regcmp, regex: compile and		regcmp(3X)
compile.	regcmp: regular expression		regcmp(1)
make: maintain, update, and	regenerate groups of programs.		make(1)
regular expression. regcmp,	regex: compile and execute		regcmp(3X)
locking: exclusive access to	regions of a file.		locking(2)
match routines. regexp:	regular expression compile and		regexp(5)
regcmp:	regular expression compile.		regcmp(1)
regex: compile and execute	regular expression. regcmp,		regcmp(3X)
file for a pattern using full	regular expressions. /search a		egrep(1)
requests. accept,	reject: allow or prevent LP		accept(1M)
sorted files. comm: select or	reject lines common to two		comm(1)
lorder: find ordering	relation for an object/		lorder(1)
join:	relational database operator.		join(1)
/receipt of an orderly	release indication.		t_rcvrel(3n)
t_sndrel: initiate an orderly	release.		t_sndrel(3n)
for a common object file.	reloc: relocation information		reloc(4)
mktpy, mvtpy: install or	relocate a PT or GT local/		mktpy(1)
ldrseek, ldnrseek: seek to	relocation entries of a/		ldrseek(3X)
common object file. reloc:	relocation information for a		reloc(4)
/fmod, fabs: floor, ceiling,	remainder, absolute value/		floor(3M)
calendar:	reminder service.		calendar(1)
adv: advertise a directory for	remote access.		adv(1M)
for returning a stream to a	remote command. /routines		rcmd(3)
uuxqt: execute	remote command requests.		uuxqt(1M)
rexec: return stream to a	remote command.		rexec(3)
rhosts:	remote equivalent users.		rhosts(4)
rexecd:	remote execution server.		rexecd(1M)
rcp:	remote file copy.		rcp(1)
administration. rfadmin:	Remote File Sharing		rfadmin(1M)
process. rfudaemon:	Remote File Sharing daemon		rfudaemon(1M)
network names. dname: print	Remote File Sharing domain and		dname(1M)
environment. rfstop: stop the	Remote File Sharing		rfstop(1M)
password. rfpasswd: change	Remote File Sharing host		rfpasswd(1M)
server master file. rfmaster:	Remote File Sharing name		rfmaster(4)
server query. nsquery:	Remote File Sharing name		nsquery(1M)
notification shell/ rfadmin:	Remote File Sharing		rfadmin(1M)
unadv: unadvertise a	Remote File Sharing resource.		unadv(1M)
/rmountall: mount, unmount	Remote File Sharing (RFS)/		rmountall(1M)
rfstart: start	Remote File Sharing.		rfstart(1M)
group mapping. idload:	Remote File Sharing user and		idload(1M)
configuration table. rtab:	Remote I/O Processor		rtab(4)
online data. riopqry: query	Remote I/O Processor for		riopqry(1M)
riopcpg: configure system for	Remote I/O Processor.		riopcpg(1M)
rlogin:	remote login.		rlogin(1)

rlogind:	remote login server.	rlogind(1M)
showmount:	show all remote mounts.	showmount(1M)
netrc:	login file for remote networks.	netrc(4)
rmount:	queue remote resource mounts.	rmount(1M)
rumount:	cancel queued remote resource request.	rumount(1M)
and unmount file systems and	remote resources. /mount	mount(1M)
rmntry:	attempt to mount remote resources.	rmntry(1M)
execution. rcmd:	remote shell command	rcmd(1)
rshd:	remote shell server.	rshd(1M)
on. Uutry:	try to contact a remote system with debugging	Uutry(1M)
ct:	spawn getty to a remote terminal.	ct(1C)
server. talkd:	remote user communication	talkd(1M)
server. fingerd:	remote user information	fingerd(1M)
table. rmtab:	remotely mounted file system	rmtab(4)
file. rmdel:	remove a delta from an SCCS	rmdel(1)
rmdir:	remove a directory.	rmdir(2)
semaphore set or/ ipcrm:	remove a message queue,	ipcrm(1)
unlink:	remove directory entry.	unlink(2)
rm, rmdir:	remove files or directories.	rm(1)
eqn constructs. deroff:	remove nroff/troff, tbl, and	deroff(1)
running process by changing/	renice: alter priority of	renice(1)
fsck, dfscck:	check and repair file systems.	fsck(1M)
uniq:	report repeated lines in a file.	uniq(1)
clock:	report CPU time used.	clock(3C)
fsize:	report file size.	fsize(1)
fsstat:	report file system status.	fsstat(1M)
communication/ ipcs:	report inter-process	ipcs(1)
blocks and i-nodes. df:	report number of free disk	df(1M)
errpt:	process a report of logged errors.	errpt(1M)
sa2, sadc:	system activity report package. sar: sa1,	sar(1M)
timex:	time a command; report process data and system/	timex(1)
ps:	report process status.	ps(1)
file. uniq:	report repeated lines in a	uniq(1)
rpcinfo:	report RPC information.	rpcinfo(1M)
sar:	system activity reporter.	sar(1)
stream. fseek, rewind, ftell:	reposition a file pointer in a	fseek(3S)
and send listener service	request message. /format	nlsrequest(3n)
cancel queued remote resource	request. rumount:	rumount(1M)
mountd: NFS mount	request server.	mountd(1M)
t_accept:	accept a connect request.	t_accept(3n)
t_listen:	listen for a connect request.	t_listen(3n)
confirmation from a connect	request. /receive the	t_rcvconnect(3)
send user-initiated disconnect	request. t_snddis:	t_snddis(3n)
reject: allow or prevent LP	requests. accept,	accept(1M)
the LP scheduler and move	requests. /lpmove: start/stop	lpsched(1M)
syslocal: special system	requests.	syslocal(2)
lp, cancel: send/cancel	requests to an LP line/	lp(1)
uuxqt: execute remote command	requests.	uuxqt(1M)
res_mkquery, res_send,	res_init, dn_comp, dn_expand:/	resolver(3)
res_init, dn_comp, dn_expand:/	res_mkquery, res_send,	resolver(3)
control. arp:	address and resolution display and	arp(1M)
arp:	Address Resolution Protocol.	arp(7)
resolv.conf:	resolver configuration file.	resolver(4)
res_init, dn_comp, dn_expand:	resolver routines. /res_send,	resolver(3)
umount of an advertised	resource. fumount: forced	fumount(1M)
rmntstat: display mounted	resource information.	rmntstat(1M)
rmount: queue remote	resource mounts.	rmount(1M)
rumount: cancel queued remote	resource request.	rumount(1M)

a Remote File Sharing	resource. unadv: unadvertise	. . . . .	unadv(1M)
file systems and remote	resources. /mount and unmount	. . . . .	mount(1M)
unmount Network File System	resources. /numountall: mount,	. . . . .	numountall(1M)
attempt to mount remote	resources. rmnttry:	. . . . .	rmnttry(1M)
Remote File Sharing (RFS)	resources. /mount, unmount	. . . . .	rmountall(1M)
dn_expand:/ res_mkquery,	res_send, res_init, dn_comp,	. . . . .	resolver(3)
and usage examples. usage:	retrieve a command description	. . . . .	usage(1)
disconnect. t_rcvdis:	retrieve information from	. . . . .	t_rcvdis(3n)
common object file/ ldgetname:	retrieve symbol name for	. . . . .	ldgetname(3X)
abs:	return integer absolute value.	. . . . .	abs(3C)
logname:	return login name of user.	. . . . .	logname(3X)
command. rexec:	return stream to a remote	. . . . .	rexec(3)
name. getenv:	return value for environment	. . . . .	getenv(3C)
stat: data	returned by stat system call.	. . . . .	stat(5)
/ruserok: routines for	returning a stream to a remote/	. . . . .	rcmd(3)
col: filter	reverse line-feeds.	. . . . .	col(1)
file pointer in a/ fseek,	rewind, ftell: reposition a	. . . . .	fseek(3S)
/readdir, telldir, seekdir,	rewinddir, closedir: directory/	. . . . .	directory(3X)
creat: create a new file or	rewrite an existing one.	. . . . .	creat(2)
remote command.	rexec: return stream to a	. . . . .	rexec(3)
server.	rexc: remote execution	. . . . .	rexc(1M)
administration.	rfadmin: Remote File Sharing	. . . . .	rfadmin(1M)
name server master file.	rfmaster: Remote File Sharing	. . . . .	rfmaster(4)
Sharing host password.	rfpasswd: change Remote File	. . . . .	rfpasswd(1M)
unmount Remote File Sharing	(RFS) resources. /mount,	. . . . .	rmountall(1M)
Sharing.	rfstart: start Remote File	. . . . .	rfstart(1M)
Sharing environment.	rfstop: stop the Remote File	. . . . .	rfstop(1M)
notification shell script.	rfuadmin: Remote File Sharing	. . . . .	rfuadmin(1M)
daemon process.	rfudaemon: Remote File Sharing	. . . . .	rfudaemon(1M)
users.	rhosts: remote equivalent	. . . . .	rhosts(4)
Remote I/O Processor.	riopcfg: configure system for	. . . . .	riopcfg(1M)
Processor for online data.	riopqry: query Remote I/O	. . . . .	riopqry(1M)
	rlogin: remote login.	. . . . .	rlogin(1)
	rlogind: remote login server.	. . . . .	rlogind(1M)
directories.	rm, mdir: remove files or	. . . . .	rm(1)
read mail. mail,	rmail: send mail to users or	. . . . .	mail(1)
SCCS file.	rmdel: remove a delta from an	. . . . .	rmdel(1)
	rmkdir: remove a directory.	. . . . .	rmkdir(2)
directories. rm,	rmkdir: remove files or	. . . . .	rm(1)
resource information.	rmntstat: display mounted	. . . . .	rmntstat(1M)
remote resources.	rmnttry: attempt to mount	. . . . .	rmnttry(1M)
mounts.	rmount: queue remote resource	. . . . .	rmount(1M)
unmount Remote File Sharing/	rmountall, rumountall: mount,	. . . . .	rmountall(1M)
system table.	rmtab: remotely mounted file	. . . . .	rmtab(4)
chroot: change	root directory.	. . . . .	chroot(2)
chroot: change	root directory for a command.	. . . . .	chroot(1M)
logarithm, power, square	root functions. /exponential,	. . . . .	exp(3M)
routing tables.	route: manually manipulate the	. . . . .	route(1M)
gateways:	routed configuration file.	. . . . .	gateways(4)
daemon.	routed: network routing	. . . . .	routed(1M)
/tekset, td: graphical device	routines and filters.	. . . . .	gdev(1G)
rcmd, resvport, ruserok:	routines for returning a/	. . . . .	rcmd(3)
Internet address manipulation	routines. /inet_netof:	. . . . .	inet(3)
common object file access	routines. ldfcn:	. . . . .	ldfcn(4)
expression compile and match	routines. regexp: regular	. . . . .	regexp(5)
dn_comp, dn_expand: resolver	routines. /res_send, res_init,	. . . . .	resolver(3)
graphical table of contents	routines. /dtoc, itoc, vtoc:	. . . . .	toc(1G)
routed: network	routing daemon.	. . . . .	routed(1M)



sendmail: mail	routing program.	sendmail(1M)
route: manually manipulate the	routing tables.	route(1M)
getrpcbyname: get	rpc entry. /getrpcbyname,	getrpcent(3)
rpcinfo: report	RPC information.	rpcinfo(1M)
getrpcport: get	RPC port number.	getrpcport(3)
rpc: Sun	rpc program number data base.	rpc(4)
portmap: DARPA port to	RPC program number mapper.	portmap(1M)
data base.	rpc: Sun rpc program number	rpc(4)
information.	rpcinfo: report RPC	rpcinfo(1M)
for returning a stream/ rcmd,	rresvport, ruserok: routines	rcmd(3)
controlling terminal's local	RS-232 channels. tp:	tp(7)
tdl, gtdl, ptdl:	RS-232 terminal download.	tdl(1)
standard/restricted/ sh,	rsh: shell, the	sh(1)
	rshd: remote shell server.	rshd(1M)
stop terminal input and/	rsterm: manually start and	rsterm(1M)
configuration table.	rtab: Remote I/O Processor	rtab(4)
priorities enabled/disabled.	rtenable: real-time	rtenable(1M)
resource request.	rumount: cancel queued remote	rumount(1M)
Remote File/ rmountall,	rumountall: mount, unmount	rmountall(1M)
nice:	run a command at low priority.	nice(1)
hangups and quits. nohup:	run a command immune to	nohup(1)
multi-user/ rc2, rc3:	run commands performed for	rc2(1M)
the operating system. rc0:	run commands performed to stop	rc0(1M)
runacct:	run daily accounting.	runacct(1M)
/prctmp, prdaily, prtacct,	runacct, shutacct, startup/	acctsh(1M)
renice: alter priority of	running process by changing/	renice(1)
nodes on local network.	ruptime: display status of	ruptime(1)
returning a/ rcmd, rresvport,	ruserok: routines for	rcmd(3)
local network.	rwho: who is logged in on	rwho(1)
	rwhod: host status server.	rwhod(1M)
activity report package. sar:	sa1, sa2, sadc: system	sar(1M)
report package. sar: sa1,	sa2, sadc: system activity	sar(1M)
editing activity.	sact: print current SCCS file	sact(1)
package. sar: sa1, sa2,	sadc: system activity report	sar(1M)
	sadp: disk access profiler.	sadp(1M)
	sag: system activity graph.	sag(1G)
activity report package.	sar: sa1, sa2, sadc: system	sar(1M)
	sar: system activity reporter.	sar(1)
space allocation. brk:	sbrk: change data segment	brk(2)
formatted input.	scanf, fscanf, sscanf: convert	scanf(3S)
bfs: big file	scanner.	bfs(1)
language. awk: pattern	scanning and processing	awk(1)
language. nawk: pattern	scanning and processing	nawk(1)
the delta commentary of an	SCCS delta. cdc: change	cdc(1)
comb: combine	SCCS deltas.	comb(1)
make a delta (change) to an	SCCS file. delta:	delta(1)
sact: print current	SCCS file editing activity.	sact(1)
get: get a version of an	SCCS file.	get(1)
prs: print an	SCCS file.	prs(1)
rm del: remove a delta from an	SCCS file.	rm del(1)
compare two versions of an	SCCS file. sccsdiff:	sccsdiff(1)
sccsfile: format of	SCCS file.	sccsfile(4)
undo a previous get of an	SCCS file. unget:	unget(1)
val: validate	SCCS file.	val(1)
admin: create and administer	SCCS files.	admin(1)
what: identify	SCCS files.	what(1)
of an SCCS file.	sccsdiff: compare two versions	sccsdiff(1)
	sccsfile: format of SCCS file.	sccsfile(4)

check file system backup	schedule. ckbupscd: . . . . .	ckbupscd(1M)
/lpmove: start/stop the LP	scheduler and move requests.	lpsched(1M)
usched: the	scheduler for the UUCP system. . . . .	uusched(1M)
common object file.	scnhdr: section header for a . . . . .	scnhdr(4)
screen image file..	scr_dump: format of curses . . . . .	scr_dump(4)
clear: clear terminal	screen. . . . .	clear(1)
optimize: optimized	screen functions. . . . .	ocurse(3X)
optimization/ curses: terminal	screen handling and . . . . .	curses(3X)
scr_dump: format of curses	screen image file.. . . . .	scr_dump(4)
display editor based on/ vi:	screen-oriented (visual) . . . . .	vi(1)
inittab:	script for the init process. . . . .	inittab(4)
terminal session.	script: make typescript of . . . . .	script(1)
Sharing notification shell	script. rfuadmin: Remote File . . . . .	rfuadmin(1M)
scsi:	scsi control device. . . . .	scsi(7)
scsimap: set mappings for	SCSI devices. . . . .	scsimap(1M)
half-inch tape. stape:	SCSI quarter-inch and . . . . .	stape(7)
devices.	scsi: scsi control device. . . . .	scsi(7)
program.	scsimap: set mappings for SCSI . . . . .	scsimap(1M)
string. fgrep:	sdb: symbolic debugger. . . . .	sdb(1)
grep:	sdiff: side-by-side difference . . . . .	sdiff(1)
using full regular/ egrep:	search a file for a character . . . . .	fgrep(1)
bsearch: binary	search a file for a pattern. . . . .	grep(1)
accounting file(s). acctcom:	search a file for a pattern . . . . .	egrep(1)
lsearch, lfind: linear	search a sorted table. . . . .	bsearch(3C)
hcreate, hdestroy: manage hash	search and print process . . . . .	acctcom(1)
tdelete, twalk: manage binary	search and update. . . . .	lsearch(3C)
object file. scnhdr:	search tables. . . . .	hsearch(3C)
object/ /read an indexed/named	search trees. tsearch, tfind, . . . . .	tsearch(3C)
the object file comment	section header for a common . . . . .	scnhdr(4)
/to line number entries of a	section header of a common . . . . .	ldhread(3X)
/to relocation entries of a	section. mcs: manipulate . . . . .	mcs(1)
/seek to an indexed/named	section of a common object/ . . . . .	ldlseek(3X)
common object/ size: print	section of a common object/ . . . . .	ldrseek(3X)
/mrand48, jrand48, srand48,	section of a common object/ . . . . .	ldsseek(3X)
section of/ ldsseek, ldnseek:	section sizes in bytes of . . . . .	size(1)
a section/ ldlseek, ldnlseek:	sed: stream editor. . . . .	sed(1)
a section/ ldrseek, ldnrseek:	seed48, lcong48: generate/ . . . . .	drand48(3C)
header of a common/ ldohseek:	seek to an indexed/named . . . . .	ldsseek(3X)
common object file. ldthseek:	seek to line number entries of . . . . .	ldlseek(3X)
/ opendir, readdir, telldir,	seek to relocation entries of . . . . .	ldrseek(3X)
shmget: get shared memory	seek to the optional file . . . . .	ldohseek(3X)
brk, sbrk: change data	seek to the symbol table of a . . . . .	ldtseek(3X)
to two sorted files. comm:	seekdir, rewinddir, closedir:/ . . . . .	directory(3X)
multiplexing.	segment identifier. . . . .	shmget(2)
greek:	segment space allocation. . . . .	brk(2)
of a file. cut: cut out	select or reject lines common . . . . .	comm(1)
file. dump: dump	select: synchronous I/O . . . . .	select(2)
semctl:	select terminal filter. . . . .	greek(1)
semop:	selected fields of each line . . . . .	cut(1)
ipcrm: remove a message queue,	selected parts of an object . . . . .	dump(1)
semget: get set of	semaphore control operations. . . . .	semctl(2)
operations.	semaphore operations. . . . .	semop(2)
t_sndudata:	semaphore set or shared memory/ . . . . .	ipcrm(1)
send a data unit. . . . .	semaphores. . . . .	semget(2)
	semctl: semaphore control . . . . .	semctl(2)
	semget: get set of semaphores. . . . .	semget(2)
	semop: semaphore operations. . . . .	semop(2)
	send a data unit. . . . .	t_sndudata(3)

putmsg:	send a message on a stream.	putmsg(2)
send, sendto:	send a message to a socket.	send(2)
a group of processes. kill:	send a signal to a process or	kill(2)
over a connection. t_snd:	send data or expedited data	t_snd(3n)
to network hosts. ping:	send ICMP ECHO_REQUEST packets	ping(1M)
nlsrequest: format and	send listener service request/	nlsrequest(3n)
mail. mail, rmail:	send mail to users or read	mail(1)
to a socket.	send, sendto: send a message	send(2)
request. t_snddis:	send user-initiated disconnect	t_snddis(3n)
line printer. lp, cancel:	send/cancel requests to an LP	lp(1)
aliases: aliases file for	sendmail.	aliases(4)
program.	sendmail: mail routing	sendmail(1M)
socket. send,	sendto: send a message to a	send(2)
/receive data or expedited data	sent over a connection.	t_rcv(3n)
control/ slipd: switched	Serial Line Internet Protocol	slipd(1M)
/sldetach: attach and detach	serial lines as network/	slattach(1M)
serstat: display	serial port error statistics.	serstat(1M)
remote user information	server. fingerd:	fingerd(1M)
File Transfer Protocol	server. ftpd: DARPA Internet	ftpd(1M)
Remote File Sharing name	server master file. rfmaster:	rfmaster(4)
mountd: NFS mount request	server.	mountd(1M)
named: Internet domain name	server.	named(1M)
Remote File Sharing name	server query. nsquery:	nsquery(1M)
rexecd: remote execution	server.	rexecd(1M)
rlogind: remote login	server.	rlogind(1M)
rshd: remote shell	server.	rshd(1M)
rwhod: host status	server.	rwhod(1M)
remote user communication	server. talkd:	talkd(1M)
telnetd: DARPA TELNET protocol	server.	telnetd(1M)
Trivial File Transfer Protocol	server. tftpd: DARPA	tftpd(1M)
uucpd, ouucpd: network uucp	servers.	uucpd(1M)
make typescript of terminal	session. script:	script(1)
buffering to a stream.	setbuf, setvbuf: assign	setbuf(3S)
/toascii, _tolower, _toupper,	setchrclass: character/	ctype(3C)
IDs. setuid,	setgid: set user and group	setuid(2)
getgrent, getgrgid, getgrnam,	setgrent, endgrent, fgetgrent:/	getgrent(3C)
/gethostbyaddr, gethostent,	sethostent, endhostent: get/	gethostbyname(3)
identifier of/ gethostid,	sethostid: get/set unique	gethostid(2)
current host. gethostname,	sethostname: get/set name of	gethostname(2)
goto.	setjmp, longjmp: non-local	setjmp(3C)
hashing encryption. crypt,	setkey, encrypt: generate	crypt(3C)
	setmnt: establish mount table.	setmnt(1M)
/getnetbyaddr, getnetbyname,	setnetent, endnetent: get/	getnetent(3)
	setpgrp: set process group ID.	setpgrp(2)
protocol/ /getprotobyname,	setprotoent, endprotoent: get	getprotoent(3)
getpwent, getpwuid, getpwnam,	setpwent, endpwent, fgetpwent:/	getpwent(3C)
/getservbyport, getservbyname,	setservent, endservent: get/	getservent(3)
options on/ getsockopt,	setsockopt: get and set	getsockopt(2)
lckpwdf,/ getspent, getspsnam,	setspent, endspent, fgetspent,	getspent(3X)
time. gettimeofday,	settimeofday: get/set date and	gettimeofday(2)
environment at/ cprofile:	setting up a C shell	cprofile(4)
login time. profile:	setting up an environment at	profile(4)
gettydefs: speed and terminal	settings used by getty.	gettydefs(4)
group IDs.	setuid, setgid: set user and	setuid(2)
	setuname: set name of system.	setuname(1M)
/getutid, getutline, pututline,	setutent, endutent, utmpname:/	getut(3C)
stream. setbuf,	setvbuf: assign buffering to a	setbuf(3S)
data in a/ sputl,	sgctl: access long integer	sputl(3X)

standard/restricted command/	sh, rsh: shell, the	sh(1)
lckpddf, ulckpddf: get	shadow. /endspent, fgetspent,	getspent(3X)
putspent: write	shadow password file entry.	putspent(3X)
xstr: extract and	share strings in C programs.	xstr(1)
chkshlib: compare	shared libraries tool.	chkshlib(1)
mkshlib: create a	shared library.	mkshlib(1)
operations. shmctl:	shared memory control	shmctl(2)
queue, semaphore set or	shared memory ID. /a message	ipcrm(1)
shmop:	shared memory operations.	shmop(2)
identifier. shmget: get	shared memory segment	shmget(2)
nfssys: common	shared NFS system calls.	nfssys(2)
rfadmin: Remote File	Sharing administration.	rfadmin(1M)
rfudaemon: Remote File	Sharing daemon process.	rfudaemon(1M)
dname: print Remote File	Sharing domain and network/	dname(1M)
rfstop: stop the Remote File	Sharing environment.	rfstop(1M)
rfpasswd: change Remote File	Sharing host password.	rfpasswd(1M)
file. rfmaster: Remote File	Sharing name server master	rfmaster(4)
nsquery: Remote File	Sharing name server query.	nsquery(1M)
script. rfadmin: Remote File	Sharing notification shell	rfuadmin(1M)
unadvertise a Remote File	Sharing resource. unadv:	unadv(1M)
/mount, unmount Remote File	Sharing (RFS) resources.	rmountall(1M)
rfstart: start Remote File	Sharing.	rfstart(1M)
mapping. idload: Remote File	Sharing user and group	idload(1M)
rcmd: remote	shell command execution.	rcmd(1)
with C-like syntax. csh: a	shell (command interpreter)	csh(1)
system: issue a	shell command.	system(3S)
cprofile: setting up a C	shell environment at login/	cprofile(4)
shl:	shell layer manager.	shl(1)
shutacct, startup, tumacct:	shell procedures for/ /runacct,	acctsh(1M)
File Sharing notification	shell script. /Remote	rfuadmin(1M)
rshd: remote	shell server.	rshd(1M)
command programming/ sh, rsh:	shell, the standard/restricted	sh(1)
	shl: shell layer manager.	shl(1)
operations.	shmctl: shared memory control	shmctl(2)
segment identifier.	shmget: get shared memory	shmget(2)
operations.	shmop: shared memory	shmop(2)
mounts.	showmount: show all remote	showmount(1M)
/prdaily, prtacct, runacct,	shutacct, startup, tumacct:/	acctsh(1M)
system, change system state.	shutdown, halt: shut down	shutdown(1M)
full-duplex connection.	shutdown: shut down part of a	shutdown(2)
program. sdiff:	side-by-side difference	sdiff(1)
abort: generate a	SIGABRT.	abort(3C)
sigpause: signal/ sigset,	sighold, sigrelse, sigignore,	sigset(2)
sigset, sighold, sigrelse,	sigignore, sigpause: signal/	sigset(2)
login:	sign on.	login(1)
sigrelse, sigignore, sigpause:	signal management. /sighold,	sigset(2)
pause: suspend process until	signal.	pause(2)
what to do upon receipt of a	signal. signal: specify	signal(2)
of processes. kill: send a	signal to a process or a group	kill(2)
ssignal, gsignal: software	signals.	ssignal(3C)
/sighold, sigrelse, sigignore,	sigpause: signal management.	sigset(2)
signal/ sigset, sighold,	sigrelse, sigignore, sigpause:	sigset(2)
sigignore, sigpause: signal/	sigset, sighold, sigrelse,	sigset(2)
lex: generate programs for	simple lexical tasks.	lex(1)
generator. rand, srand:	simple random-number	rand(3C)
atan, atan2:/ trig:	sin, cos, tan, asin, acos,	trig(3M)
functions.	sinh, cosh, tanh: hyperbolic	sinh(3M)
fsize: report file	size.	fsize(1)

get descriptor table	size. getdtablesize:	getdtablesize(2)
object/ size: print section	sizes in bytes of common	size(1)
detach serial lines as/ an interval.	slattach, sldetach: attach and sleep: suspend execution for interval.	slattach(1M) sleep(1) sleep(3C)
documents, view graphs, and typesetting view graphs and linker, load socket/ Internet Protocol control/ current/ ttyslot: find the spline: interpolate	slides. mmt, mvt: typeset ./macro package for slink, ldsocket: STREAMS slipd: switched Serial Line slot in the utmp file of the smooth curve.	mmt(1) mv(5) slink(1) slipd(1M) ttyslot(3C) spline(1G)
sno:	SNOBOL interpreter.	sno(1)
bind: bind a name to a ldsocket: STREAMS linker, load	socket. socket configuration.	bind(2) slink(1)
initiate a connection on a communication.	socket. connect:	connect(2) socket(2)
listen for connections on a getsockname: get	socket. listen: socket name.	listen(2) getsockname(2)
receive a message from a sendto: send a message to a get and set options on ctinstall: install	socket. recv, recvfrom: socket. send, sockets. /setsockopt: software.	recv(2) send(2) getsockopt(2) ctinstall(1)
interface. lo: ssignal, gsignal:	software loopback network software signals.	lo(7) ssignal(3C)
qinstall: install and verify sort:	software using the mkfs(1)/ sort and/or merge files.	qinstall(1) sort(1)
qsort: quicker	sort.	qsort(3C)
tsort: topological	sort: sort and/or merge files. sort.	sort(1) tsort(1)
or reject lines common to two bsearch: binary search a object file. list: produce C	sorted files. comm: select sorted table.	comm(1) bsearch(3C)
brk, sbrk: change data segment /unexpand: expand tabs to terminal. ct:	source listing from a common space allocation. spaces, and vice versa.	list(1) brk(2) expand(1)
the/ tapedrives: tape drive cftime: language fspec: format	spawn getty to a remote specific information used by specific strings.	ct(1C) tapedrives(4) cftime(4) fspec(4)
receipt of a signal. signal: /set terminal type, modes, /set terminal type, modes, used by getty. gettydefs:	specification in text files. specify what to do upon speed, and line discipline. speed, and line discipline.	signal(2) getty(1M) uugetty(1M) gettydefs(4)
spelling/ spell, hashmake, spellin, hashcheck: find curve.	speed and terminal settings spellin, hashcheck: find spelling errors. /hashmake, spline: interpolate smooth	spell(1) spell(1) spline(1G)
split: csplit: context efl files. fsplit:	split a file into pieces. split.	split(1) csplit(1) fsplit(1)
uucleanup: uucp lpr: line printer lpadmin: configure the LP output. printf, fprintf, integer data in a/ power./ exp, log, log10, pow, exponential, logarithm, power, generator. rand, /nrand48, mrand48, jrand48, input. scanf, fscanf,	spool directory clean-up. spooler. spooling system. sprintf: print formatted sputl, sgetl: access long sqrt: exponential, logarithm, square root functions. /sqrt: srand: simple random-number srand48, seed48, lcong48:/ sscanf: convert formatted	uucleanup(1M) lpr(1) lpadmin(1M) printf(3S) sputl(3X) exp(3M) exp(3M) rand(3C) drand48(3C) scanf(3S)

	signals.	ssignal, gsignal: software	ssignal(3C)
	package.	stdio: standard buffered input/output	stdio(3S)
communication/	stdipc, ftok:	standard interprocess	stdipc(3C)
	sh, rsh: shell, the	standard/restricted command/	sh(1)
	half-inch tape.	stape: SCSI quarter-inch and	stape(7)
and output.	rsterm: manually	start and stop terminal input	rsterm(1M)
	rfstart:	start Remote File Sharing.	rfstart(1M)
	operating system for/	starter: information about the	starter(1)
and/	lpsched, lpshut, lpmove:	start/stop the LP scheduler	lpsched(1M)
	/pracct, runacct, shutacct,	startup, turnacct: shell/	acctsh(1M)
	useful with graphical/	stat, fstat: get file status.	stat(2)
	stat: data returned by	stat: statistical network	stat(1G)
	system information.	stat system call.	stat(5)
	with graphical/	statfs, fstatfs: get file	statfs(2)
	stat:	statistical network useful	stat(1G)
	ff: file name and	statistics for a file system.	ff(1M)
nfsstat:	Network File System	statistics.	nfsstat(1M)
	display serial port error	statistics. serstat:	serstat(1M)
	ustat: get file system	statistics.	ustat(2)
	fsstat: report file system	status.	fsstat(1M)
/extract	error records and	status information from dump.	errdead(1M)
	lpstat: print LP	status information.	lpstat(1)
feof, clearerr,	fileno: stream	status inquiries. ferror,	ferror(3S)
	control. uustat: uucp	status inquiry and job	uustat(1C)
	communication facilities	status. /report inter-process	ipes(1)
	netstat: show network	status.	netstat(1)
network.	ruptime: display	status of nodes on local	ruptime(1)
	ps: report process	status.	ps(1)
	rwhod: host	status server.	rwhod(1M)
	stat, fstat: get file	status.	stat(2)
	input/output package.	stdio: standard buffered	stdio(3S)
interprocess	communication/	stdipc, ftok: standard	stdipc(3C)
		stime: set time.	stime(2)
	wait for child process to	stop or terminate. wait:	wait(2)
	rsterm: manually start and	stop terminal input and/	rsterm(1M)
rc0:	run commands performed to	stop the operating system.	rc0(1M)
	environment. rfstop:	stop the Remote File Sharing	rfstop(1M)
	nextkey:/	store, delete, firstkey,	dbm(3X)
	dbm: init, fetch	strace: print STREAMS trace	strace(1M)
	messages.	strcat, strdup, strcat,	string(3C)
strcmp, strcmp,/	string:	strchr, strchr, strpbrk,/	string(3C)
	/strcpy, strcpy, strlen,	strclean: STREAMS error logger	strclean(1M)
	cleanup program.	strcmp, strcmp, strcpy,/	string(3C)
	/strcat, strdup, strcat,	strcpy, strcpy, strlen,/	string(3C)
	/strmcat, strcmp, strcmp,	strcspn, strtok: string/	string(3C)
	/strchr, strpbrk, strspn,	strdup, strmcat, strcmp,	string(3C)
	strcmp,/	string: strcat,	string(3C)
	string: strcat,	stream editor.	sed(1)
	sed:	stream. fclose,	fclose(3S)
	fflush: close or flush a	stream.	fopen(3S)
fopen, freopen,	fdopen: open a	stream. fseek, rewind, ftell:	fseek(3S)
	reposition a file pointer in a	stream. /getchar, fgetc, getw:	getc(3S)
	get character or word from a	stream.	getmsg(2)
getmsg:	get next message off a	stream. gets,	gets(3S)
	fgets: get a string from a	stream. /putchar, fputc, putw:	putc(3S)
	put character or word on a	stream.	putmsg(2)
putmsg:	send a message on a	stream.	puts(3S)
	puts, fputs: put a string on a	stream. setbuf,	setbuf(3S)
setvbuf:	assign buffering to a	stream status inquiries.	ferror(3S)
	/feof, clearerr, fileno:		

/routines for returning a	stream to a remote command. . . . .	rcmd(3)
rexec: return	stream to a remote command. . . . .	rexec(3)
push character back into input	stream. ungetc: . . . . .	ungetc(3S)
commands.	STREAMS iocbl . . . . .	streamio(7)
open any minor device on a	STREAMS driver. clone: . . . . .	clone(7)
program. strclean:	STREAMS error logger cleanup . . . . .	strclean(1M)
strerr:	STREAMS error logger daemon. . . . .	strerr(1M)
event/ log: interface to	STREAMS error logging and . . . . .	log(7)
multiplexing. poll:	STREAMS input/output . . . . .	poll(2)
streamio:	STREAMS ioctl commands. . . . .	streamio(7)
slink, lsocket:	STREAMS linker, load socket/ . . . . .	slink(1)
Interface cooperating	STREAMS module. /Transport . . . . .	timod(7)
Interface read/write interface	STREAMS module. /Transport . . . . .	tirdwr(7)
sxt:	STREAMS multiplexor. . . . .	sxt(7)
strace: print	STREAMS trace messages. . . . .	strace(1M)
daemon.	strerr: STREAMS error logger . . . . .	strerr(1M)
long integer and base-64 ASCII	string. /l64a: convert between . . . . .	a64l(3C)
convert date and time to	string. /asctime, tzset: . . . . .	ctime(3C)
floating-point number to	string. /fcvt, gcvt: convert . . . . .	ecvt(3C)
search a file for a character	string. fgrep: . . . . .	fgrep(1)
gps: graphical primitive	string, format of graphical/ . . . . .	gps(4)
gets, fgets: get a	string from a stream. . . . .	gets(3S)
puts, fputs: put a	string on a stream. . . . .	puts(3S)
bcmp, bzero: bit and byte	string operations. bcopy, . . . . .	bstring(3)
strspn, strcspn, strtok:	string operations. /strpbrk, . . . . .	string(3C)
number. strtod, atof: convert	string to double-precision . . . . .	strtod(3C)
strtol, atol, atoi: convert	string to integer. . . . .	strtol(3C)
ctime: language specific	strings. . . . .	ctime(4)
text strings in a file.	strings: extract the ASCII . . . . .	strings(1)
xstr: extract and share	strings in C programs. . . . .	xstr(1)
number information from a/	strip: strip symbol and line . . . . .	strip(1)
information from a/ strip:	strip symbol and line number . . . . .	strip(1)
/stmcmp, strcpy, stncpy,	strlen, strchr, strchr/ . . . . .	string(3C)
string: strcat, strdup,	stmcats, stncmp, stncmp/ . . . . .	string(3C)
/strdup, stmcats, stncmp,	stmcmp, strcpy, stncpy/ . . . . .	string(3C)
/stncmp, stncmp, strcpy,	stmcpy, strlen, strchr/ . . . . .	string(3C)
/strlen, strchr, strchr,	strpbrk, strspn, strcspn/ . . . . .	string(3C)
/stmcpy, strlen, strchr,	strchr, strpbrk, strspn/ . . . . .	string(3C)
/strchr, strrchr, strpbrk,	strspn, strcspn, strtok:/ . . . . .	string(3C)
to double-precision number.	strtod, atof: convert string . . . . .	strtod(3C)
/strpbrk, strspn, strcspn,	strtok: string operations. . . . .	string(3C)
string to integer.	strtol, atol, atoi: convert . . . . .	strtol(3C)
processes using a file or file	structure. fuser: identify . . . . .	fuser(1M)
t_alloc: allocate a library	structure. . . . .	t_alloc(3n)
t_free: free a library	structure. . . . .	t_free(3n)
terminal.	stty: set the options for a . . . . .	stty(1)
another user.	su: become super-user or . . . . .	su(1M)
firstkey, nextkey: database	subroutines. /store, delete, . . . . .	dbm(3X)
dbm_clearerr: database	subroutines. /dbm_error, . . . . .	ndbm(3X)
plot: graphics interface	subroutines. . . . .	plot(3X)
/same lines of several files or	subsequent lines of one file. . . . .	paste(1)
count of a file.	sum: print checksum and block . . . . .	sum(1)
du:	summarize disk usage. . . . .	du(1M)
accounting/ acctcms: command	summary from per-process . . . . .	acctcms(1M)
base. rpc:	Sun rpc program number data . . . . .	rpc(4)
sync: update the	super block. . . . .	sync(1M)
sync: update	super block. . . . .	sync(2)
inetd: internet	"super-server". . . . .	inetd(1M)

/file for inetd (internet	''super-server'').	inetd.conf(4)
su: become	super-user or another user.	su(1M)
interval. sleep:	suspend execution for an	sleep(1)
interval. sleep:	suspend execution for	sleep(3C)
pause:	suspend process until signal.	pause(2)
swab:	swap bytes.	swab(3C)
swab:	swap administrative interface.	swab(1M)
swab:	swap bytes.	swab(3C)
interface. swap:	swap administrative	swab(1M)
Protocol control/ slipd:	switched Serial Line Internet	slipd(1M)
file.	swrite: synchronous write on a	swrite(2)
	sxt: STREAMS multiplexor.	sxt(7)
information from/ strip:	strip symbol and line number	strip(1)
file/ ldgetname: retrieve	symbol name for common object	ldgetname(3X)
name for common object file	symbol table entry. /symbol	ldgetname(3X)
object/ /compute the index of a	symbol table entry of a common	ldtbindex(3X)
ldtbread: read an indexed	symbol table entry of a common/	ldtbread(3X)
syms: common object file	symbol table format.	syms(4)
object/ ldtbseek: seek to the	symbol table of a common	ldtbseek(3X)
unistd: file header for	symbolic constants.	unistd(4)
sdb:	symbolic debugger.	sdb(1)
common CTIX system terms and	symbols. /definitions of	glossary(1)
mkdbsym: load	symbols in kernel debugger.	mkdbsym(1M)
symbol table format.	syms: common object file	syms(4)
	sync: update super block.	sync(2)
	sync: update the super block.	sync(1M)
/correct the time to allow	synchronization of the system/	adjtime(2)
update: provide disk	synchronization.	update(1M)
t_sync:	synchronize transport library.	t_sync(3n)
select:	synchronous I/O multiplexing.	select(2)
swrite:	synchronous write on a file.	swrite(2)
interpreter) with C-like	syntax. csh: a shell (command	csh(1)
definition.	sysdef: output system	sysdef(1M)
error/ perror, ermo,	sys_errlist, sys_nerr: system	perror(3C)
information.	sysfs: get file system type	sysfs(2)
requests.	syslocal: special system	syslocal(2)
perror, ermo, sys_errlist,	sys_nerr: system error/	perror(3C)
shutdown, halt: shut down	system, change system state.	shutdown(1M)
binary search a sorted	table. bsearch:	bsearch(3C)
for common object file symbol	table entry. /symbol name	ldgetname(3X)
/compute the index of a symbol	table entry of a common object/	ldtbindex(3X)
file. /read an indexed symbol	table entry of a common object	ldtbread(3X)
common object file symbol	table format. syms:	syms(4)
master device information	table. master:	master(4)
mnttab: mounted file system	table.	mnttab(4)
ldtbseek: seek to the symbol	table of a common object file.	ldtbseek(3X)
/dtoc, ttoc, vtoc: graphical	table of contents routines.	toc(1G)
remotely mounted file system	table. rmtab:	rmtab(4)
I/O Processor configuration	table. rtab: Remote	rtab(4)
setmnt: establish mount	table.	setmnt(1M)
getdtablesize: get descriptor	table size.	getdtablesize(2)
classification and conversion	tables. /generate character	chrtbl(1M)
tbl: format	tables for nroff or troff.	tbl(1)
hdestroy: manage hash search	tables. hsearch, hcreate,	hsearch(3C)
manipulate the routing	tables. route: manually	route(1M)
tabs: set	tabs on a terminal.	tabs(1)
expand, unexpand: expand	tabs to spaces, and vice/	expand(1)
request.	t_accept: accept a connect	t_accept(3n)



ctags: create a	tags file. . . . .	ctags(1)
a file.	tail: deliver the last part of	tail(1)
talk:	talk to another user. . . . .	talk(1)
communication server.	talkd: remote user . . . . .	talkd(1M)
structure.	t_alloc: allocate a library . . . . .	t_alloc(3n)
trigonometric/ trig: sin, cos,	tan, asin, acos, atan, atan2: . . . . .	trig(3M)
sinh, cosh,	tanh: hyperbolic functions. . . . .	sinh(3M)
V/TAPE 3200 half-inch	tape controller. /Interphase . . . . .	ipt(7)
set drive parameters for	tape controllers. tapeset: . . . . .	tapeset(1M)
information used/ tapedrives:	tape drive specific . . . . .	tapedrives(4)
tsioctl: facilitate usage of a	tape drive. . . . .	tsioctl(1)
Hewlett-Packard 2645A terminal	tape file archiver. hpio: . . . . .	hpio(1)
tar:	tape file archiver. . . . .	tar(1)
recover files from a backup	tape. frec: . . . . .	frec(1M)
tio:	tape io filter. . . . .	tio(1)
qic: interface for QIC	tape. . . . .	qic(7)
quarter-inch and half-inch	tape. stape: SCSI . . . . .	stape(7)
specific information used by/	tapedrives: tape drive . . . . .	tapedrives(4)
for tape controllers.	tapeset: set drive parameters . . . . .	tapeset(1M)
programs for simple lexical	tar: tape file archiver. . . . .	tar(1)
transport endpoint.	tasks. lex: generate . . . . .	lex(1)
deroff: remove nroff/troff,	t_bind: bind an address to a . . . . .	t_bind(3n)
or troff.	tbl, and eqn constructs. . . . .	deroff(1)
endpoint.	tbl: format tables for nroff . . . . .	tbl(1)
connection with another/	t_close: close a transport . . . . .	t_close(3n)
Control Protocol.	t_connect: establish a . . . . .	t_connect(3n)
/hpd, erase, hardcopy, tekset,	tcp: Internet Transmission . . . . .	tcp(7)
search trees. tsearch, tfind,	td: graphical device routines/ . . . . .	gdev(1G)
terminal download.	tdelete, twalk: manage binary . . . . .	tsearch(3C)
gdev: hpd, erase, hardcopy,	tdl, gtdl, ptdl: RS-232 . . . . .	tdl(1)
4014: paginator for the	tee: pipe fitting. . . . .	tee(1)
initialization. init,	tekset, td: graphical device/ . . . . .	gdev(1G)
directory: opendir, readdir,	Tektronix 4014 terminal. . . . .	4014(1)
telnetd: DARPA	telinit: process control . . . . .	init(1M)
telnet: user interface to	telldir, seekdir, rewinddir./ . . . . .	directory(3X)
server.	TELNET protocol server. . . . .	telnetd(1M)
temporary file. tmpnam,	TELNET protocol. . . . .	telnet(1)
tmpfile: create a	telnetd: DARPA TELNET protocol . . . . .	telnetd(1M)
tempnam: create a name for a	tempnam: create a name for a . . . . .	tmpnam(3S)
terminals.	temporary file. . . . .	tmpfile(3S)
term: format of compiled	temporary file. tmpnam, . . . . .	tmpnam(3S)
terminfo/ captainfo: convert a	term: conventional names for . . . . .	term(5)
data base.	term file. . . . .	term(4)
for the Tektronix 4014	termcap description into a . . . . .	captainfo(1M)
functions of the DASI 450	termcap: terminal capability . . . . .	termcap(4)
interface. tiop:	terminal. 4014: paginator . . . . .	4014(1)
termcap:	terminal. 450: handle special . . . . .	450(1)
terminfo:	terminal accelerator . . . . .	tiop(7)
console: console	terminal capability data base. . . . .	termcap(4)
ct: spawn getty to a remote	terminal capability data base. . . . .	terminfo(4)
generate file name for	terminal. . . . .	console(7)
tdl, gtdl, ptdl: RS-232	terminal. . . . .	ct(1C)
/terminal interface, and	terminal. ctermid: . . . . .	ctermid(3S)
greek: select	terminal download. . . . .	tdl(1)
/getstr, tgoto, tputs:	terminal environment. . . . .	tset(1)
/manually start and stop	terminal filter. . . . .	greek(1)
	terminal independent/ . . . . .	otermcap(3X)
	terminal input and output. . . . .	rsterm(1M)

terminal/ tset: set terminal,	terminal interface, and	tset(1)
termio: general	terminal interface.	termio(7)
tty: controlling	terminal interface.	tty(7)
dial: establish an out-going	terminal line connection.	dial(3C)
list of terminal types by	terminal number. ttytype:	ttytype(4)
database. tput: initialize a	terminal or query terminfo	tput(1)
clear: clear	terminal screen.	clear(1)
optimization package. curses:	terminal screen handling and	curses(3X)
script: make typescript of	terminal session.	script(1)
getty. gettydefs: speed and	terminal settings used by	gettydefs(4)
stty: set the options for a	terminal.	stty(1)
tabs: set tabs on a	terminal.	tabs(1)
hpio: Hewlett-Packard 2645A	terminal tape file archiver.	hpio(1)
and terminal/ tset: set	terminal, terminal interface,	tset(1)
system/ conlocate: locate a	terminal to use as the virtual	conlocate(1M)
tty: get the name of the	terminal.	tty(1)
isatty: find name of a	terminal. ttyname,	ttyname(3C)
and line/ getty: set	terminal type, modes, speed,	getty(1M)
and line/ ugetty: set	terminal type, modes, speed,	ugetty(1M)
number. ttytype: list of	terminal types by terminal	ttytype(4)
vt: virtual	terminal.	vt(7)
functions of DASI 300 and 300s	terminals. /handle special	300(1)
functions of Hewlett-Packard	terminals. hp: handle special	hp(1)
channels. tp: controlling	terminal's local RS-232	tp(7)
term: conventional names for	terminals.	term(5)
kill:	terminate a process.	kill(1)
exit, _exit:	terminate process.	exit(2)
demon. errstop:	terminate the error-logging	errstop(1M)
for child process to stop or	terminate. wait: wait	wait(2)
tic:	terminfo compiler.	tic(1M)
initialize a terminal or query	terminfo database. tput:	tput(1)
a termcap description into a	terminfo description. /convert	captainfo(1M)
infocmp: compare or print out	terminfo descriptions.	infocmp(1M)
data base.	terminfo: terminal capability	terminfo(4)
interface.	termio: general terminal	termio(7)
/of common CTIX system	terms and symbols.	glossary(1)
message.	t_error: produce error	t_error(3n)
command.	test: condition evaluation	test(1)
isnan: isnand, isnanf:	test for floating point NaN/	isnan(3C)
quiz:	test your knowledge.	quiz(6)
ed, red:	text editor.	ed(1)
ex:	text editor.	ex(1)
casual users). edit:	text editor (variant of ex for	edit(1)
change the format of a	text file. newform:	newform(1)
fspec: format specification in	text files.	fspec(4)
/checkeq: format mathematical	text for nroff or troff.	eqn(1)
prepare constant-width	text for troff. cw, checkcw:	cw(1)
ms:	text formatting macros.	ms(5)
nroff: format	text.	nroff(1)
plock: lock process,	text, or data in memory.	plock(2)
more, page:	text perusal.	more(1)
strings: extract the ASCII	text strings in a file.	strings(1)
troff: typeset	text.	troff(1)
binary search trees. tsearch,	tfind, tdelete, twalk: manage	tsearch(3C)
structure.	t_free: free a library	t_free(3n)
user interface to the DARPA	TFTP protocol. tftp:	tftp(1)
Transfer Protocol server.	tftpd: DARPA Trivial File	tftpd(1M)
tgetstr, tgoto, tputs:/	tgetent, tgetnum, tgetflag,	otermcap(3X)

tputs:/ tgetent, tgetnum,	tget flag, tgetstr, tgoto,	otermcap(3X)
protocol-specific service/	t_getinfo: get	t_getinfo(3n)
tgoto, tputs:/ tgetent,	tgetnum, tgetflag, tgetstr,	otermcap(3X)
state.	t_getstate: get the current	t_getstate(3)
tgetent, tgetnum, tgetflag,	tgetstr, tgoto, tputs:/	otermcap(3X)
/tgetnum, tgetflag, tgetstr,	tgoto, tputs: terminal/	otermcap(3X)
	tic: terminfo compiler.	tic(1M)
ttt, cubic:	tic-tac-toe.	ttt(6)
data and system/ timex:	time a command; report process	timex(1)
time:	time a command.	time(1)
execute commands at a later	time. at, batch:	at(1)
a C shell environment at login	time. cprofile: setting up	cprofile(4)
systems for optimal access	time. dcopy: copy file	dcopy(1M)
	time: get time.	time(2)
settimeofday: get/set date and	time. gettimeofday,	gettimeofday(2)
profil: execution	time profile.	profil(2)
up an environment at login	time. profile: setting	profile(4)
stime: set	time.	stime(2)
time: get	time.	time(2)
of the/ adjtime: correct the	time to allow synchronization	adjtime(2)
tzset: convert date and	time to string. /ascftime,	ctime(3C)
clock: report CPU	time used.	clock(3C)
timezone: set default system	time zone.	timezone(4)
process times.	times: get process and child	times(2)
update access and modification	times of a file. touch:	touch(1)
get process and child process	times. times:	times(2)
file access and modification	times. utime: set	utime(2)
process data and system/	timex: time a command; report	timex(1)
time zone.	timezone: set default system	timezone(4)
cooperating STREAMS module.	timod: Transport Interface	timod(7)
	tio: tape io filter.	tio(1)
interface.	tiop: terminal accelerator	tiop(7)
read/write interface STREAMS/	tirdwr: Transport Interface	tirdwr(7)
request.	t_listen: listen for a connect	t_listen(3n)
event on a transport/	t_look: look at the current	t_look(3n)
file.	tmpfile: create a temporary	tmpfile(3S)
for a temporary file.	tmpnam, tmpnam: create a name	tmpnam(3S)
/isascii, tolower, toupper,	toascii, _tolower, _toupper,/	ctype(3C)
/tolower, _toupper, _tolower,	toascii: translate characters.	conv(3C)
graphical table of contents/	toc: dtoc, ttoc, vtoc:	toc(1G)
popen, pclose: initiate pipe	to/from a process.	popen(3S)
/toupper, tolower, _toupper,	_tolower, toascii: translate/	conv(3C)
tolower, toupper, toascii,	_tolower, _toupper,/ /isascii,	ctype(3C)
toascii:/ conv: toupper,	tolower, _toupper, _tolower,	conv(3C)
compare shared libraries	tool. chkshlib:	chkshlib(1)
endpoint.	t_open: establish a transport	t_open(3n)
tsort:	topological sort.	tsort(1)
a transport endpoint.	t_optmgmt: manage options for	t_optmgmt(3n)
acctmrg: merge or add	total accounting files.	acctmrg(1M)
modification times of a file.	touch: update access and	touch(1)
/toupper, toascii, _tolower,	_toupper, setchrclass:/	ctype(3C)
conv: toupper, tolower,	_toupper, _tolower, toascii:/	conv(3C)
local RS-232 channels.	tp: controlling terminal's	tp(7)
query terminfo database.	tplot: graphics filters.	tplot(1G)
/tgetflag, tgetstr, tgoto,	tput: initialize a terminal or	tput(1)
	tputs: terminal independent/	otermcap(3X)
	tr: translate characters.	tr(1)
strace: print STREAMS	trace messages.	strace(1M)

ptrace: process	trace. . . . .	ptrace(2)
error logging and event	tracing. /interface to STREAMS	log(7)
ftp: ARPANET file	transfer program. . . . .	ftp(1)
ftpd: DARPA Internet File	Transfer Protocol server. . . . .	ftpd(1M)
tftpd: DARPA Trivial File	Transfer Protocol server. . . . .	tftpd(1M)
/_toupper, _tolower, toascii:	translate characters. . . . .	conv(3C)
tr:	translate characters. . . . .	tr(1)
tcp: Internet	Transmission Control Protocol. . . . .	tcp(7)
t_bind: bind an address to a	transport endpoint. . . . .	t_bind(3n)
t_close: close a	transport endpoint. . . . .	t_close(3n)
look at the current event on a	transport endpoint. t_look: . . . . .	t_look(3n)
t_open: establish a	transport endpoint. . . . .	t_open(3n)
/manage options for a	transport endpoint. . . . .	t_optmgmt(3n)
t_unbind: disable a	transport endpoint. . . . .	t_unbind(3n)
cooperating STREAMS/ timod:	Transport Interface . . . . .	timod(7)
interface STREAMS/ tirdwr:	Transport Interface read/write . . . . .	tirdwr(7)
t_sync: synchronize	transport library. . . . .	t_sync(3n)
system. uuicico: file	transport program for the uuicp . . . . .	uuicico(1M)
nlsprovider: get name of	transport provider. . . . .	nlsprovider(3n)
a connection with another	transport user. /establish . . . . .	t_connect(3n)
expedited data sent over a/	t_rcv: receive data or . . . . .	t_rcv(3n)
confirmation from a connect/	t_rcvconnect: receive the . . . . .	t_rcvconnect(3)
from disconnect.	t_rcvdis: retrieve information . . . . .	t_rcvdis(3n)
of an orderly release/	t_rcvrel: acknowledge receipt . . . . .	t_rcvrel(3n)
unit.	t_rcvdata: receive a data . . . . .	t_rcvdata(3)
data error indication.	t_rcvuderr: receive a unit . . . . .	t_rcvuderr(3)
ftw: walk a file	tree. . . . .	ftw(3C)
twalk: manage binary search	trees. /find, tdelete, . . . . .	tsearch(3C)
trk:	trekkie game. . . . .	trk(6)
tan, asin, acos, atan, atan2:	trigonometric functions. /cos, . . . . .	trig(3M)
server. tftpd: DARPA	Trivial File Transfer Protocol . . . . .	tftpd(1M)
	trk: trekkie game. . . . .	trk(6)
constant-width text for	troff. cw, checkcw: prepare . . . . .	cw(1)
mathematical text for nroff or	troff. /neqn, checkeq: format . . . . .	eqn(1)
typesetting view graphs/ mv: a	troff macro package for . . . . .	mv(5)
format tables for nroff or	troff. tbl: . . . . .	tbl(1)
	troff: typeset text. . . . .	troff(1)
true, false: provide	truth values. . . . .	true(1)
with debugging on. Uutry:	try to contact a remote system . . . . .	Uutry(1M)
twalk: manage binary search/	tsearch, tfind, tdelete, . . . . .	tsearch(3C)
interface, and terminal/	tset: set terminal, terminal . . . . .	tset(1)
tape drive.	tsioctl: facilitate usage of a . . . . .	tsioctl(1)
data over a connection.	t_snd: send data or expedited . . . . .	t_snd(3n)
disconnect request.	t_snddis: send user-initiated . . . . .	t_snddis(3n)
release.	t_sndrel: initiate an orderly . . . . .	t_sndrel(3n)
	t_sndudata: send a data unit. . . . .	t_sndudata(3)
	tsort: topological sort. . . . .	tsort(1)
library.	t_sync: synchronize transport . . . . .	t_sync(3n)
contents routines. toc: dtoc,	ttoc, vtoc: graphical table of . . . . .	toc(1G)
	ttt, cubic: tic-tac-toe. . . . .	ttt(6)
interface.	tty: controlling terminal . . . . .	tty(7)
terminal.	tty: get the name of the . . . . .	tty(1)
a terminal.	ttyname, isatty: find name of . . . . .	ttyname(3C)
utmp file of the current/	ttyslot: find the slot in the . . . . .	ttyslot(3C)
types by terminal number.	ttytype: list of terminal . . . . .	ttytype(4)
endpoint.	t_unbind: disable a transport . . . . .	t_unbind(3n)
/runacct, shutacct, startup,	turnacct: shell procedures for/ . . . . .	acctsh(1M)
tsearch, tfind, tdelete,	twalk: manage binary search/ . . . . .	tsearch(3C)

file:	determine file type.	file(1)
sysfs:	get file system type information.	sysfs(2)
getty:	set terminal type, modes, speed, and line/	getty(1M)
uugetty:	set terminal type, modes, speed, and line/	uugetty(1M)
tytype:	list of terminal types by terminal number.	tytype(4)
nodes for assorted device types.	/create device types: primitive system data	createdev(1M)
session. script:	typescript of terminal	types(5)
graphs, and slides. mmt, mvt:	typeset documents, view	script(1)
troff:	typeset text.	mmt(1)
mv: a troff macro package for	typesetting view graphs and/	troff(1)
to/ /asctime, cftime, ascftime,	tzset: convert date and time	mv(5)
control.	uadmin: administrative	ctime(3C)
control.	uadmin: administrative	uadmin(1M)
system.	uconf: configure the operating	uadmin(2)
Protocol.	udp: Internet User Datagram	uconf(1M)
getpw: get name from	UID.	udp(7)
	ul: do underlining.	getpw(3C)
/endspent, fgetspent, lckpwdf,	ulckpwdf: get shadow.	ul(1)
limits.	ulimit: get and set user	endspent(3X)
creation mask.	umask: set and get file	ulimit(2)
mask.	umask: set file-creation mode	umask(2)
systems and remote/ mount,	umount: mount and unmount file	umask(1)
	umount: unmount a file system.	umount(1M)
multiple file/ mountall,	umountall: mount, unmount	umount(2)
Sharing resource. unadv:	unadvertise a Remote File	mountall(1M)
CTIX system.	uname: get name of current	unadv(1M)
CTIX system.	uname: print name of current	uname(2)
ul: do	underlining.	uname(1)
file. unget:	undo a previous get of an SCCS	ul(1)
spaces, and vice/ expand,	unexpand: expand tabs to	unget(1)
an SCCS file.	unget: undo a previous get of	expand(1)
into input stream.	ungetc: push character back	unget(1)
/seed48, lcong48: generate	uniformly distributed/	ungetc(3S)
a file.	uniq: report repeated lines in	drand48(3C)
mktemp: make a	unique file name.	uniq(1)
gethostid, sethostid: get/set	unique identifier of current/	mktemp(3C)
symbolic constants.	unistd: file header for	gethostid(2)
t_rcvuderr: receive a	unit data error indication.	unistd(4)
t_rcvudata: receive a data	unit.	t_rcvuderr(3)
t_sndudata: send a data	unit.	t_rcvudata(3)
	units: conversion program.	t_sndudata(3)
mc68k, miti, mini, mega,	unixpc, machid:	units(1)
execution. uux:	UNIX-to-UNIX system command	machid(1)
uucp, uulog, uuname:	UNIX-to-UNIX system copy.	uux(1C)
uuto, uupick: public	UNIX-to-UNIX system file copy.	uucp(1C)
link, unlink: link and	unlink files and directories.	uuto(1C)
entry.	unlink: remove directory	link(1M)
umount:	unmount a file system.	unlink(2)
mount, umount: mount and	unmount file systems and/	umount(2)
mountall, umountall: mount,	unmount multiple file systems.	mount(1M)
nmountall, numountall: mount,	unmount Network File System/	mountall(1M)
resource. fumount: forced	unmount of an advertised	nmountall(1M)
rmountall, rumountall: mount,	unmount Remote File Sharing/	fumount(1M)
manage notifications. notify,	unnotify, evwait, evnowait:	rmountall(1M)
files. pack, pcat,	unpack: compress and expand	notify(2)
times of a file. touch:	update access and modification	pack(1)
of programs. make: maintain,	update, and regenerate groups	touch(1)
		make(1)

pwconv: install and	update /etc/shadow with/	pwconv(1M)
pwunconv: install and	update /etc/shadow with/	pwunconv(1M)
lfind: linear search and	update. lsearch,	lsearch(3C)
synchronization.	update: provide disk	update(1M)
sync:	update super block.	sync(2)
sync:	update the super block.	sync(1M)
du: summarize disk	usage.	du(1M)
a command description and	usage examples. /retrieve	usage(1)
tsioctl: facilitate	usage of a tape drive.	tsioctl(1)
description and usage/	usage: retrieve a command	usage(1)
stat: statistical network	useful with graphical/	stat(1G)
id: print	user and group IDs and names.	id(1M)
setuid, setgid: set	user and group IDs.	setuid(2)
idload: Remote File Sharing	user and group mapping.	idload(1M)
talkd: remote	user communication server.	talkd(1M)
crontab:	user crontab file.	crontab(1)
character login name of the	user. cuserid: get	cuserid(3S)
udp: Internet	User Datagram Protocol.	udp(7)
/getgid, getegid: get real	user, effective user, real/	getuid(2)
environ:	user environment.	environ(5)
disk accounting data by	user ID. diskusg: generate	diskusg(1M)
program. finger:	user information lookup	finger(1)
fingerd: remote	user information server.	fingerd(1M)
protocol. telnet:	user interface to TELNET	telnet(1)
TFTP protocol. tftp:	user interface to the DARPA	tftp(1)
ulimit: get and set	user limits.	ulimit(2)
logname: return login name of	user.	logname(3X)
/get real user, effective	user, real group, and/	getuid(2)
become super-user or another	user. su:	su(1M)
talk: talk to another	user.	talk(1)
with another transport	user. /establish a connection	t_connect(3n)
the utmp file of the current	user. /find the slot in	ttyslot(3C)
write: write to another	user.	write(1)
request. t_snddis: send	user-initiated disconnect	t_snddis(3n)
(variant of ex for casual	users). edit: text editor	edit(1)
mail, rmail: send mail to	users or read mail.	mail(1)
rhosts: remote equivalent	users.	rhosts(4)
operating system for beginning	users. /information about the	starter(1)
wall: write to all	users.	wall(1)
fuser: identify processes	using a file or file/	fuser(1M)
search a file for a pattern	using full regular/ egrep:	egrep(1)
identify a CTIX system command	using keywords. locate:	locate(1)
assist: assistance	using CTIX system commands.	assist(1)
/install and verify software	using the mkfs(1) proto file/	qinstall(1)
failed login attempts.	/usr/adm/loginlog: log of	loginlog(4)
statistics.	ustat: get file system	ustat(2)
gutil: graphical	utilities.	gutil(1G)
modification times.	utime: set file access and	utime(2)
utmp, wtmp:	utmp and wtmp entry formats.	utmp(4)
endutent, utmpname: access	utmp file entry. /setutent,	getut(3C)
ttyslot: find the slot in the	utmp file of the current user.	ttyslot(3C)
/pututline, setutent, endutent,	utmpname: access utmp file/	getut(3C)
directories and permissions/	uucheck: check the uucp	uucheck(1M)
for the uucp system.	uucico: file transport program	uucico(1M)
directory clean-up.	uucleanup: uucp spool	uucleanup(1M)
/configuration file for	uucp communications lines.	Devices(5)
uucheck: check the	uucp directories and/	uucheck(1M)
uucpd, ouucpd: network	uucp servers.	uucpd(1M)

uucleanup:	uucp pool directory clean-up.	uucleanup(1M)
control. uustat:	uucp status inquiry and job	uustat(1C)
file transport program for the	uucp system. uuico:	uuico(1M)
uusched: the scheduler for the	UUCP system.	uusched(1M)
UNIX-to-UNIX system copy.	uucp, uulog, uuname:	uucp(1C)
servers.	uucpd, ouucpd: network uucp	uucpd(1M)
modes, speed, and line/	uugetty: set terminal type,	uugetty(1M)
system copy. uucp,	uulog, uuname: UNIX-to-UNIX	uucp(1C)
copy. uucp, uulog,	uuname: UNIX-to-UNIX system	uucp(1C)
system file copy. uuto,	uupick: public UNIX-to-UNIX	uuto(1C)
UUCP system.	uusched: the scheduler for the	uusched(1M)
and job control.	uustat: uucp status inquiry	uustat(1C)
UNIX-to-UNIX system file/	uuto, uupick: public	uuto(1C)
system with debugging on.	Uutry: try to contact a remote	Uutry(1M)
command execution.	uux: UNIX-to-UNIX system	uux(1C)
requests.	uuxqt: execute remote command	uuxqt(1M)
val:	validate SCCS file.	val(1)
abs: return integer absolute	value.	abs(3C)
getenv: return	value for environment name.	getenv(3C)
ceiling, remainder, absolute	value functions. /fabs: floor,	floor(3M)
putenv: change or add	value to environment.	putenv(3C)
/htons, ntohl, ntohs: convert	values between host and/	byteorder(3)
values.	values: machine-dependent	values(5)
true, false: provide truth	values.	true(1)
values: machine-dependent	values.	values(5)
/print formatted output of a	varargs argument list.	vprintf(3S)
argument list.	varargs: handle variable	varargs(5)
varargs: handle	variable argument list.	varargs(5)
users). edit: text editor	(variant of ex for casual	edit(1)
	vc: version control.	vc(1)
option letter from argument	vector. getopt: get	getopt(3C)
assert:	verify program assertion.	assert(3X)
mkfs(1)/ qinstall: install and	verify software using the	qinstall(1)
tabs to spaces, and vice	versa. /unexpand: expand	expand(1)
vc:	version control.	vc(1)
get: get a	version of an SCCS file.	get(1)
sccsdiff: compare two	versions of an SCCS file.	sccsdiff(1)
formatted output of/ vprintf,	vfprintf, vsprintf: print	vprintf(3S)
manipulate Volume Home Blocks	(VHB). libdev:	libdev(3X)
display editor based on ex.	vi: screen-oriented (visual)	vi(1)
expand tabs to spaces, and	vice versa. expand, unexpand:	expand(1)
mmt, mvt: typeset documents,	view graphs, and slides.	mmt(1)
macro package for typesetting	view graphs and slides. /troff	mv(5)
/a terminal to use as the	virtual system console.	conlocate(1M)
vt:	virtual terminal.	vt(7)
on ex. vi: screen-oriented	(visual) display editor based	vi(1)
vme:	VME bus interface.	vme(7)
file system.	volcopy: make literal copy of	volcopy(1M)
file system: format of system	volume. fs:	fs(4)
libdev: manipulate	Volume Home Blocks (VHB).	libdev(3X)
iv: initialize and maintain	volume.	iv(1)
print formatted output of a/	vprintf, vfprintf, vsprintf:	vprintf(3S)
	vt: virtual terminal.	vt(7)
ipt: interface for Interphase	V/TAPE 3200 half-inch tape/	ipt(7)
contents/ toc: dtoc, ttoc,	vtoc: graphical table of	toc(1G)
process.	wait: await completion of	wait(1)
or terminate. wait:	wait for child process to stop	wait(2)
ftw:	walk a file tree.	ftw(3C)

	wall: write to all users. . . . .	wall(1)
	wc: word count. . . . .	wc(1)
	what: identify SCCS files. . . . .	what(1)
signal.	signal: specify what to do upon receipt of a . . . . .	signal(2)
	whodo: who is doing what. . . . .	whodo(1M)
network.	rwho: who is logged in on local . . . . .	rwho(1)
	who: who is on the system. . . . .	who(1)
	whodo: who is doing what. . . . .	whodo(1M)
fold long lines for finite	width output device. fold: . . . . .	fold(1)
	wm: window management primitives. . . . .	window(7)
	wm: window management. . . . .	wm(1)
	cd: change working directory. . . . .	cd(1)
	chdir: change working directory. . . . .	chdir(2)
get path-name of current	working directory. getcwd: . . . . .	getcwd(3C)
	pwd: working directory name. . . . .	pwd(1)
swrite: synchronous	write on a file. . . . .	swrite(2)
	write: write on a file. . . . .	write(2)
	putpwent: write password file entry. . . . .	putpwent(3C)
entry.	putspent: write shadow password file . . . . .	putspent(3X)
	wall: write to all users. . . . .	wall(1)
	write: write to another user. . . . .	write(1)
	write: write on a file. . . . .	write(2)
open: open for reading or	writing. . . . .	open(2)
utmp, wtmp: utmp and	wtmp entry formats. . . . .	utmp(4)
accounting records.	fwtmp, wtmpfix: manipulate connect . . . . .	fwtmp(1M)
	hunt-the-wumpus. wump: the game of . . . . .	wump(6)
list(s) and execute command.	xargs: construct argument . . . . .	xargs(1)
	strings in C programs. xstr: extract and share . . . . .	xstr(1)
	bessel: j0, j1, jn, y0, y1, yn: Bessel functions. . . . .	bessel(3M)
	compiler-compiler. yacc: yet another . . . . .	yacc(1)
set default system time	zone. timezone: . . . . .	timezone(4)



**NAME**

adb - absolute debugger

**SYNOPSIS**

**adb** [ **-w** ] [ *objfil* [ *corfil* ] ]

**DESCRIPTION**

The *adb* program is a general-purpose debugging program. It can be used to examine files and to provide a controlled environment for the execution of CTIX programs.

The *objfil* parameter is normally an executable program file, preferably containing a symbol table; if not, the symbolic features of *adb* cannot be used, but the file can still be examined. The default for *objfil* is **a.out**. The *corfil* parameter is assumed to be a core image file produced after executing *objfil*; the default for *corfil* is **core**.

Requests to *adb* are read from the standard input and responses are to the standard output. If the **-w** flag is present, both *objfil* and *corfil* are created, if necessary, and opened for reading and writing so that files can be modified using *adb*. Note that *adb* ignores QUIT; INTERRUPT causes return to the next *adb* command.

In general, requests to *adb* are of the following form:

[ *address* ] [ , *count* ] [ *command* ] [ ; ]

If *address* is present, *dot* is set to *address*. Initially, *dot* is set to 0. For most commands, *count* specifies how many times the command is executed. The default *count* is 1. *Address* and *count* are expressions.

The interpretation of an address depends on the context it is used. If a subprocess is being debugged, then addresses are interpreted in the usual way in the address space of the subprocess. For further details of address mapping, see *ADDRESSES*.

**EXPRESSIONS**

- .
  - +
  - ^
  - "
- integer* Hexadecimal by default or if preceded by 0x; octal if preceded by 0o or 0O; decimal if preceded by 0t or 0T.
- integer.fraction* A 32-bit floating-point number.

*'cccc'* The ASCII value of up to four characters. A *\* may be used to escape a *'*.

*< name* The value of *name*, which is either a 68010/68020/68040 register name or a variable name. *adb* maintains a number of variables (see *VARIABLES*) named by single letters or digits. If *name* is a register name, then the value of the register is obtained from the system header in *corfil*. The registers are **d0** through **d7**, **a0** through **a7**, **sp**, **pc**, **cc**, **sr**, and **usp**.

*symbol* A *symbol* is a sequence of upper or lowercase letters, underscores, or digits, not starting with a digit. The value of the *symbol* is taken from the symbol table in *objfil*.

From C, only external variables are available as symbols. The symbol name is the same as the C variable name, except that an underscore (*\_*) is prepended to any name that is the same as the name for a register.

*(exp)* The value of the expression *exp*.

Monadic operators:

*\*exp* The contents of the location addressed by *exp* in *corfil*.

*@exp* The contents of the location addressed by *exp* in *objfil*.

*-exp* Integer negation.

*~exp* Bitwise complement.

Dyadic operators are left associative and are less binding than monadic operators.

*e1 + e2* Integer addition.

*e1 - e2* Integer subtraction.

*e1 \* e2* Integer multiplication.

*e1 % e2* Integer division.

*e1 & e2* Bitwise conjunction.

*e1 | e2* Bitwise disjunction.

*e1 # e2* *E1* rounded up to the next multiple of *e2*.

## COMMANDS

Most commands consist of a verb followed by a modifier or list of modifiers. The following verbs are available. (The commands *?* and */* may be followed by *\**; see *ADDRESSES* for further details.)

- ?f Locations starting at *address* in *objfil* are printed according to the format *f*. The value of *dot* is incremented by the sum of the increments for each format letter (q.v.).
- /f Locations starting at *address* in *corfil* are printed according to the format *f* and *dot* is incremented as for ?.
- =f The value of *address* is printed in the styles indicated by the format *f*. (For *i* format ? is printed for the parts of the instruction that reference subsequent words.)

A *format* consists of one or more characters that specify a style of printing. Each format character may be preceded by a decimal integer that is a repeat count for the format character. While stepping through a format, *dot* is incremented by the amount given for each format letter. If no format is given, then the last format is used. The format letters available are as follows:

- o 2 Prints 2 bytes in octal. All octal numbers output by *adb* are preceded by 0.
- O 4 Prints 4 bytes in octal.
- q 2 Prints in signed octal.
- Q 4 Prints long signed octal.
- d 2 Prints in decimal.
- D 4 Prints long decimal.
- x 2 Prints 2 bytes in hexadecimal.
- X 4 Prints 4 bytes in hexadecimal.
- u 2 Prints as an unsigned decimal number.
- U 4 Prints long unsigned decimal.
- f 4 Prints the 32-bit value as a floating-point number.
- F 8 Prints double floating point.
- b 1 Prints the addressed byte in octal.
- c 1 Prints the addressed character.
- C 1 Prints the addressed character using the following escape convention. Character values 000 to 040 are printed as @, followed by the corresponding character in the range 0100 to 0140. The character @ is printed as @@.
- s *n* Prints the addressed characters until a zero character is reached.

- S** *n* Prints a string using the **@** escape convention. The value *n* is the length of the string including its zero terminator.
- Y** 4 Prints 4 bytes in date format [see *ctime*(3C)].
- i** *n* Prints as machine instructions. The value *n* is the number of bytes occupied by the instruction. This style of printing causes variables 1 and 2 to be set to the offset parts of the source and destination, respectively.
- a** 0 Prints the value of *dot* in symbolic form. Symbols are checked to ensure that they have an appropriate type as indicated below:
- / local or global data symbol
  - ? local or global text symbol
  - = local or global absolute symbol
- p** 2 Prints the addressed value in symbolic form using the same rules for symbol lookup as **a**.
- t** 0 When preceded by an integer, tabs to the next appropriate tab stop. For example, **8t** moves to the next eight-space tab stop.
- r** 0 Prints a space.
- n** 0 Prints a new-line.
- "..."** 0 Prints the enclosed string.
- ^** The value of *dot* is decremented by the current increment. Nothing is printed.
- +** The value of *dot* is incremented by 1. Nothing is printed.
- The value of *dot* is decremented by 1. Nothing is printed.

new-line

Repeats the previous command with a *count* of 1.

[?/] *value mask*

Words starting at *dot* are masked with *mask* and compared with *value* until a match is found. If **L** is used, then the match is for 4 bytes at a time instead of 2. If no match is found, then *dot* is unchanged; otherwise *dot* is set to the matched location. If *mask* is omitted, then -1 is used.

[?/]w *value* ...

Writes the 2-byte *value* into the addressed location. If the command is **W**, write 4 bytes. Odd addresses are not allowed when writing to the subprocess address space.

[?/]m *b1 e1 f1*{?/}

New values for (*b1, e1, f1*) are recorded. If less than three expressions are given, then the remaining map parameters are left unchanged. If the ? or / is followed by \*, the second segment (*b2, e2, f2*) of the mapping is changed. If the list is terminated by ? or /, the file (*objfil* or *corfil*, respectively) is used for subsequent requests. (So that, for example, /m? causes / to refer to *objfil*.)

>*name* The value of *dot* is assigned to the variable or register named.

! A shell is called to read the rest of the line following !.

*Smodifier*

Miscellaneous commands. The available *modifiers* follow:

- <*f* Reads commands from the file *f* and return.
- >*f* Sends output to the file *f*, which is created if it does not exist.
- r Prints the general registers and the instruction addressed by **pc**. The value of *dot* is set to **pc**.
- b Prints all breakpoints and their associated counts and commands.
- c C stack backtrace. If *address* is given, then it is taken as the address of the current frame (instead of **fp**). If *count* is given, then only the first *count* frames are printed.
- e The names and values of external variables are printed.
- w Sets the page width for output to *address* (default 80).
- s Sets the limit for symbol matches to *address* (default 255).
- o All integers input are regarded as octal.
- d Resets integer input as described in *EXPRESSIONS*.
- q Exits from *adb*.

- v** Prints all nonzero variables.
- f** Prints the 68881 floating-point registers.
- m** Prints the address map.

*:modifier*

Manage a subprocess. Available modifiers are as follows:

- bc** Sets breakpoint at *address*. The breakpoint is executed *count*-1 times before causing a stop. Each time the breakpoint is encountered, the command *c* is executed. If this command sets *dot* to zero, the breakpoint causes a stop.
- d** Deletes breakpoint at *address*.
- r** Runs *obifil* as a subprocess. If *address* is given explicitly, then the program is entered at this point; otherwise, the program is entered at its standard entry point. The value *count* specifies how many breakpoints are to be ignored before stopping. Arguments to the subprocess can be supplied on the same line as the command. An argument starting with < or > causes the standard input or output to be established for the command. All signals are turned on entry to the subprocess.
- cs** The subprocess is continued with signal *s* [see *signal(2)*]. If *address* is given, the subprocess is continued at this address. If no signal is specified, then the signal that caused the subprocess to stop is sent. Breakpoint skipping is the same as for **r**.
- ss** As for **c** except that the subprocess is single-stepped *count* times. If there is no current subprocess, then *obifil* is run as a subprocess as for **r**. In this case, no signal can be sent; the remainder of the line is treated as arguments to the subprocess.
- k** The current subprocess, if any, is terminated.

## VARIABLES

The *adb* command provides a number of variables. Named variables are set initially by *adb*, but they are not used subsequently. Numbered variables are reserved for communication as follows.

- 0** The last value printed.
- 1** The last offset part of an instruction source.
- 2** The previous value of variable 1.

On entry, the following are set from the system header in the *corfil*. If *corfil* does not appear to be a **core** file, these values are set from *objfil*.

- b**      The base address of the data segment.
- d**      The data segment size.
- e**      The entry point.
- m**      The “magic” number (0407, 0410, or 0413).
- s**      The stack segment size.
- t**      The text segment size.

## ADDRESSES

The address in a file associated with a written address is determined by a mapping associated with that file. Each mapping is represented by two triples (*b1*, *e1*, *f1*) and (*b2*, *e2*, *f2*) and the *file address* corresponding to a written *address* is calculated as follows:

$$b1 \leq \text{address} < e1 \Rightarrow \text{file address} = \text{address} + f1 - b1$$

otherwise,

$$b2 \leq \text{address} < e2 \Rightarrow \text{file address} = \text{address} + f2 - b2,$$

otherwise, the requested *address* is not legal. In some cases (for example, for programs with separated I and D space), the two segments for a file can overlap. If a ? or / is followed by an asterisk (\*), only the second triple is used.

The initial setting of both mappings is suitable for normal **a.out** and **core** files. If either file is not of the kind expected, then for that file, *b1* is set to 0, *e1* is set to the maximum file size, and *f1* is set to 0; in this way, the whole file can be examined with no address translation.

So you can use *adb* on large files, all appropriate values are kept as signed 32-bit integers.

## FILES

/dev/kmem  
 /dev/swap  
 a.out  
 core

## SEE ALSO

ptrace(2), a.out(4), core(4).

**DIAGNOSTICS**

*adb* when there is no current command or format. Comments about inaccessible files, syntax errors, abnormal termination of commands, and so on. Exit status is 0, unless last command failed or returned nonzero status.

**BUGS**

A breakpoint set at the entry point is not effective on initial entry to the program.

When single-stepping, system calls do not count as an executed instruction.

Local variables whose names are the same as an external variable may foul up the accessing of the external.

Shared libraries are not included in the map.



**NAME**

as - common assembler

**SYNOPSIS**

as [ options ] filename

**DESCRIPTION**

The *as* command assembles the named file. The following flags may be specified in any order:

- o *objfile* Puts the output of the assembly in *objfile*. By default, the output filename is formed by removing the *.s* suffix, if there is one, from the input filename and appending a *.o* suffix.
- n Turns off long/short address optimization. By default, address optimization takes place.
- m Runs the *m4* macro processor on the input to the assembler.
- R Removes (unlink) the input file after assembly is completed.
- dl Does not produce line number information in the object file.
- T Truncates symbols to eight characters.
- V Writes the version number of the assembler being run on the standard error output.
- Y [*md*],*dir* Finds the **m4** preprocessor (**m**) and/or the file of predefined macros (**d**) in directory *dir* instead of in the customary place.

**FILES**

*TMPDIR*/\* temporary files

*TMPDIR* is usually */tmp* but can be redefined by setting the environment variable **TMPDIR** [see *tempnam()* in *tempnam(3S)*].

**SEE ALSO**

cc(1), ld(1), m4(1), nm(1), strip(1), tmpnam(3S), a.out(4).

*Programmer's Guide: CTIX Supplement.*

**WARNING**

If the **-m** (*m4* macro processor invocation) option is used, keywords for *m4* [see *m4(1)*] cannot be used as symbols (variables, functions, or labels) in the input file since *m4* cannot determine which are assembler symbols and which are real *m4* macros.

**CAVEATS**

Arithmetic expressions may only have one forward referenced symbol per expression.

**NOTES**

Wherever possible, the assembler should be accessed through a compilation system interface program [such as *cc(1)*].

**NAME**

cc - C compiler

**SYNOPSIS**

cc [ options ] files

**DESCRIPTION**

The *cc* command is the interface to the C Compilation System. The compilation tools consist of a preprocessor, optimizing compiler, assembler, and link editor. The *cc* command processes the supplied options and then executes the various tools with the proper arguments. The *cc* command accepts several types of files as arguments.

Files whose names end with *.c* are taken to be C source programs and may be preprocessed, compiled, optimized, assembled, and link edited. The compilation process may be stopped after the completion of any pass if the appropriate options are supplied. If the compilation process runs through the assembler, then an object program is produced and is left in the file whose name is that of the source with *.o* substituted for *.c*. However, the *.o* file is normally deleted if a single C program is compiled and then immediately link edited. In the same way, files whose names end in *.s* are taken to be assembly source programs, and may be assembled and link edited; files whose names end in *.i* are taken to be preprocessed C source programs and may be compiled, optimized, assembled, and link edited. Files whose names do not end in *.c*, *.s*, or *.i* are handed to the link editor.

Since the *cc* command usually creates files in the current directory during the compilation process, it is necessary to run the *cc* command in a directory in which a file can be created. The following options are interpreted by *cc*.

**-#**

**-##** These options cause *cc* to display each command that it would generate if it were to execute, but to fully execute only in the case of **-#**. Thus, **-#** specifies execution in verbose mode, and **-##** specifies verbose mode (what *cc* would do if it were to execute), but does nothing.

**-c**

Suppresses the link editing phase of the compilation, and does not remove any produced object files.

**-g**

Causes the compiler to generate additional information needed for the use of *sdb* (1).

**-o outfile**

Produces an output object file by the name *outfile*. The name of the default file is **a.out**. This is a link editor option.

- p** Arranges for the compiler to produce code that counts the number of times each routine is called; also, if link editing takes place, profiled versions of *libc.a* and *libm.a* (with *-lm* option) are linked and *monitor(3C)* is automatically called. A **mon.out** file will then be produced at normal termination of execution of the object program. An execution profile can then be generated by use of *prof(1)*.
- w** Suppresses warnings. Previous versions of the compiler passed the **-w** option to the linker (*ld*) to suppress warnings from *ld* about truncated names. Use **-wl,-w** instead.

**-Bstring****-t/p02a/**

These options will be removed in the next release. Use the **-Y** option.

- E** Runs only *cpp(1)* on the named C programs, and sends the result to the standard output.
- H** Prints out on *stderr* the pathname of each file included during the current compilation.
- O** Does compilation phase optimization. This option will not have any affect on *.s* files.
- On** Optimizes code, where *n* is a decimal number specifying the optimizations to be done:
  - 2 Branch optimization. (Jump to jump, and so forth.)
  - 4 Strength reduction, like replacing constant multiplications with shifts and adds.
  - 8 Reaching analysis. Used by common subexpression eliminations, constant propagations, loop optimizations, and so forth.
  - 16 Loop optimizations. Move loop-invariants out of loop, replace array references with pointers, invert loop-index and count to zero, loop unrolling, and so forth.
  - 32 Lifetime analysis. Lets variables share registers.
  - 64 Remove useless code, like unreachable statements and assignments to unused variables.
  - 128 Optimize more than one function at a time. See **-Xparse-size (-X20)**.
  - 256 Remove link/unlink, if possible.

Other optimizations, like peephole and automatic register allocation are always done.

Optimizations can be combined by adding their codes; for example, to have both strength reduction and reaching analysis, *n* would be specified as **12**. **-O** without a number implies **-O511**, which is a combination of all optimizations listed above. If **-O** is not given, then **-O6** is implied.

- P** Runs only *cpp*(1) on the named C programs and leaves the result in corresponding files suffixed *.i*. This option is passed to *cpp*(1).
- S** Compiles and does not assemble the named C programs, and leaves the assembler-language output in corresponding files suffixed *.s*.
- T** Truncates variable names to eight characters. Tells the loader to match eight character names (same as **-G** in the loader).
- V** Displays the current version number.

**-Wc, arg1[, arg2...]**

Hands off the argument[s] *argi* to pass *c* where *c* is one of [**p0al**] indicating the preprocessor, compiler, assembler, or link editor, respectively. For example: **-Wa,-m** passes **-m** to the assembler.

- X** Since the descriptions for these options take up so many pages, the **-X** options are listed after the **-Y** options below.

**-Y [p0alSILUc], dirname | processor**

Specifies a new pathname, *dirname*, for the locations of the tools and directories designated in the first argument; or selects a processor type, *processor*, for which to generate code. [**p0alSILUc**] represents:

- p** preprocessor
- 0** compiler
- a** assembler
- l** link editor
- S** directory containing the start-up routines
- I** default include directory searched by *cpp*(1)
- L** first default library directory searched by *ld*(1)
- U** second default library directory searched by *ld*(1)
- c** selects the processor type, specified by the second argument, for which to generate code: **68040**, **68020**, **68010**, **68881**. For example, **-Y c,68020** selects the 68020 processor with software floating-point instructions. Note that **68881** implies **68020**, and that if a compile is being done for a 68000- or 68010-based system, *occ* is called automatically.

If the location of a tool is being specified, then the new pathname for the tool will be *dirname/tool*. If more than one **-Y** option is applied to any one tool or directory, then the last occurrence holds.

**-Xname**

**-Xn** The **-X** switch tells the compiler how to behave in certain areas. There are two ways to set these options, either by a number, **-Xn**, or with a mnemonic, **-Xname**.

Some options can be set to a decimal value, *m*, by using an equals sign, **-Xn=m** or **-Xname=m**.

To turn off an option, set it to zero: **-Xn=0** or **-Xname=0**. If nothing else is said, the default value of *m* is 1. For example, **-X6**, **-X6=1**, and **-Xtest-at-top** are the same option.

Currently the following **-X** options are implemented:

**-Xstruct-compress**

**-X1=1** Uses same alignment in structure as the member with the biggest alignment requirement. On systems where the minimum structure alignment is always bigger than 1, this option can be used to match external hardware.

**-Xstruct-unaligned**

**-X1=2** (Default) Does not align structures when pushing them on the stack as arguments.

**-Xstruct-default**

**-X1=0** Uses default structure alignment.

**-Xmismatch-warning**

**-X2** (Default) Generates only a warning instead of a fatal error when pointers of different types or integers are mixed in expressions. For example, the line

```
long i1, i2 = &i1;
```

is illegal in the C programming language, but some older programs depend on the compiler to handle lines of code like this, anyway. **-X2** is also set by **-Xpcc** (**-X7=3**).

**-Xuse-float**

**-X3** Forces the compiler not to convert float operands to double in expressions and as function arguments.

**-Xmemory-is-volatile**

**-X4** Does not do optimizations that can make drivers (or other programs) fail. By default, the compiler tries to store as much data in registers as is possible, whenever it is safe. Problems occur if a memory location changes because it is mapped to some external hardware and the compiler, unaware of this change, continues to use the old value that is stored in a register. These situations can be handled with the ANSI-C keyword **volatile**, but to be able to compile older programs, **-Xmemory-is-volatile** is provided.

**-Xlocals-on-stack**

**-X5** By default, the compiler tries to allocate all local variables to registers. If **-Xlocals-on-stack** is given, only variables declared with the register keyword are assigned to registers.

**-Xtest-at-bottom**

**-X6=0** Uses one loop test at the bottom of a loop. The default is X6=1.

**-Xtest-at-top**

**-X6=1** (Default) Uses one loop test at the top of a loop.

**-Xtest-at-both**

**-X6=2** Forces the compiler to always test loops both before the loop is started and at the bottom of the loop. Use this option to create the fastest possible code, although using somewhat more space. Even if **-Xtest-at-both** is not set, other optimizations sometimes make the compiler generate double tests.

**-Xk-and-r**

**-X7=0** With this option, the compiler follows the C language standard as defined by the Kernighan & Ritchie C reference manual, but with all new ANSI-C features added. Where Kernighan & Ritchie and ANSI differs, **-Xk-and-r** follows Kernighan & Ritchie. See Table 1, later in this section, which explains the cases where the **-X7** switch has an effect. The default is X7=3.

**-Xpcc**

**-X7=3** (Default) With this option, the compiler follows the C standard as defined by the Unix System V Release 3 C compiler. See Table 1 for details.

**-Xenum-is-small**

**-X8=0** Uses the smallest integer type possible for **enums**.

**-Xenum-is-int**

**-X8** (Default) The **enum** type is always equal to **int**. This option is also set by the **-Xpcc (-X7=3)** option.

**-Xforce-declarations**

**-X9** With this option, the compiler generates warnings if a function is used without a previous declaration. This option is useful in combination with prototypes to make C a strongly “typed” language.

**-Xstack-probe**

**-X10** Stack checking (probing) is done on some machines. This can be avoided by giving **-X10**.

**-Xpass-source**

**-X11** Outputs the C source as comments in the generated assembly language code.

**-Xsigned-bitfields**

**-X12=0** Bitfields without the **signed** or **unsigned** keyword are treated as signed integers.

**-Xunsigned-bitfields**

**-X12** (Default) Bitfields without the **signed** or **unsigned** keyword are treated as unsigned integers.

**-Xswap-cr-nl**

**-X13** Swaps **\n** and **\r**. Used on systems where carriage return and linefeed are reversed.

**-Xsuppress-warnings**

**-X14** Same as the **-w** switch. No warnings are generated.

**-Xunroll=m**

**-X15=m**

Unrolls small loops this number of times. This is set to 2 as a default if **-O** is given. The number *m* must be a power of 2.



**-Xunroll-size=*m***

**-X16=*m***

Specifies the number of nodes a loop can contain at the most, to be considered for loop unrolling. Every operator and every operand count as one node, thus the expression **a=b-c**; contains five nodes. The number *m* is set to 20 as a default if **-O** is given.

**-Xstruct-best-align**

**-X17** If this option is given, members of a structure are aligned on their natural alignment instead of the default alignment. This switch produces faster but noncompatible code on machines that put integers on 2-byte boundaries to remain compatible with preceding processors.

**-Xinline=*m***

**-X19=*m***

Inline functions with less than *m* nodes. This option is available from release 2.36 of the compiler.

**-Xparse-size=*m***

**-X20=*m***

Waits with code generation of functions until *m* KB of internal memory is used. By delaying generation, the compiler can do intrafunction optimizations like inlining and register tracking. This option is set to 500 as a default.

**-Xbottom-up-init**

**-X21** (Default) Both Kernighan & Ritchie and ANSI-C specify clearly that structure and array initializations with missing braces should be parsed top-down, but some C compilers parse these bottom-up instead. For example:

```

struct z { int a, b; };
struct x {
    struct z    z1[2];
    struct z    z2[2];
} x = { { 1,2},{3,4} };

/* could be parsed as
*    { { 1,2},{0,0} } , { {3,4},{0,0} } }; ansi & k&r
* or   { { 1,2},{3,4} } , { {0,0},{0,0} } }; bottom-up
*/

```

**-Xbottom-up-init** makes the compiler parse the above example bottom-up. This option is set when **-Xpcc (-X7=3)** is set.

**-Xtruncate=*m***

**-X22=*m***

Truncates all identifiers after *m* characters.

**-Xptr-values-in-a0**

**-X32** On 68XXX machines that return pointer function values both in register **d0** and **a0**, this option forces the compiler to use the **a0** value instead of the **d0** value. This option is useful when mixing assembler written routines that only return values in **a0**, with C routines.

**-Xno-libc-inlining**

**-X33** Sometimes some **libc** routines like **strcpy()** can be inlined by the compiler. This option prevents this.

**-X34=0** The MC68020 with the MC68881/2 floating point co-processor used the **fintrz** instruction to truncate floating-point values before converting them to a 32-bit integer. This causes a problem on the MC68040, because that instruction is not implemented in hardware; rather, it is emulated in the kernel.

**-X34=1** (Default) This switch makes the compiler set the floating-point rounding-mode to “round-to-zero” before every float-to-integer conversion and resetting it afterwards. This is done to avoid the kernel-trap that the **fintrz** instruction causes on the MC68040.

**-X34=2** This switch uses neither of the above methods to round a floating-point value. It is considerably faster than these constructs, but is dependent on rounding-mode in the processor being set to “round-to-zero” in advance. Note that this mode produces illegal programs if the rounding-mode is not set correctly.

**-X35=0** (Default) A 16-bit relative jump-table is to be used for switch-statements. This can cause problems in huge switch-statements (> 32K). The code will be smaller but somewhat slower.

**-X35=1** A 32-bit absolute jump-table is to be used for switch-statements. The code is bigger but runs faster.

The **-X7** switch (**-Xk-and-r**, and **-Xpcc**) controls the way the compiler behaves in certain areas. Table 1 explains the cases where this switch has an effect.

Table 1

Case	K-&-R	PCC
<b>long float</b> is same as <b>double</b> .	y	y
The <b>asm</b> keyword is defined.	y	y
The <b>volatile</b> , <b>const</b> , and <b>signed</b> keywords are defined.	y	n
The type of the hexadecimal constant 0xffffffff is unsigned int (u) or int (i).	i	i
Declarations inside a function using the <b>extern</b> keyword have block (b) or file (f) scope. File scope is the same as if the declaration was made outside of the function.	f	f
In ANSI-C, it is legal to initialize automatic arrays and structures/unions. The compiler always accepts this and is either silent (s) or gives a warning (w).	s	w
When two integers are mixed in an expression, they cause conversions and the result type is either "unsigned wins" (u) or "smallest possible wins" (s). For example, the expression,  $((\text{unsigned char})1 > -1)$ is 0 if (u) and 1 if (s).	u	u
When prototypes are used and the arguments do not match, the compiler will generate an error (e) or warning (w).	w	w

Table 1 (Continued)

Case	K-&-R	PCC
Float expressions are computed in float (f) or double (d).	d	d
Not fully braced structure and array initializers can either be parsed top-down (t) or bottom-up (b). This is the same as the <b>-Xbottom-up (-X21)</b> option.	t	b
When pointers and integers are mismatched, the compiler will generate an error (e) or a warning (w). Same as the <b>-Xmismatch-warning</b> .	e	w
When an array is declared illegally without a dimension, the compiler will give an error (e) or warning (w).	e	w
Trigraphs, for example, ?? sequences, are either recognized (r) or not (n).	r	n
Illegal structure references give either an error (e) or a warning (w): <b>int *p; p-&gt;a = 1;</b>	e	w
Comments are replaced by nothing (n) or a space (s).	n	n
Macro arguments are replaced in strings and in character constants: <b>#define x(a) if (a) printf("a\n");</b>	y	y

Table 1 (Continued)

Case	K-&R	PCC
Preprocessor errors are either errors (e) or warnings (w).	e	w
<code>__STDC__</code> macro is predefined.	n	n
Spaces are legal before <code>cpp</code> <code>#directives</code> .	n	n
Predefined macros like <code>unix</code> , <code>m68k</code> , and so on, are available.	y	y

The `cc` command also recognizes `-C`, `-D`, `-H`, `-I`, and `-U` and passes these options and their arguments directly to the preprocessor without using the `-W` option. Similarly, the `cc` command recognizes `-l`, `-m`, `-o`, `-r`, `-s`, `-t`, `-u`, `-w`, `-x`, `-z`, `-F`, `-G`, `-L`, `-M`, `-N`, `-V`, and `-Z` and passes these options and their arguments directly to the loader. See the man pages for `cpp(1)` and `ld(1)` for descriptions.

Other arguments are taken to be C-compatible object programs, typically produced by an earlier `cc` run or perhaps libraries of C-compatible routines, and are passed directly to the link editor. These programs, together with the results of any compilations specified, are link edited (in the order given) to produce an executable program with name `a.out` unless the `-o` option of the link editor is used.

The C compiler uses one of four code generators for the 68010, 68020, 68020/68881, and 68040. If 68010 is selected, `cc` invokes `occ` and passes all options directly to `occ`. There are several ways to select a particular code generator, but the selection is normally done using one of two basic mechanisms.

The first is to specify the processor on the `cc` command line, for example, by using the `-Y` option. (An equivalent mechanism is provided by the `gencc(1M)` command, and also by the `cc1sw(1)`, `cc2sw`, or `cc2fp` command.) The `-Y` option has additional arguments that allow you to specify pathnames of default libraries, include files, and tools as described earlier.

The second mechanism is to use the `CENVIRON` shell variable. Note that the first mechanism, specifying the processor and/or search path of libraries and include files, overrides the `CENVIRON` and any other shell variable settings.

The CENVIRON variable has the following syntax:

```
CPU=xxxxx,FPU=yyyyy
```

where CPU indicates the central processor for which code is to be generated and FPU indicates the style of floating-point math to use. Use the following table to determine which combinations are legal and which style of floating-point math will be used.

CPU	FPU		
	SOFTWARE	68881	FPU = omitted
68000	software	illegal	software
68010	software	illegal	software
68020	software	hardware	software
68040	illegal	illegal	hardware

If you are compiling for a 68040-based system but you want to use software floating point, use CPU=68020 and FPU=SOFTWARE, or just CPU=68020.

The FPU parameter may be omitted; the default is either SOFTWARE or HARDWARE, as shown in the table above. The CENVIRON variable should always be set to the appropriate values in the **.profile** or **.cshrc** files or in the makefile. [See *hinv*(1M).]

The C compiler interprets two shell variables which, along with the CENVIRON variable, allow cross-compilation for any CTIX machine:

- LIBROOT**        This variable is a path that is prepended to normal library names when searching for a library. See also *ld*(1).
- INCROOT**        This variable is a path that is prepended to the **/usr/include** and **/usr/include/sys** directories during include file searches. See also *cpp*(1).

The C language standard was extended to allow arbitrary length variable names. The option pair “**-Wp,-T -W0,-XT**” will cause *cc* to truncate arbitrary length variable names to eight characters.

## FILES

- file.c            C source file  
 file.i            preprocessed C source file  
 file.o            object file

file.s	assembly language file
a.out	link edited output
<i>LIBDIR</i> /*crt1.o	start-up routine
<i>LIBDIR</i> /crt0.o	start-up routine
<i>TMPDIR</i> /*	temporary files
<i>BINDIR</i> /as	assembler, <i>as</i> (1)
<i>BINDIR</i> /ld	link editor, <i>ld</i> (1)
<i>LIBDIR</i> /libc.a	standard C library
<i>LIBDIR</i> /libc_s.a	standard C shared library

*LIBDIR* is usually /lib

*BINDIR* is usually /bin

*TMPDIR* is usually /tmp but can be redefined by setting the environment variable **TMPDIR** [see *tempnam()* in *tempnam(3S)*].

#### SEE ALSO

*as*(1), *ld*(1), *cpp*(1), *gcc*(1), *lint*(1), *prof*(1), *sdb*(1), *tempnam*(3S).

Kernighan, B. W., and Ritchie, D. M., *The C Programming Language*, Prentice-Hall, 1978.

*Programmer's Guide: CTIX Supplement*

#### CAVEATS

*cc* will complain if it encounters inconsistencies between the processor selected and default libraries or include files. *occ* is invoked to generate code for 68000 or 68010 processors.

Sometimes the range for a branch does not fit inside a word; in this case, an error message is printed. For suggested workarounds, see the section called "Span-Dependent Optimization" in Chapter 14 of the *Programmer's Guide: CTIX Supplement*.

#### DIAGNOSTICS

The diagnostics produced by the C compiler are sometimes cryptic. Occasional messages may be produced by the assembler or link editor.

#### NOTES

By default, the return value from a compiled C program is completely random. The only two guaranteed ways to return a specific value is to explicitly call *exit*(2) or to leave the function *main*() with a "return expression;" construct.

—

—

—



**NAME**

*cc1sw*, *cc2sw*, *cc2fp*, *cc4* - front-end to the *cc* command

**SYNOPSIS**

**cc1sw** [ options ] files

**cc2sw** [ options ] files

**cc2fp** [ options ] files

**cc4** [ options ] files

**DESCRIPTION**

*cc1sw*, *cc2sw*, *cc2fp*, and *cc4* provide a front-end to *cc* for use in cross-compilation. *cc1sw* generates code for a 68010 processor with software floating point, *cc2sw* generates code for a 68020 processor with software floating point, *cc2fp* generates code for a 68020 processor with hardware floating point, and *cc4* generates code for a 68040 processor. The commands call *cc* with the following **-Y** options:

*cc1sw*   **-Y c,68010**  
           **-Y S,/cross/1sw/lib**  
           **-Y L,/cross/1sw/lib**  
           **-Y U,/cross/1sw/usr/lib**

*cc2sw*   **-Y c,68020**  
           **-Y S,/cross/2sw/lib**  
           **-Y L,/cross/2sw/lib**  
           **-Y U,/cross/2sw/usr/lib**

*cc2fp*   **-Y c,68881**  
           **-Y S,/cross/2fp/lib**  
           **-Y L,/cross/2fp/lib**  
           **-Y U,/cross/2fp/usr/lib**

*cc4*      **-Y c,68040**  
           **-Y S,/cross/2fp/lib**  
           **-Y L,/cross/2fp/lib**  
           **-Y U,/cross/2fp/usr/lib**

*Options* are those options available for *cc*.

The default include directories searched by *cc* (called by *cc1sw*, *cc2sw*, or *cc2fp*) are **/usr/include** and **/usr/include/sys**. The default include directories can be overridden by using the **-Y I,dirname** option or setting the **INCROOT** environment variable [see *cc(1)*].

**FILES**

<code>file.c</code>	C source file
<code>file.i</code>	preprocessed C source file
<code>file.o</code>	object file
<code>file.s</code>	assembly language file
<code>a.out</code>	link edited output
<code><i>LIBDIR</i>/*crt1.o</code>	start-up routine
<code><i>LIBDIR</i>/crt0.o</code>	start-up routine
<code><i>TMPDIR</i>/*</code>	temporary files
<code><i>LIBDIR</i>/cpp</code>	preprocessor, <i>cpp</i> (1)
<code><i>BINDIR</i>/as</code>	assembler, <i>as</i> (1)
<code><i>BINDIR</i>/ld</code>	link editor, <i>ld</i> (1)
<code><i>LIBDIR</i>/</code>	standard C library
<code><i>LIBDIR</i>/libc_s.a</code>	standard C shared library

*LIBDIR* is usually `/lib`

*BINDIR* is usually `/bin`

*TMPDIR* is usually `/tmp` but can be redefined by setting the environment variable **TMPDIR** [see *tempnam*( ) in *tempnam*(3S)].

**NOTES**

`cc2sw` can be used for generating software floating point on 68040-based systems.

**SEE ALSO**

`cc`(1), `gcc`(1M).

## NAME

config - configure a CTIX system

## SYNOPSIS

```
/etc/config [ -l file ] [ -c file ] [ -m file ] [ -t ] [ -b num ] [ -d num ]
[ -s num ] [ -f num ] dfile
```

## DESCRIPTION

The *config* program takes a description of a CTIX system, generates a configuration table file, and generates a hardware interface file. The configuration table file is a C program defining the configuration tables for the various devices on the system. The hardware interface file provides information about the interface between the hardware and device handlers.

The options to *config*(1M) are as follows:

- l Specifies the name of the hardware interface file; **low.s** is the default.
- c Specifies the name of the configuration table file; **conf.c** is the default.
- m Specifies the name of the file that contains information about supported devices; **/etc/master** is the default name. This file is supplied with the CTIX system and should *not* be modified unless the user *fully* understands its construction.
- t Requests a short table of major device numbers for character- and block-type devices. This can facilitate the creation of special files.
- b Specifies the minimum number of entries in the *bdevsw* array. The default value is **20**.
- d Specifies the minimum number of entries in the *cdevsw* array. The default value is **128**.
- s Specifies the minimum number of entries in the *fmodsw* array. The default value is **16**.
- f Specifies the minimum number of entries in the *fstypsw* array. The default value is **8**. The **-b**, **-d**, **-s**, and **-f** options are provided to ensure that a sufficient number of empty slots are available for loadable modules, such as drivers, stream modules, software modules, and file system types.

The user must supply *dfile*, which must contain device information for the user's system. The *dfile* is divided into two parts: the first part contains physical device specifications; the second part contains system-dependent information. Any line with an asterisk (\*) in column 1 is a comment. A sample *dfile* file is provided in the **/usr/sys/cf** directory.

### First Part of *dfile*

Each line in the first part of the *dfile* contains one field, *devname*, which is the name of the device, software module, stream module, or file system type (as it appears in the */etc/master* device table).

The disk driver section is the first group of the first part; this first group must contain only disk driver *devname* entries.

Note that for disk controllers, the position of *devname* in *dfile* determines the */dev/dsk* controller number assigned to the *devname* driver. For tape controllers, the position of *devname* has no significance. The tape controller number mapping is performed by the CTIX installation tools, using the **mknod(1M)** command.

The following example shows how controller numbers are assigned from an S/640 or S/480 *dfile*:

<i>dfile</i> Entry	Controller Number
<b>diskonbd</b>	<i>/dev/dsk/c0d?</i> (ST506)
<b>Vsmd3200</b>	<i>/dev/dsk/c1d?</i> (1st SMD)
<b>scsidisk</b>	<i>/dev/dsk/c2d?</i> (SCSI)
<b>Vsmd3200</b>	<i>/dev/dsk/c3d?</i> (2nd SMD)
:	
<b>scsi</b>	(required line for SCSI)
<b>stape</b>	(SCSI QIT and HIT)

In the example, above the quarter-inch tape (QIT) drive is a SCSI device, assigned to */dev/rmt/c0d?*. The VME-based half-inch tape controller is assigned */dev/rmt/c1d?* by the installation tools. The driver (*/etc/lddrrv/ipt.o*) is loaded dynamically at boot time, so **ipt** is not entered in the *dfile*.

For the S/120, S/22x, and S/320, disk controller numbers are assigned as they are on the S/640 and S/480, but tape controller numbers are assigned differently, as shown below:

<i>dfile</i> Entry	Controller Number
:	
<b>qic</b>	<i>/dev/rmt/c0d0</i> (QIC-2)
<b>scsi</b>	(required line for SCSI)
<b>stape</b>	(SCSI QIT and HIT)

As shown above, the first quarter-inch tape drive is QIC-2 controller-based; it is assigned */dev/rmt/c0d0*. Again, the VME-based half-inch tape drive controller

is assigned `/dev/rmt/c1d?` by the installation tools. The SCSI quarter-inch tape and half-inch tape drives are assigned `/dev/rmt/c0d1` to `/dev/rmt/c0d7`.

For the S/80 and S/280, the controller number assignment is different, because the onboard disk controller is the SCSI controller and there is no VME expansion:

<i>dfile</i> Entry	Controller Number
<b>scsidisk</b>	<code>/dev/dsk/c0d?</code> (SCSI)
:	
<b>scsi</b>	(required line for SCSI)
<b>stape</b>	(SCSI QIT and HIT)

As shown above, the first quarter-inch tape drive is assigned `/dev/rmt/c0d0`. Remaining tape drives are assigned `/dev/rmt/c0d?` (where ? is 1 through 7).

### Second Part of *dfile*

The second part of the *dfile* contains four types of lines, listed and described below. Note that *all* specifications of this part *are required*, although the order is arbitrary.

1. *Root/pipe device specification*

Two lines of three fields each:

```

root devname minor
pipe devname minor

```

where *minor* is the minor device number (in decimal) of the slice on the fixed disk.

2. *Swap device specification*

One line that contains five fields, as follows:

```

swap devname minor swplo nswap

```

where *swplo* is the lowest disk block (decimal) in the swap area and *nswap* is the maximum number of 1-KB disk blocks (decimal) in the swap area. The kernel sizes the actual swap area size and configures itself for up to this maximum.

3. *Dynamic device number assignment*

The *devnames* for **root**, **swap**, and **pipe** can be specified as **any**. The major device numbers are undetermined until boot time.

The key word parameter **dynamic** can be used with the **any devname** to force the major device number of the **boot** device to 0. For example, if the **devnames** in the *dfile* are **diskonbd** and **scsidisk**, and the **boot** device is the **scsidisk**, the kernel swaps the device numbers so that **scsidisk** can be accessed through major device number 0 instead of 1, as specified in the *dfile*. All access to SCSI disks is through **/dev/dsk/c0d?s?**, and all access to ST506 disks is through **/dev/dsk/c1d?s?**. If **any** is not used, **dynamic** has no effect.

#### 4. *Parameter specification*

Any number of lines of two fields each, chosen from the following list. *Number* is decimal. Note that the following parameter list is *not* complete; parameters not on the list either must not be changed or have no effect.

<b>buffers</b>	<b>number</b>	<i>/* number of 1024-byte file system caching buffers */</i>
<b>buffers_4k</b>	<b>number</b>	<i>/* number of 4096-byte file system caching buffers */</i>
<b>dmmxsiz</b>	<b>number</b>	<i>/* max number of pages per loadable driver */</i>
<b>inodes</b>	<b>number</b>	<i>/* max open inodes in system */</i>
<b>s5inodes</b>	<b>number</b>	<i>/* max open s5inodes in system */</i>
<b>files</b>	<b>number</b>	<i>/* max open files in system */</i>
<b>flkrec</b>	<b>number</b>	<i>/* max locks active in system */</i>
<b>mounts</b>	<b>number</b>	<i>/* max file systems mounted */</i>
<b>regions</b>	<b>number</b>	<i>/* total number of regions in system */</i>
<b>procs</b>	<b>number</b>	<i>/* max processes in system */</i>
<b>maxproc</b>	<b>number</b>	<i>/* max processes per user ID */</i>
<b>maxfsiz</b>	<b>number</b>	<i>/* ulimit default in 512-byte blocks */</i>
<b>maxumem</b>	<b>number</b>	<i>/* max number of pages per process */</i>
<b>cbufsize</b>	<b>number</b>	<i>/* console circular buffer size in bytes */</i>
<b>msgmax</b>	<b>number</b>	<i>/* max chars in a message */</i>
<b>msgmni</b>	<b>number</b>	<i>/* max active message queues */</i>
<b>msgmnb</b>	<b>number</b>	<i>/* max total chars in message queues */</i>
<b>msgtql</b>	<b>number</b>	<i>/* max messages in system */</i>
<b>msgssz</b>	<b>number</b>	
<b>msgseg</b>	<b>number</b>	<i>/* msgssz * msgseg = number bytes of system buffering */</i>
<b>nlldrv</b>	<b>number</b>	<i>/* max number of loadable drivers */</i>
<b>semnmi</b>	<b>number</b>	<i>/* max active semaphores */</i>
<b>semnms</b>	<b>number</b>	<i>/* max semaphores in system */</i>
<b>semmsi</b>	<b>number</b>	<i>/* max semaphores per ID */</i>

```

semopm    number /* max operations per semop call */
semume    number /* max undo structures per process */
semnmu    number /* max undo structures in system */
diriosz   number /* direct I/O default size */
shmmax    number /* max bytes in a shared segment */
shmmn     number /* min bytes in a shared segment */
shmmni    number /* max active shared segments */
shmseg    number /* max attached segments per process */
shmbk     number /* gap in pages between data and
                  shared memory */

nqueue    number /* max stream queues */
nstream   number /* max streams */
nbik4096  number /* number of 4096 byte stream bufs */
nbik2048  number /* number of 2048 byte stream bufs */
nbik1024  number /* number of 1024 byte stream bufs */
nbik512   number /* number of 512 byte stream bufs */
nbik256   number /* number of 256 byte stream bufs */
nbik128   number /* number of 128 byte stream bufs */
nbik64    number /* number of 64 byte stream bufs */
nbik16    number /* number of 16 byte stream bufs */
nbik4     number /* number of 4 byte stream bufs */
shlbmax   number /* max # of shared libs per process */
nofiles   number /* max # of open files per process */
ntimod    number /* max # of TLI connections */
ntirdwr   number /* max # of TLI read/write connections */
nsp       number /* max # of stream pipes */

```

Certain parameters, if set to 0, allow the kernel to autoconfigure. For example, **procs**, **regions**, **clists**, **i\_nodes**, **s5inodes**, **files**, and **buffers** are autoconfigurable. The value of **procs** is based on the number of users; the values for **regions**, **i\_nodes**, **s5inodes**, and **files** are based on the value of **procs**. The value of **clists** is based on the number of serial and cluster ports. The value of **buffers** is based on the amount of physical memory. Any or all of the autoconfigured values can be overridden. The value of **maxumem** can also be set to 0; in which case, it floats between 1 MB and one quarter of the total swap space.

#### EXAMPLE

This example assumes an S/640 system with the following devices:

- onboard ST506 disks (root)
- first interphase SMD disk controller

- SCSI disks
- second interphase SMD disk controller
- RS-232-C (any number of ports)
- SCSI tape drives
- one parallel line printer
- root device is a disk (drive 0, section 1)
- pipe device is a disk (drive 0, section 1)
- swap device is a disk (drive 0, section 2), with a swplo of 1 and an nswap of 8000
- number of buffers is 100
- number of processes is 100
- maximum number of processes per user ID is 25
- number of mounts is 6
- number of inodes is 100
- number of files is 120
- number of character buffers is 64
- messages are to be included
- semaphores are to be included

The S/640 system configuration would be specified as follows in the *dfile*:

```

diskonbd
Vsmd3200
scsidisk
Vsmd3200
serial
scsi
stape
console
plp
root    any           01
pipe   any           01
swap   any           02  0    8000
* Comments are inserted in this manner
buffers  100
procs   100
maxproc 25
mounts  6
inodes  100
files   120
mesg    1
sema    1
clists  64
    
```



**FILES**

/etc/master	default input master device table
/usr/sys/cf/dfile	default system configuration
/usr/sys/cf/low.s	default output hardware interface
/usr/sys/cf/conf.c	default output configuration table

**SEE ALSO**

ldeeprom(1M), uconf(1M), master(4).  
*S!Series CTIX Administrator's Guide.*

**DIAGNOSTICS**

Diagnostics are routed to the standard output and are self-explanatory.

**BUGS**

The **-t** option does not know about devices that have aliases.

—

—

—

**NAME**

cpp - the C language preprocessor

**SYNOPSIS**

**LIBDIR/cpp** [ option ... ] [ ifile [ ofile ] ]

**DESCRIPTION**

The C language preprocessor, *cpp*, is invoked as the first pass of any C compilation by the *cc*(1) command. Thus, *cpp*'s output is designed to be in a form acceptable as input to the next pass of the C compiler. If you are running a 6.3 or later release of CTIX, *cpp* is a built-in feature of the C compiler. See *m4*(1) for a general macro processor. The use of *cpp* other than through the *cc*(1) command is not suggested, since the functionality of *cpp* has been moved in the latest release.

*cpp* optionally accepts two filenames as arguments. *Ifile* and *ofile*, respectively, are the input and output for the preprocessor. They default to standard input and standard output if not supplied.

The following *options* to *cpp* are recognized:

- P Preprocesses the input without producing the line control information used by the next pass of the C compiler.
- C By default, *cpp* strips C-style comments. If the *-C* option is specified, all comments (except those found on *cpp* directive lines) are passed along.
- U*name* Removes any initial definition of *name*, where *name* is a reserved symbol that is predefined by the particular preprocessor. Following is the current list of these possibly reserved symbols.

operating system:	unix, gcos, ibm, os, tss
hardware:	interdata, mc68k, mc68k32, m68k, mc68000, mc68010, mc68020, mc68881, mc68040, pdp11, u370, u3b, u3b5, u3b2, u3b20d, vax
UNIX system variant:	RES, RT
<i>lint</i> (1):	lint

**-D*name*****-D*name*=*def***

Defines *name* with value *def* as if by a **#define**. If no *=def* is given, *name* is defined with value 1. The **-D** option has lower precedence than the **-U** option. That is, if the same name is used in both a

-U option and a -D option, the name will be undefined regardless of the order of the options.

- T The -T option forces *cpp* to use only the first eight characters to distinguish preprocessor symbols and is included for backward compatibility.
- Idir Changes the algorithm for searching for **#include** files whose names do not begin with / to look in *dir* before looking in the directories on the standard list. Thus, **#include** files whose names are enclosed in " " will be searched for first in the directory of the file with the **#include** line, then in directories named in -I options, and last in directories on a standard list. For **#include** files whose names are enclosed in <>, the directory of the file with the **#include** line is not searched. By default, *cpp* searches for the name enclosed in < > in */usr/include*; however, if the shell variable INCROOT is set, *cpp* prepends the value of INCROOT to the standard list. This is particularly useful for cross-compilation.
- Ydir Uses directory *dir* in place of the standard list of directories when searching for **#include** files. The -Y option overrides the value for INCROOT if it is set.
- H Prints, one per line on standard error, the pathnames of included files.

Two special names are understood by *cpp*. The name `__LINE__` is defined as the current line number (as a decimal integer) as known by *cpp*, and `__FILE__` is defined as the current filename (as a C string) as known by *cpp*. They can be used anywhere (including in macros) just as any other defined name.

All *cpp* directive lines start with # in column 1. Any number of blanks and tabs is allowed between the # and the directive. The directives are as follows:

**#define** *name token-string*

Replaces subsequent instances of *name* with *token-string*.

**#define** *name( arg, ..., arg ) token-string*

Notice that there can be no space between *name* and the (. Replace subsequent instances of *name* followed by a (, a list of comma-separated sets of tokens, and a ) followed by *token-string*, where each occurrence of an *arg* in the *token-string* is replaced by the corresponding set of tokens in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded *token-string* unchanged. After the entire *token-string* has been expanded, *cpp* restarts its scan for names to expand at the beginning of the newly created *token-string*.

**#undef** *name*

Causes the definition of *name* (if any) to be forgotten from now on. No additional tokens are permitted on the directive line after *name*.

**#ident** "*string*"

Puts *string* into the .comment section of an object file.

**#include** "*filename*"**#include** <*filename*>

Includes at this point the contents of *filename* (which will then be run through *cpp*). When the <*filename*> notation is used, *filename* is only searched for in the standard places. See the **-I** and **-Y** options above for more detail. No additional tokens are permitted on the directive line after the final " or >.

**#line** *integer-constant* "*filename*"

Causes *cpp* to generate line control information for the next pass of the C compiler. *Integer-constant* is the line number of the next line and *filename* is the file from which it comes. If "*filename*" is not given, the current filename is unchanged. No additional tokens are permitted on the directive line after the optional *filename*.

**#endif**

Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**. No additional tokens are permitted on the directive line.

**#ifdef** *name*

The lines following will appear in the output if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**. No additional tokens are permitted on the directive line after *name*.

**#ifndef** *name*

The lines following will appear in the output if and only if *name* has not been the subject of a previous **#define**. No additional tokens are permitted on the directive line after *name*.

**#if** *constant-expression*

Lines following will appear in the output if and only if the *constant-expression* evaluates to nonzero. All binary nonassignment C operators, the ?: operator, the unary -, !, and ~ operators are all legal in *constant-expression*. The precedence of the operators is the same as defined by the C language. There is also a unary operator **defined**, which can be used in *constant-expression* in these two forms: **defined**

( *name* ) or **defined** *name*. This allows the utility of **#ifdef** and **#ifndef** in a **#if** directive. Only these operators, integer constants, and names which are known by *cpp* should be used in *constant-expression*. In particular, the **sizeof** operator is not available.

To test whether either of two symbols, *foo* and *fum*, are defined, use

```
#if defined(foo) || defined(fum)
```

**#elif** *constant-expression*

An arbitrary number of **#elif** directives is allowed between a **#if**, **#ifdef**, or **#ifndef** directive and a **#else** or **#endif** directive. The lines following the **#elif** directive will appear in the output if and only if the preceding test directive evaluates to zero, all intervening **#elif** directives evaluate to zero, and the *constant-expression* evaluates to nonzero. If *constant-expression* evaluates to nonzero, all succeeding **#elif** and **#else** directives will be ignored. Any *constant-expression* allowed in a **#if** directive is allowed in a **#elif** directive.

**#else** The lines following will appear in the output if and only if the preceding test directive evaluates to zero, and all intervening **#elif** directives evaluate to zero. No additional tokens are permitted on the directive line.

The test directives and the possible **#else** directives can be nested.

## FILES

<i>INCDIR</i>	standard directory list for <b>#include</b> files, usually <i>/usr/include</i>
<i>LIBDIR</i>	usually <i>/lib</i>

## SEE ALSO

*cc(1)*, *lint(1)*, *m4(1)*.

## DIAGNOSTICS

The error messages produced by *cpp* are intended to be self-explanatory. The line number and filename where the error occurred are printed along with the diagnostic.

## NOTES

The unsupported **-W** option enables the **#class** directive. If it encounters a **#class** directive, *cpp* will exit with code 27 after finishing all other processing. This option provides support for “C with classes.”

Because the standard directory for included files may be different in different environments, this form of **#include** directive:

```
#include <file.h>
```

should be used, rather than one with an absolute path, like:

```
#include "/usr/include/file.h"
```

*cpp* warns about the use of the absolute pathname.

—

—

—



**NAME**

crash - examine system images

**SYNOPSIS**

*/etc/crash* [ **-d** *dumpfile* ] [ **-n** *namelist* ] [ **-w** *outputfile* ]

**DESCRIPTION**

The *crash* command is used to examine the system memory image of a live or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments to *crash* are *dumpfile*, *namelist*, and *outputfile*.

*dumpfile* is the file containing the system memory image. The default *dumpfile* is */dev/kmem*. The system image can also be slice zero of the raw disk that contains the dump area (for example, */dev/rdisk/c0d0s0*); or it can be the pathname of a file produced using *dd* to copy slice zero or just the dump area; or in the case of a tape dump, the second file on the tape.

The unstripped executable file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is */etc/lddrv/unix.exec* if examining a running system or */etc/lddrv/prev.unix.exec* if examining a dump. If neither of these files exists, the default is */unix*. If a system image from another machine is to be examined, the corresponding *prev.unix.exec* must be copied from that machine. The *prev.unix.exec* is preferred to */unix* because it also contains the *namelist* for all the loaded drivers at the correct addresses.

When the *crash* command is invoked, a session is initiated. The output from a *crash* session is directed to *outputfile*. The default *outputfile* is the standard output.

Input during a *crash* session is of the form:

function [ argument ... ]

where *function* is one of the *crash* functions described in the FUNCTIONS section of this man page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system and the process that was running at the time of the crash for a crashed system. If the contents of a table are being dumped, the default is all active table entries.

The following function options are available to *crash* functions wherever they are semantically valid:

- e            Displays every entry in a table.
- f            Displays the full structure.
- p            Interprets all address arguments in the command line as *physical* addresses.
- s process    Specifies a process slot other than the default.
- w file        Redirects the output of a function to *file*.

Note that if the **-p** option is used, all address and symbol arguments explicitly entered on the command line will be interpreted as physical addresses. If they are not physical addresses, results will be inconsistent.

The functions *mode*, *defproc*, and *redirect* correspond to the function options **-p**, **-s**, and **-w**. The *mode* function may be used to set the address translation mode to physical or virtual for all subsequently entered functions; *defproc* sets the value of the process slot argument for subsequent functions; and *redirect* redirects all subsequent output.

Output from *crash* functions may be piped to another program in the following way:

```
function [ argument ... ] ! shell_command
```

For example,

```
mount ! grep rw
```

will write all mount table entries with an *rw* flag to the standard output. The redirection option (**-w**) cannot be used with this feature.

Depending on the context of the function, numeric arguments will be assumed to be in a specific radix. Counts are assumed to be decimal. Addresses are always hexadecimal. Table address arguments larger than the size of the function table will be interpreted as hexadecimal addresses; those smaller will be assumed to be decimal slots in the table. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal will be interpreted as hexadecimal if it is preceded by *0x* and as octal if it is preceded by *0*. Decimal override is designated by *0d*, and binary by *0b*.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as *p* for *proc*, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information will accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a-b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be +, -, \*, /, &, or |. An operand that is a number should be preceded by a radix prefix if it is not a decimal number (*0* for octal, *0x* for hexadecimal, and *0b* for binary). The expression must be enclosed in parentheses (). Other functions will accept any of these argument forms that are meaningful.

Two abbreviated arguments to *crash* functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

```
table_entry = table entry | address | symbol | range | expression
```

```
start_addr = address | symbol | expression
```

## FUNCTIONS

? [-w file]

List available functions.

!cmd Escapes to the shell to execute a command.

adv [-e] [-w file] [{-p] table\_entry ...]

Prints the advertise table.

base [-w file] number ...

Prints *number* in binary, octal, decimal, and hexadecimal. A number in a radix other than decimal should be preceded by a prefix that indicates its radix as follows: *0x*, hexadecimal; *0*, octal; and *0b*, binary.

buffer [-w file] [-format] bufferslot

or

buffer [-w file] [-format] [-p] start\_addr

Alias: **b**.

Prints the contents of a buffer in the designated format. The following format designations are recognized: **-b**, byte; **-c**, character; **-d**, decimal; **-x**, hexadecimal; **-o**, octal; **-r**, directory; and **-i**, inode. If no format is given, the previous format is used. The default format at the beginning of a *crash* session is hexadecimal.

**bufhdr** [-f] [-w file] [[-p] table\_entry ... ]  
 Alias: **buf**.  
 Prints system buffer headers.  
 The **-f** option produces different output depending on whether the buffer is local or remote (contains RFS data).

**callout** [-w file ]  
 Alias: **c**.  
 Prints the callout table.

**cblk** [-e] [-p] [-w file] [-t type] [ table\_entry ... ]  
 Displays contents of cblocks.

**cd**       Equivalent to  
           **tty -t cd**  
           (See **tty** function below.)

**clist** [-e] [-p] [-w file] [-t type] [ table\_entry ... ]  
 Displays usage of clists.

**conbuf** [-w file ]  
 Displays console buffer.

**dballoc** [-w file] [ class ... ]  
 Prints the **dballoc** table. If a class is entered, only data block allocation information for that class will be printed.

**dbfree** [-w file] [ class ... ]  
 Prints free streams data block headers. If a class is entered, only data block headers for the class specified will be printed.

**dblock** [-e] [-w file] [-c class ... ]  
 or  
**dblock** [-e] [-w file] [[-p] table\_entry ... ]  
 Prints allocated streams data block headers. If the class option (-c) is used, only data block headers for the class specified will be printed.

**defproc** [-w file] [-c ]  
 or

**defproc** [ -w file ] [ slot ]

Sets the value of the process slot argument. The process slot argument may be set to the current slot number (-c) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a *crash* session, the process slot is set to the current process.

**dis** [ -w file ] [ -a start\_addr [ count ]

Disassembles from the start address for *count* instructions. The default count is 1. The absolute option (-a) specifies a nonsymbolic disassembly.

**disk** [ -w file ]

Displays disk information.

**drvtable** [ -w file ]

Displays loadable driver table information.

**ds** [ -w file ] virtual\_address ...

Prints the data symbol whose address is closest to, but not greater than, the address entered.

**fcallout** [ -w file ]

Alias: **fc**.

Prints the fast callout table.

**file** [ -e ] [ -w file ] [ [-p] table\_entry ... ]

Alias: **f**.

Prints the file table.

**findaddr** [ -w file ] table slot

Prints the address of *slot* in *table*. Only tables available to the *size* function are available to *findaddr*.

**findslot** [ -w file ] virtual\_address ...

Prints the table, entry slot number, and offset for the address entered. Only tables available to the *size* function are available to *findslot*.

**fs** [ -w file ] [ [-p] table\_entry ... ]

Prints the file system information table.

**gdp** [ -e ] [ -f ] [ -w file ] [ [-p] table\_entry ... ]

Prints the gift descriptor protocol table.

**gt** Equivalent to

**tty -t gt**

(See *tty* function below.)

**help** [-w file] function ...

Prints a description of the named function, including syntax and aliases.

**inode** [-e] [-f] [-w file] [[-p] table\_entry ...]

Alias: **i**.

Prints the inode table, including file system switch information.

**iop16** Equivalent to

**tty -t iop16**

(See **tty** function below.)

**kfp** [-w file] [-s process] [-r]

or

**kfp** [-w file] [-s process] [value]

Prints the frame pointer for the start of a kernel stack trace. The **kfp** value can be set using the value argument or the reset option (**-r**), which sets the **kfp** from the saved **kfp** in a dump. If no argument is entered, the current value of the **kfp** is printed.

**lck** [-e] [-w file] [[-p] table\_entry ...]

Alias: **l**.

Prints record locking information. If the **-e** option is used or table address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to inodes is printed.

**linkblk** [-e] [-w file] [[-p] table\_entry ...]

Prints the linkblk table.

**major** [-w file] [entry ...]

Prints the MAJOR table.

**map** [-w file] mapname ...

Prints the map structure of the given mapname.

**mbfree** [-w file]

Prints free streams message block headers.

**mblock** [-e] [-w filename] [[-p] table\_entry ...]

Prints allocated streams message block headers.

**mode** [-w file] [mode]

Sets address translation of arguments to virtual (**v**) or physical (**p**) mode. If no mode argument is given, the current mode is printed. At the start of a *crash* session, the mode is virtual.

**mount** [-e] [-w file] [[-p] table\_entry ...]

Alias: **m**.

Prints the mount table.

**msg** [-e] [-f] [-p] [-w file] [-s process] [table\_entry ...]

Displays IPC message queue headers.

**msginfo** [-p] [-w file]

Displays IPC message information.

**msgtext** [-e] [-p] [-w file] [-s process] [table\_entry ...]

Displays IPC message data.

**nm** [-w file] symbol ...

Prints value and type for the given symbol.

**notify** [-e] [-p] [-w file] symbols

**od** [-p] [-w file] [-format] [-mode] [-s process] start\_addr [count]

Alias: **rd**.

Prints *count* values starting at the start address in one of the following formats: character (-c), decimal (-d), hexadecimal (-x), octal (-o), ASCII (-a), or hexadecimal/character (-h), and one of the following modes: long (-l), short (-t), or byte (-b). The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format -h prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a *crash* session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.

**pd** [-e] [-w file] [-s process] section segment

or

**pd** [-e] [-w file] [-s process] [-p] start\_addr [count]

*S/640 Only:*

The page descriptor table of the designated memory *section* and *segment* is printed. Alternatively, the page descriptor table starting at the start address for *count* entries is printed. If no count is entered, 1 is assumed.

**pfdat** [-e] [-w file] [[-p] table\_entry ...]

Prints the pfdata table.

**pfree** [-e] [-p] [-w file] table\_entry ...

Displays free list entries.

**phash** [-e] [-p] [-w file]  
 Displays page hash table.

**proc** [-e] [-f] [-w file] [[-p] table\_entry ... #procid ...]  
 or  
**proc** [-f] [-w file] [-r]  
 Alias: **p**.  
 Prints the process table. Process table information may be specified in two ways. First, any mixture of table entries and process ids may be entered. Each process id must be preceded by a #. Alternatively, process table information for runnable processes may be specified with the runnable option (-r).

**pt** Equivalent to  
**tty -t pt**  
 (See **tty** function below.)

**qrun** [-w file]  
 Prints the list of scheduled streams queues.

**queue** [-e] [-w file] [[-p] table\_entry ...]  
 Prints streams queues.

**quit** Alias: **q**.  
 Terminates the *crash* session.

**rcvd** [-e] [-f] [-w file] [[-p] table\_entry ...]  
 Prints the receive descriptor table.

**redirect** [-w file] [-c]  
 or  
**redirect** [-w file] [file]  
 Used with a filename, redirects output of a *crash* session to the named file. If no argument is given, the filename to which output is being redirected is printed. Alternatively, the close option (-c) closes the previously set file and redirects output to the standard output.

**region** [-e] [-f] [-w file] [[-p] table\_entry ...]  
 Prints the region table.

**scsi** [-w file]  
 Displays SCSI tables.

**scsirqb** [-f] [-w file] [tbl\_entry | start\_addr]  
 Displays SCSI request blocks.



**sdt** [-e] [-w file] [-s process] section

or

**sdt** [-e] [-w file] [-s process] [-p] start\_addr [count]

*S/640 Only:*

The segment descriptor table for the named memory section is printed. Alternatively, the segment descriptor table starting at start address for *count* entries is printed. If no count is given, a count of 1 is assumed.

**search** [-p] [-w file] [-m mask] [-s process] pattern start\_addr length

Prints the words in memory that match *pattern*, beginning at the start address for *length* words. The mask is anded (&) with each memory word and the result compared against the pattern. The mask defaults to 0xffffffff.

**ser** Equivalent to

**tty -t ser**

(See **tty** function below.)

**shm** [-e] [-f] [-p] [-w file] table\_entry ...

Displays IPC shared memory headers.

**shminfo** [-p] [-w file]

Displays system IPC shared memory information.

**size** [-w file] [-x] [structure\_name ...]

Prints the size of the designated structure. The (-x) option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

**sndd** [-e] [-f] [-w file] [[-p] table\_entry ...]

Prints the send descriptor table.

**sptb** [-e] [-p] [-w file] [start\_addr]

Displays sptballoc maps.

**srmount** [-e] [-w file] [[-p] table\_entry ...]

Prints the server mount table.

**stack** [-w file] [-u] [process]

or

**stack** [-w file] [-k] [process]

or

**stack** [ **-w** file ] [ [ **-p** ] **-i** start\_addr ]

Alias: **s**.

Dumps stack. The (**-u**) option prints the user stack. The (**-k**) option prints the kernel stack. The (**-i**) option prints the interrupt stack starting at the start address. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

**stat** [ **-w** file ]

Prints system statistics.

**stream** [ **-e** ] [ **-f** ] [ **-w** file ] [ [ **-p** ] table\_entry ... ]

Prints the streams table.

**strstat** [ **-w** file ]

Prints streams statistics.

**swap** Displays swap map statistics.

**swapinfo**

Displays swap statistics.

**trace** [ **-w** file ] [ **-r** ] [ process ]

or

**trace** [ **-w** file ] [ [ **-p** ] **-i** start\_addr ]

Alias: **t**.

Prints stack trace. The kfp value is used with the **-r** option. The interrupt option prints a trace of the interrupt stack beginning at the start address. The interrupt stack trace and the stack trace for the current process are not available on a running system.

**ts** [ **-w** file ] virtual\_address ...

Prints closest text symbol to the designated address.

**tty** [ **-e** ] [ **-f** ] [ **-w** file ] [ **-t** type [ [ **-p** ] table\_entry ... ] ]

or

**tty** [ **-e** ] [ **-f** ] [ **-w** file ] [ [ **-p** ] start\_addr ]

Valid types: **cd**, **gt**, **iop16**, **pt**, **ser**, **vt**, **vtd**, **wxt**.

Prints the tty table. If no arguments are given, the tty table for all tty types is printed. If the **-t** option is used, the table for the single tty type specified is printed. If no argument follows the type option, all entries in the table are printed. A single tty entry may be specified from the start address.

**unnotify** [ -e ] [ -p ] [ -w file ] [ -s process ] symbols  
 Displays queued notifications for process.

**user** [ -f ] [ -w file ] [ process ]  
 Alias: **u**.  
 Prints the ublock for the designated process.

**var** [ -w file ]  
 Alias: **v**.  
 Prints the tunable system parameters.

**vt** Equivalent to  
**tty -t vt**  
 (See **tty** function above.)

**vtd** Equivalent to  
**tty -t vtd**  
 (See **tty** function above.)

**vtop** [ -w file ] [ -s process ] start\_addr ...  
 Prints the physical address translation of the virtual start address.

**wxt** Equivalent to  
**tty -t wxt**  
 (See **tty** function.)

## FILES

/dev/kmem system image of currently running system  
 /dev/rdisk/c?d?s0 used to access system image on disk

—

—

—

**NAME**

createdev - create device nodes for assorted device types

**SYNOPSIS**

```
createdev -d device [ -c controller ] [ -v ] [ -r ] [ -p ] [ -t ]
```

**DESCRIPTION**

The *createdev* command is used to create device nodes of various types. After parsing various parameters, the command invokes *mknod* to create the specified device node or sets of device nodes.

The **-d** option specifies the device number (for example, unit, drive, or line number), and is required for every invocation.

The **-c** option specifies the controller number of the specified device.

The **-v** option specifies that disk devices are to be created. Both block-type and character-type device nodes of the form */dev/rdsk/cxdysz* and */dev/dsk/cxdysz* are added. Each invocation creates as many slices as the disk supports on CTIX (currently 16). For the **-v** option, both the **-d** and **-c** options are required.

The **-r** option specifies that devices are created to provide access to streaming tape drives; these are the character-type device nodes. These devices are of the form: */dev/rmt/cxdy*, */dev/rmt/cxdyc*, */dev/rmt/cxdyh*, */dev/rmt/cxdyhn*, */dev/rmt/cxdyl*, */dev/rmt/cxdyln*, */dev/rmt/cxdym*, */dev/rmt/cxdymn*, and */dev/rmt/cxdyn*.

This option is useful when adding SCSI tape devices. For the **-r** option, both the **-d** and **-c** options are required.

For 68040-based machines, the **-t** option allows devices to be created of the type */dev/ttyaxx*, where *a* is an alphabetic character from a-z that signifies the expansion slot in which the TTY port is located, and *xx* is a two-digit decimal that signifies the TTY port number. For 68020-based machines, the **-t** option allows devices to be created of the type */dev/ttyxxx*, where *xxx* is a three-digit decimal that signifies the TTY port number. For both types of machines, these are character-type devices, typically used for terminals and line printers, as well as other peripherals. For this option, the **-d** option is required.

The **-p** option allows devices to be created of the type */dev/ttypxx*. These are character-type devices, typically used for virtual login sessions. An example of this is an Ethernet connection. The **-d** option is required.

**FILES**

/dev/tty\*  
/dev/tty\*  
/dev/dsk/\*  
/dev/rdisk/\*  
/dev/rmt/\*

**SEE ALSO**

mknod(1M).

**NAME**

*fsck*, *dfck* - check and repair file systems

**SYNOPSIS**

```
/etc/fsck [ -y ] [ -n ] [ -sc:s ] [ -s ] [ -Sc:s ] [ -S ] [ -t file ] [ -q ] [ -D ]
[ -f ] [ -p ] [ -bB ] [ -O ] [ -M ] [ -E ] [ file-systems ]
/etc/dfck [ options1 ] filsys1 ... - [ options2 ] filsys2 ...
```

**DESCRIPTION****Fck**

*fsck* audits and interactively repairs inconsistent conditions for CTIX file systems. If the file system is consistent, the number of files, number of blocks used, and number of free blocks are reported. If the file system is inconsistent, the operator is prompted for concurrence before each correction is attempted. It should be noted that some corrective actions result in some loss of data: the amount and severity of data lost can be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond **yes** or **no**. If the operator does not have write permission, *fsck* defaults to an **-n** action. Upon completion, *fsck* reports the number of used and free 512-byte blocks and the number of files in the file system.

Modifying a mounted (**root**) file system requires special precautions by *fsck*, because a single *sync* (2) undoes all of *fsck*'s repair work. To prevent this, *fsck* performs a *uadmin*(A\_REMOUNT,0,0) [see *uadmin*(2)]. The system call forces CTIX to reread the super-block from the disk. If there is extensive damage to the mounted file system, *fsck* reboots CTIX.

The following options are interpreted by *fsck*:

- y** Assumes a yes response to all *fsck* prompts.
- n** Assumes a no response to all questions asked by *fsck* prompts; does not open the file system for writing.
- sc:s**
- s** Ignores the actual free list or bit map and (unconditionally) reconstructs a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent.

If *c:s* is given on a standard file system, the free list is organized with *c* blocks-per-cylinder and *s* blocks skipped. If *c:s* is omitted, the values originally specified to *mkfs* are used. If these values were not specified, the value *400:7* is used.

- Sc:s**
- S** Conditionally reconstructs the free list or bit map. This option is like **-s**, described above, except that the free list or bit map is rebuilt only if no discrepancies are discovered in the file system. **-S** forces a no response to *fsck* prompts. This option is useful for forcing free list or bit map reorganization on uncontaminated file systems.
- t** If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the **-t** option is specified, the file named in the next argument is used as the scratch file, if needed. Without the **-t** flag, *fsck* prompts for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes.
- q** Quiets *fsck*. Does not print size-check messages in Phase 1. Unreferenced FIFOs are silently removed. If *fsck* requires it, counts in the super-block are automatically fixed and the free list or bit map is salvaged.
- D** Checks directories for consistency. This is useful after system crashes. The following inconsistencies are sought:
- Entries with null names but nonzero i-numbers.
  - Entries that are not padded to full size with nulls.
  - Invalid . and .. entries.
  - Names that contain a slash (/).
  - Final blocks that are not cleared past end-of-file.
- f** Fast checks. Checks block and sizes (Phase 1) and checks the free list or bit map (Phase 5). The free list or bit map is reconstructed (Phase 6) if it is necessary.
- p** Preens file systems only; intended for autoboot. The *fsck* program does not prompt for operator input; instead, it applies standard fixes whenever the fix doesn't involve loss of data. Only the following problems are subject to this kind of fix:
- Unreferenced i-nodes.
  - Link counts in i-nodes too large.
  - Missing blocks in the free list.
  - Blocks in the free list also in files.
  - Counts in the super-block wrong.



Any problem not of this type causes *fsck* to terminate with an error status. The startup script that runs *fsck* (normally */etc/bcheckrc*) can specify the **-p** option to *fsck* and make a normal boot contingent upon a normal *fsck* return status.

**-b or -B**

If the file system being checked is the **root** file system and modifications have been made, this resyncs the file system, or reboots if necessary.

**-E** Converts file system to extended bit-map format.

**-M** Converts file system to new bit-map free list format.

**-O** Converts file system to old free list format.

The **-E**, **-M**, and **-O** options imply **-s**.

If no *file-systems* are specified, *fsck* reads a list of default file systems from the file */etc/checklist*.

Inconsistencies are checked as follows:

1. Blocks claimed by more than one i-node or the free list.
2. Blocks claimed by an i-node or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
  - Incorrect number of blocks.
  - Directory size not 16-byte aligned.
5. Bad i-node format.
6. Blocks not accounted for anywhere.
7. Directory checks:
  - File pointing to unallocated i-node.
  - I-node number out of range.
8. Super-block checks:
  - More than 65536 i-nodes.
  - More blocks for i-nodes than exist in the file system.
9. Bad free block list format.
10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory, if the files are nonempty. The user is notified if the file or directory is empty or not. If it is empty, *fsck* silently removes them. The *fsck* program forces the reconnection of nonempty directories. The name assigned is the i-node number. The only restriction is that the directory **lost+found** must

preexist in the root of the file system being checked and must have empty slots in which entries can be made. The **lost+found** directory is normally created by running *mklost+found*(1M) just after the file system is created with *mkfs*(1M).

Checking the raw device is almost always faster and should be used with everything but the **root** file system.

### Dfscck

The *dfscck* program allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A dash (-) is the separator between the file system groups.

The *dfscck* program permits an operator to interact with two *fsck*(1M) programs at once. To aid in this, *dfscck* prints the file system name for each message to the operator. When answering a question from *dfscck*, the operator must prefix the response with a 1 or a 2 (indicating that the answer refers to the first or second file system group).

Do not use *dfscck* to check the **root** file system.

### FILES

<i>/etc/checklist</i>	default list of file systems to check
<i>/etc/checkall</i>	optimizing <i>dfscck</i> shell file

### SEE ALSO

*cli*(1M), *init*(1M), *mklost+found*(1M), *uadmin*(2), *ncheck*(1M), *checklist*(4), *fs*(4).  
*S/Series CTIX Administrator's Guide*.

### DIAGNOSTICS

The diagnostics produced by *fsck* are intended to be self-explanatory.

If **-p** was specified and preening was inadequate, a nonzero status is returned.

### NOTES

Always unmount file systems before running *fsck* except in the case of the **root** file system.

The *block* device must be used with mounted file systems; thus, the **root** file system must always be specified as the block device.

The maintenance tape can be used to check the normal **root** file system as a raw device, unmounted. (In this case, the **root** file system is on the RAM disk.)

The *fsck* program determines the file system type (1K or 4K) on its own.

## BUGS

I-node numbers for . and .. in each directory should be checked for validity.

The *fscck* program does not know how to create a **lost+found** directory.

—

—

—

**NAME**

*fstyp* - determine file system identifier

**SYNOPSIS**

*fstyp special*

**DESCRIPTION**

The *fstyp* command allows the user to determine the file system identifier of mounted or unmounted file systems using heuristic programs. The file system type is required by *mount(2)* and sometimes by *mount(1M)* to mount file systems of different types.

The directory */etc/fstyp.d* contains a program for each file system type to be checked; each program applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file-system identifier for that type and exits with a return code of 0; otherwise, it prints error messages on standard error and exits with a nonzero return code. The *fstyp* command runs the programs in */etc/fstyp.d* in alphabetical order, passing *special* as an argument; if any program succeeds, its file-system type identifier is printed and *fstyp* exits immediately. If no program succeeds, *fstyp* prints "Unknown\_fstyp" to indicate failure.

**NOTE**

*fstyp* reports "S51K" for both 1K and 4K System V file systems.

**WARNING**

The use of heuristics implies that the result of *fstyp* is not guaranteed to be accurate.

**SEE ALSO**

*mount(1M)*, *mount(2)*, *sysfs(2)*.

—

—

—

## NAME

hinv - hardware inventory

## SYNOPSIS

*/etc/hinv* option

*/etc/hinv* hardware-item

## DESCRIPTION

The *hinv* command provides hardware configuration information. There are two forms of the command. In the first form, an *option* is given and the result is printed on *stdout*; in the second form, a particular hardware item is specified, and *hinv* exits with **0** if it exists, or with **1** otherwise.

*Option* is one of the following:

- p** Prints hardware configuration. Items are printed one per line.
- c** Prints CPU type (68020 or 68040).
- f** Prints FPU type (68040 or 68881). (For S/280 only: 68882 is reported as 68881.)
- s** Prints system type.
- u** Returns a meaningless value of 128; included for compatibility only.
- m** Prints total physical memory in bytes.

*Hardware-item* is one of the following:

- 68881** 68881 (on S/80, S/480, or S/640) or 68882 (on S/280) floating-point processor is present.
- iop** Terminal Accelerator Board is present.
- 422** Any RS-422 Cluster Board.
- 422-2** Two-channel RS-422 Cluster Board.
- 422-4** Four-channel RS-422 Cluster Board.
- vme** VME Interface Board.
- sn** RS-232 Board *n*.
- scsi** A SCSI interface is present.
- S0** Onboard SCSI is present.
- Sn** SCSI Combo Board *n*.

<b>ipt</b>	Interphase Tape Controller is in EEPROM.
<b>smd</b>	Interphase SMD Controller is in EEPROM.
<b>mpcc</b>	Multiprotocol Communications Controller is in EEPROM.
<b>serial</b>	Gives number of serial ports present.
<b>disks</b>	Gives number of disks present.
<b>eprom</b>	VME EEPROM valid for UNIX.
<b>enet</b>	Ethernet Combo Board is present or a CMC Ethernet Board is in EEPROM.
<b>cmcenp</b>	CMC Ethernet Board is in EEPROM.
<b>En</b>	Ethernet Combo Board <i>n</i> .
<b>In</b>	IOP16 Board <i>n</i> is present ( $n = 1 \dots 4$ ).
<b>hwfloat</b>	Hardware floating point is present.

**BUGS**

The *hinv* command does not know about particular VME cards, which can be plugged in when the VME Interface Board is present.



**NAME**

includes - determine C language preprocessor include files

**SYNOPSIS**

**includes** [ option ... ] file ...

**DESCRIPTION**

The *includes* command determines the **#include** files necessary to compile a C language source file using the C language preprocessor *cpp*(1); the command is based on *cpp*(1) and takes the same options. (If you are running a 6.3 or later release of CTIX, *cpp* is a built-in feature of the C compiler.) Multiple source files can be named on the command line. However, instead of producing preprocessed code, *includes* produces on standard output a list of the **#include** file dependencies (directly or nested) of the named source files.

The output format is suitable for direct use in a *makefile* to be used by the *make*(1) command. For each named source file, the **#include** files are listed, one per line, preceded by the name of the source file (with the last letter of its name changed to the letter **o**). The two names are separated by a colon and a space : . For example, if source file **pgm.c** depends only on the **#include** file **incl.h**, the output of **includes** for the source file **pgm.c** would be

```
pgm.o: incl.h
```

The following *options* to *includes* are recognized:

**-Uname** Removes any initial definition of *name*, where *name* is a reserved symbol that is predefined by the particular preprocessor. The current list of possibly-reserved symbols includes the following:

```
operating system:  ibm, gcos, os, tss, unix
hardware:          interdata, mc68k, mc68k32, m68k,
                  mc68000, mc68010, mc68020 mc68881,
                  mc68040, pdp11, u370, u3b, vax
system variants:  RES, RT
```

**-Dname**

**-Dname=def** Defines *name* with value *def* as if by a **#define**. If no *=def* is given, *name* is defined with value 1. The **-D** option has lower precedence than the **-U** option. That is, if the same name is used in both a **-U** option and a **-D** option, the name is undefined, regardless of the order of the options.

- T           The **-T** option forces *includes* to use only the first eight characters to distinguish preprocessor symbols and is included for backward compatibility.
- Idir        Changes the algorithm for searching for **#include** files whose names do not begin with / to look in *dir* before looking in the directories on the standard list. Thus, **#include** files with names enclosed in double quotation marks (“”) are searched for first in the directory of the file with the **#include** line, then in directories named in **-I** options, and last in directories on a standard list. For **#include** files with names enclosed in angle brackets (<>), the directory of the file with the **#include** line is not searched. By default, *includes* searches for the name enclosed in angle brackets in */usr/include*; however, if the shell variable INCROOT is set, *includes* prepends the value of INCROOT to the standard list; this is particularly useful for cross-compilation.
- Ydir        Uses directory *dir* in place of the standard list of directories when searching for **#include** files. The **-Y** option overrides the value for INCROOT if it is set.
- H           Prints, one per line on standard error, the pathnames of included files.

Two special names are understood by *includes*: `__LINE__` is defined as the current line number (as a decimal integer) as known by *includes*, and `__FILE__` is defined as the current filename (as a C string) as known by *includes*. The special names can be used anywhere (including in macros), just as any other defined name.

All *cpp* directives understood by *includes* start with lines begun by #. The directives are as follows:

**#define name token-string**

Replaces subsequent instances of *name* with *token-string*.

**#define name( arg, ..., arg ) token-string**

Notice that there can be no space between the *name* and the (. Replace subsequent instances of *name* followed by a (, a list of comma-separated tokens, and a ) by *token-string* where each occurrence of an *arg* in the *token-string* is replaced by the corresponding token in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded *token-string* unchanged. After the entire *token-string*

has been expanded, *includes* restarts its scan for names to expand at the beginning of the newly created *token-string*.

**#ident** "*string*"

This directive has no effect.

**#undef** *name* Causes the definition of *name* (if any) to be forgotten from now on.

**#include** "*filename*"

**#include** <*filename*>

Includes at this point the contents of *filename* (which is then run through *includes*). When the <*filename*> notation is used, *filename* is searched for only in the standard places; see the descriptions for the **-I** and **-Y** options for more detail.

**#line** *integer-constant* "*filename*"

This directive has no effect.

**#endif** Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**.

**#ifdef** *name* The lines following are processed if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

**#ifndef** *name* The lines following are not processed if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

**#if** *constant-expression*

Lines following are processed if and only if the *constant-expression* evaluates to nonzero. All binary nonassignment C operators, the ?: operator, the unary -, !, and ~ operators are all legal in *constant-expression*. The precedence of the operators is the same as defined by the C language. There is also a unary operator **defined**, which can be used in *constant-expression* in these two forms: **defined** ( *name* ) or **defined** *name*. This allows the utility of **#ifdef** and **#ifndef** in a **#if** directive. Only these operators, integer constants, and names that are known by *includes* should be used in *constant-expression*. In particular, the **sizeof** operator is not available.

**#elif** *constant-expression*

An arbitrary number of **#elif** directives is allowed between a **#if**, **#ifdef**, or **#ifndef** directive and a **#else** or **#endif** directive. The lines following the **#elif** directive will appear in the output if and

only if the preceding test directive evaluates to zero, all intervening **#elif** directives evaluate to zero, and the *constant-expression* evaluates to nonzero. If *constant-expression* evaluates to nonzero, all succeeding **#elif** and **#else** directives will be ignored. Any *constant-expression* allowed in a **#if** directive is allowed in a **#elif** directive.

**#else** The lines following will appear in the output if and only if the preceding test directive evaluates to zero, and all intervening **#elif** directives evaluate to zero.

The test directives and the possible **#else** directives can be nested.

## FILES

<i>INCDIR</i>	standard directory list for <b>#include</b> files, usually /usr/include
<i>LIBDIR</i>	usually /lib

## SEE ALSO

cc(1), cpp(1), m4(1).

## DIAGNOSTICS

The error messages produced by *includes* are intended to be self-explanatory. The line number and filename where the error occurred are printed along with the diagnostics.

**NAME**

*iopdump* - upload a Front-end I/O Processor's RAM

**SYOPSIS**

`/usr/local/bin/iopdump [ -p ] [ -i iop16number ] address length`

**DESCRIPTION**

*iopdump* uploads and displays **length** number of memory data bytes beginning at **address** from a front-end I/O processor.

The **address** argument is a hexadecimal value from 0 to ffff.

The **length** argument is a decimal value.

The default front-end processor is an IOP.

The **-i** option specifies that the type of front-end I/O processor is an IOP16. For the **-i** option, the number **iop16number** must be a decimal number in the range 0 to 3. There is no default number.

The **-p** option causes the retrieved data to be printed as an ASCII hexadecimal dump to the standard output. Without this option, the binary data is sent to the standard output.

**BUGS**

For the IOP16, *iopdump* obtains only the data from the first board.

—

—

—

**NAME**

`iv` - initialize and maintain volume

**SYNOPSIS**

`iv -iuostdlvq special [ descriptionfile ]`

**DESCRIPTION**

The `iv` command initializes and maintains a disk volume. *Special* and *descriptionfile*, described below, specify the disk and a description file for the disk volume. The `iv` command performs one of five operations, specified by the following options:

- i Completely initializes a volume. This consists of five phases:
  1. Initializes *iv*'s internal Volume Home Block, based on *descriptionfile* and the disk type. If the disk can support bad block handling, this creates an internal Bad Block Table. Puts bad block data from *descriptionfile* and volume's existing Bad Block Table (if any) in internal Bad Block Table.
  2. Formats medium.
  3. Performs a surface check. If the disk can support bad block handling, this adds bad blocks to the Bad Block Table. If the disk cannot support bad block handling, the first bad spot causes the disk to be rejected.
  4. Writes out the Volume Home Block. This has the effect of dividing the volume into slices (partitions).
  5. Allocates and writes out the files that share the Reserved Area (slice 0) with the Volume Home Block. If the disk can support bad block handling, one of these files is the Bad Block Table. Other files are specified in *descriptionfile*.
- u Updates the Volume Home Block. This is the same as -i, except that the second and third phases (medium formatting and surface check) are skipped.
- o Outputs a Volume Home Block and partition 0 to any file; requires a *descriptionfile*. The following command produces a dump tape:
 

```
iv -o /dev/rmt0 /usr/lib/iv/desc.tdump
```
- s Performs a surface test. Any bad blocks discovered are added to the Bad Block Table.

- t Tells volume description. Displays volume home block in human-readable form. No description file is needed. The volume's contents are not affected.
- d Displays description file. A description file that describes the current state of the volume is written to the standard output. If the Reserved Area contains a loader, the **loader** keyword's value is written as **/usr/lib/iv/loader**. If the Reserved Area contains a download image area, the Download Area Description lists files whose names are of the following form:

**/usr/lib/iv/wsxxx.yyy**

where *xxx* is the numeric device identification; *yyy* is **422** if *xxx* is even and **232** if *xxx* is odd.

The **-f** option, equivalent to **-u**, is provided for compatibility with older versions of *iv*. It should not be used, as it may disappear in future releases.

In addition to the single operation option (**-i**, **-u**, **-s**, **-t**, or **-d**) you can specify any or all of the following options:

- v Verbose display output. If the display includes the Volume Home Block, it also includes the Bad Block Table.
- l A normal surface test consists of a single pass over the disk; **-l** specifies ten passes.
- w A normal surface test pass consists of a read pass; **-w** specifies a write pass before each read pass.
- q Prints the size of the disk (in megabytes).

### File Parameters

*Special* is the character special file for slice zero on the drive. This name takes the form **/dev/rdisk/cndts0**, where *n* is the controller number and *t* is the drive number.

*Descriptionfile* is a text file that describes the volume. It is required by the **-i** and **-u** options. The description file consists of five parts:

- General Description
- Reserved Area Description
- Bad Blocks Description
- Partition Table Description
- Download Area Description



Each description is separated from the next by a line that contains only a single dollar sign (\$). Specifics for each of the five descriptions are given under separate headings below.

### General Description

Each line in the General Description begins with a keyword. Some keywords are followed by values; the value is separated from the keyword by spaces or tabs. For example:

```
ecc
cylinders      1024
```

Each keyword is used only once; valid keywords follow:

<b>type</b>	Mandatory unless the volume is already initialized in the appropriate format. Value is disk type: <b>HD</b> for onboard ST506 hard disk; <b>RD</b> for RAM disk; <b>V3200</b> for SMD controller; <b>SCSI</b> for SCSI disk; and <b>FD</b> for floppy disk.
<b>name</b>	Mandatory unless the volume is already initialized in the appropriate format. Value is the volume name. Any characters except spaces or tabs are permitted in the volume name; the serial number of the disk is the recommended volume name. The actual name in the Volume Home Block is always exactly six characters; <i>iv</i> right truncates names that are too long and right pads with nulls names that are too short.
<b>cylinders</b>	Mandatory unless the volume is already initialized in the appropriate format. Value is the number of cylinders on the disk. For SCSI, cylinders X heads X sectors should be just less than twice the number of 1K logical blocks on the disk.
<b>heads</b>	Mandatory unless the volume is already initialized in the appropriate format. Value is the number of heads on the disk.
<b>sectors</b>	Mandatory unless the volume is already initialized in the appropriate format. Value is the number of physical sectors per track. The standard value is 32.
<b>lsectors</b>	The number of logical 512-byte sectors on a SCSI disk. If this value is not supplied, the total number of available logical sectors on the device is used.
<b>formatextra</b>	The SMD drive is formatted with an extra sector on each track. (This sector is ignored by CTIX but is required for some disk drives, notably the Eagle-XP.)

- steprate** Mandatory for ST506 unless the volume is already initialized in the appropriate format. Value is a number that is passed to the disk controller. The normal steprate for ST506 drives is 14; 0 can be used for slower drives. See the disk manufacturer's documentation for further information.
- exchangeable** If this keyword is present, the disk can be removed from its drive.
- hitech** (ST506 drives only) If this keyword is present, the reduced write current line to the disk is used for head-select bit 3 to allow more than eight heads.
- precomp** (ST506 drives only) The value is  $c/16$ , where  $c$  is the cylinder at which precompensation should start. See the disk manufacturer's documentation for further information.
- ecc** The disk has been prepared to function in ECC mode.
- gap1**  
**gap2** Gap size for SMD drives. See the disk manufacturer's documentation for further information.
- unformattedbytes**  
The number of unformatted bytes per sector. This value is required if the "cyl head offset" format is used for the bad block table entries.

### Reserved Area Description

The Reserved Area Description describes the files that share slice zero with the Volume Home Block. Each line in the Reserved Area Description consists of a keyword followed by one or more parameters; one or more tabs or spaces separate keywords and parameters from each other. Here are the valid keywords and their meanings. (A logical block is 1024 bytes long.)

- loader** Describes the loader area. The first (mandatory) parameter is the full pathname of an a.out file to put in the loader area. The second (optional) parameter is the size of the loader area in logical blocks. If the second parameter is missing, the size of the a.out file is used. The standard value is 128.
- badblocktable** Describes the Bad Block Table. The first (mandatory) parameter is the size of the Bad Block Table in logical blocks. The second (optional) parameter is only used when an existing Bad Block Table contains errors; this parameter is *empty* to clear the Bad Block Table, missing otherwise.

**dump** Describes area to contain dump after crash. The only (mandatory) parameter specifies the size of the dump area in logical blocks.

**downloadarea** Describes area to contain system images for downloading. The only (mandatory) parameter specifies the size of the download area in logical blocks. (The files actually put in this area are described separately. See the Download Area Description heading below.)

All lines valid for the Reserved Area Description are optional. However, the Bad Block Table is mandatory on a volume that supports bad block handling (other than SCSI); the loader area is mandatory on a volume that is to hold an operating system.

### Bad Block Description

The Bad Block Description explicitly specifies up to 889 bad blocks to be added to the Bad Block Table. The *iv* command merges specified bad block information with information already in the Bad Block Table (if there already is one) and bad block information discovered through the surface test.

Each bad block entry is a single line. There are three forms:

*sector*

where *sector* is a physical sector number;

*cylinder head offset*

where *cylinder* is a cylinder number, *head* is a head number, and *offset* is the byte offset of the bad spot;

*cylinder head sector*

where *cylinder* is a cylinder number, *head* is a head number, and *sector* is a physical sector number of the bad spot. The third form is selected by placing the keyword **sector** on the line preceding the first entry of this type in the bad block description. Entries using the third form must come last in the list of bad blocks. All three forms condemn a single sector; the second form condemns the sector that contains the specified byte.

The last sector on each track serves as a bad block alternate. *iv* chooses the alternates in a way that minimizes extra seeking for alternate blocks.

### Partition Table Description

The Partition Table Description specifies where the slices (partitions) on the disk are to begin and end. Each line in the description specifies the starting logical block of a slice. Start blocks must be on track boundaries, except in the

case of SCSI drives, where start blocks need only be on a logical block boundary.

Except for overlapping partitions, slices must be listed in ascending numeric order; the beginning of a slice defines the end of the previous slice.

It is possible to specify overlapping partitions, although care must be taken in doing so. A \$ following any block number indicates that the slice extends to the end of the disk, beyond the next boundary number. Any slice with a starting block number that is larger than its successor must extend to the end of the disk (and must therefore be followed by the \$ parameter).

For example, the following description specifies five slices; the fifth slice extends from the second slice to the end of the disk:

```
0
16
20016
40016 $
16 $
```

The first logical block boundary number in the description must always be 0. The last slice in the description always extends to the end of the disk (\$ is optional).

There can be at most 16 slices on a disk.

It is a fatal error to specify a slice 1 that does not leave enough room in slice 0 for the Volume Home Block and the slice 0 files.

### Download Area Description

The Download Area Description specifies system images to be included in the Download Area. Each line in the description consists of a numeric device identification and the full pathname of the file to be copied into the download area; the two parts of the line are separated by one or more spaces or tabs.

### EXAMPLES

The following example shows a disk description file for a nonbootable disk (bad blocks expressed in "cylinder/head/sector" format):

```
# MAXTOR 85 MB disk
type          HD
name          Serno
cylinders     1024
heads         8
sectors       17
steprate      14
hitech
```

```

ecc
$
badblocktable 1
$
sector
15      5      4
$
0
8
$
$

```

The following file describes a bootable SMD (bad blocks expressed in "cylinder/head/offset" format):

```

type          V3200
name          Serno
cylinders     1489
heads        11
sectors       33
ecc
gap1          16
gap2          16
unformattedbytes 620
$
badblocktable 3
dump          1024
downloadarea  300
loader        /usr/lib/iv/loader 128
$
12      3      405
187     9      1010
692     4      5228
66      2      657
985     5      3398
$
0
1456
17360
25360

```

```

45360
85360
125360
165360 $
$
100 /usr/lib/iv/ws100.422
200 /usr/lib/iv/ws200.422
$

```

The following file describes a bootable Hitachi drive (bad blocks expressed as physical sector numbers):

```

type          HD
name          Serno
cylinders     823
heads        10
sectors       17
steprate      14
hitech
ecc
$
badblocktable 1
dump          1024
downloadarea  300
loader        /usr/lib/iv/loader 128
$
1048
2441
5064
15119
15678
23533
23534
42091
43918
60466
60467
$
0
1456

```

```

17730
25922
46402  $
$
100   /usr/lib/iv/ws100.422
200   /usr/lib/iv/ws200.422
$

```

The following file describes a drive without a dump area (no bad blocks specified):

```

type           HD
name           Serno
cylinders      645
heads         7
sectors       17
steprate      14
precomp       80
hitech
ecc
$
badblocktable 1
downloadarea  300
loader        /usr/lib/iv/loader 128
$
$
0
432
12328
18328  $
$
100   /usr/lib/iv/ws100.422
200   /usr/lib/iv/ws200.422
$

```

#### FILES

```

/dev/rdisk/*      disk character special files
/usr/lib/iv/desc.*  prototype description files

```

**SEE ALSO**

disk(7).

*S/Series CTIX Administrator's Guide.*

“WD2010-05 Winchester Disk Controller” in *Storage Management Products Handbook*. Irvine, Calif.: Western Digital Corp., 1984.

**NOTES**

Any device in physical mode (for example, while surface testing or formatting is being done) is an exclusive open device: use the maintenance tape to reformat or run surface tests on the boot device.

A typical disk has fewer bad spots than the total number of megabytes (a 40-MB drive should have fewer than 40 bad spots, and so forth).

**WARNINGS**

The **-i**, **-u**, and **-s** operations are dangerous or fatal to existing volume data. Always precede these operations with a backup.

When a new bad block is itself an alternate block, *iv* may produce messages that appear spurious but are actually correct. If the bad block is already in use as an alternate, the message can appear twice for one block.

Do not run *mkfs*(1M) on an overlapping partition.

Do not use Partition Table Descriptions from pre-5.0 versions of CTIX that specify partitions by track numbers, rather than by logical block boundaries.

The system does not allow dumps of size greater than or equal to 64 MB.



## NAME

machid: mc68k, miti, mini, mega, unixpc, i386, i286, pdp11, u3b, u3b2, u3b5, u3b15, u370, vax - get processor type truth value

## SYNOPSIS

**mc68k**

**miti**

**mini**

**mega**

**unixpc**

**i386**

**i286**

**pdp11**

**u3b**

**u3b2**

**u3b5**

**u3b15**

**u370**

**vax**

## DESCRIPTION

The following commands return a true value (exit code of 0) if you are on a processor that the command name indicates.

**mc68k** True if you are on a 68000-, 68010-, 68020-, or 68040-based computer.

**miti** True if you are on an S/Series computer.

**mini** True if you are on a MiniFrame computer.

**mega** True if you are on an S/1280 computer.

**unixpc** True if you are on a Unix PC computer.

**i386** True if you are on an Intel 80386-based computer.

**i286** True if you are on an Intel 80286-based computer.

**pdp11** True if you are on a PDP-11/45 or PDP-11/70.

**u3b** True if you are on a 3B20 computer.

- u3b2** True if you are on a 3B2 computer.
- u3b5** True if you are on a 3B5 computer.
- u3b15** True if you are on a 3B15 computer.
- u370** True if you are on an IBM 370 computer.
- vax** True if you are on a VAX-11/750 or VAX-11/780.

The commands that do not apply will return a false (nonzero) value. These commands are often used within makefiles [see *make(1)*] and shell procedures [see *sh(1)*] to increase portability.

**SEE ALSO**

*make(1)*, *sh(1)*, *test(1)*, *true(1)*.

**NAME**

`mcs` - manipulate the object file comment section

**SYNOPSIS**

`mcs` [ options ] object-file ...

**DESCRIPTION**

The `mcs` command manipulates the comment section, normally the “.comment” section, in an object file. It is used to add, delete, print, and compress the contents of the comment section in a CTIX system object file. `mcs` must be given one or more of the options described below. It takes each of the options given and applies them in order to the *object-files*.

If the object file is an archive, the file is treated as a set of individual object files. For example, if the `-a` option is specified, the string is appended to the comment section of each archive element.

The following options are available.

**-a** *string*

Appends *string* to the comment section of the *object-files*. If *string* contains embedded blanks, it must be enclosed in quotation marks.

**-c**

Compresses the contents of the comment section. All duplicate entries are removed. The ordering of the remaining entries is not disturbed.

**-d**

Deletes the contents of the comment section from the object file. The object file comment section header is removed also.

**-n** *name*

Specifies the name of the section to access. By default, `mcs` deals with the section named *.comment*. This option can be used to specify another section.

**-p**

Prints the contents of the comment section on the standard output. If more than one name is specified, each entry printed is tagged by the name of the file from which it was extracted, using the format “filename:string.”

**EXAMPLES**

To print a file’s *comment* section:

`mcs -p file`

To append *string* to *file*’s *comment* section:

`mcs -a string file`

## FILES

<code>/usr/tmp/mcs*</code>	temporary files
<code>/usr/tmp/*</code>	temporary files

## SEE ALSO

`cpp(1)`, `a.out(4)`.

## NOTES

`mcs` cannot insert or delete comment sections in executable objects with magic number 0413. (By default, `ld` creates executable objects with magic number 0413 [see `a.out(4)` and `ld(1)`].) However, the `-d` option to `mcs` can make the comment section of a 0413 file zero-length. This allows use of `mcs` with the `-a` and `-c` options on such files. All libraries provided with CTIX have comment sections in each library member as well as in the C start-up routines; thus, there would normally be a comment section in every `a.out` file.

**NAME**

mkfs - construct a file system

**SYNOPSIS**

```
/etc/mkfs special [ -O ] [ -M ] [ -E ] blocks[:i-nodes] [ gap blocks/cyl ]
[ -b blocksize ]
```

```
/etc/mkfs special [ -O ] [ -M ] [ -E ] proto [ gap blocks/cyl ] -b blocksize ]
```

```
/etc/mkfs special [ -O ] [ -M ] [ -E ]
```

**DESCRIPTION**

The *mkfs* command constructs a file system by writing on the *special* file using the values found in the remaining arguments of the command line (except in the case of the third form of the command, discussed below). The command waits 10 seconds before starting to construct the file system. During this 10-second pause, the command can be aborted with a delete character.

The **-b** *blocksize* option specifies the logical block size for the file system: the number of bytes read or written by the operating system in a single I/O operation. Valid values for *blocksize* are 1024 and 4096. If the **-b** option is omitted, the default block size is 1024.

Note that if you make a 4K file system, you must add the **buffers\_4k** parameter in the *dfile* to specify the number of 4K file system buffers to use.

If the second argument is a string of digits, the size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* (512-byte) disk blocks the file system occupies. If the number of i-nodes is not given, the default is the number of *logical* (1024- or 4096-byte) blocks divided by 4 (rounded down); i-nodes are allocated in groups of 16. The *mkfs* command builds a file system with a single empty directory on it. The boot program block (block zero) is left uninitialized.

If the second argument is the name of a file that can be opened, *mkfs* assumes it to be a prototype file *proto*, and takes its directions from that file. The prototype file contains tokens separated by spaces or new-lines. A sample prototype specification follows (line numbers are added for clarity):

```
1      /stand/diskboot
2      4872 110
3      d--777 3 1
4      usr      d--777 3 1
5              sh      ---755 3 1 /bin/sh
6              ken      d--755 6 1
7              $
```

```

8          b0      b-644 3 1 0 0
9          c0      c-644 3 1 0 0
10         $
11        $

```

Line 1 in the example is the name of a file to be copied onto block zero as the bootstrap program.

Line 2 specifies the number of *physical* (512-byte) blocks the file system is to occupy and the number of *i*-nodes in the file system.

Lines 3 through 9 tell *mkfs* about files and directories to be included in this file system.

Line 3 specifies the **root** directory.

Lines 4 through 6, 8, and 9 specify other directories and files.

The dollar sign (\$) on line 7 instructs *mkfs* to end the branch of the file system it is on and continue from the next higher directory. The dollar signs on lines 10 and 11 end the process, since no additional specifications follow.

File specifications give the mode, the user ID, the group ID, and the initial contents of the file. Valid syntax for the contents field depends on the first character of the mode.

The mode for a file is specified by a six-character string. The first character specifies the type of the file. The character range is **-bcd** to specify regular, block special, character special, and directory files, respectively. The second character of the mode is either **u** or **-**, to specify set-user-ID mode or not. The third character is **g** or **-**, for the set-group-ID mode. The rest of the mode is a three-digit octal number specifying the owner, group, and other read, write, and execute permissions [see *chmod*(1)].

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token of the specification can be a pathname to where the contents and size are copied. If the file is a block or character special file, two decimal numbers follow, which give the major and minor device numbers. If the file is a directory, *mkfs* makes the entries **.** and **..** and then reads a list of names and (recursively) file specifications for the entries in the directory. As noted above, the scan is terminated with the token dollar sign (\$).

The first two forms of the command allow the rotational *gap* and the number of *blocks/cyl* to be specified. The default *gap* size is 7. The default *blocks/cylinder* is 400. The default is used if the supplied *gap* and *blocks/cyl* are considered illegal values, or if a short argument count occurs.

The **-O** option makes a file system with a free list instead of a bit map. This is the default for removeable disks.

The **-M** option makes a file system with a bit map in addition to a free list. This is the default for fixed disks.

The **-E** option makes a file system with a bit map only one file system.

*Special* must be a disk slice. The third form of the *mkfs* command extracts the slice size from the Volume Home Block and creates a file system the same size; this third option cannot be used where there are overlapping partitions. The number of i-nodes is the number of logical blocks divided by 4. Optimal values for *gap* size and *blocks/cylinder* are calculated; these may not be 7 and 400.

#### SEE ALSO

chmod(1), dir(4), fs(4).  
*S/Series CTIX Administrator's Guide.*

#### BUGS

With a prototype file, there is no way to specify links. The maximum number of i-nodes configurable is 65500.

—

—

—



**NAME**

mount, umount - mount and unmount file systems and remote resources

**SYNOPSIS**

*/etc/mount*

*/etc/mount* directory

*/etc/mount* [ **-r** ] [ **-f** fstyp ] special directory

*/etc/mount* [ **-r** ] **-f** NFS [,options ] special directory

*/etc/mount* [ **-r** ] [ **-c** ] **-d** resource directory

*/etc/umount* special

*/etc/umount* directory

*/etc/umount* [ **-d** ] resource

**DESCRIPTION**

File systems other than **root** (/) are considered *removable* in the sense that they can be either available to users or unavailable. The *mount* command announces to the system that *special* (a block special device) or *resource* (a remote resource) is available to users from the mount point *directory*. Note that *directory* must already exist; it becomes the name of the root of the newly mounted *special* or *resource*. A unique resource can be mounted only once (no multiple mounts).

When invoked with no arguments, *mount* displays the entire mount table. When entered with arguments, *mount* adds an entry to the table of mounted devices, */etc/mnttab*. The *umount* command removes the entry. If invoked with any of the following partial argument lists, *mount* searches */etc/fstab* for the missing arguments: *special*, **-d resource**, *directory*, or **-d directory**. No special syntax is required to mount a 4K file system.

The following options are available:

- v** With no other arguments, prints a more verbose mount table containing file system type identifier (S51K, DUFST, NFS); with other arguments, prints the fully expanded *mount* command before mounting.
- r** Indicates that *special* or *resource* is to be mounted read-only. If *special* or *resource* is write-protected or read-only advertised, this flag must be used.
- c** Indicates that remote reads and writes should not be cached in the local buffer pool. **-c** is used in conjunction with **-d**.

- d** Indicates that *resource* is a remote resource to be mounted on *directory* or unmounted. To mount a remote resource, Remote File Sharing (RFS) or the Network File System (NFS) must be up and running, and the resource must be advertised by a remote computer [see *rfstart*(1M) and *adv*(1M)]. If **-d** is not used, and this is not an NFS mount, *special* must be a local block special device.
- f *fstyp*** Indicates that *fstyp* is the file system type to be mounted. If this argument is omitted, it defaults to the **root** *fstyp*.

If *fstyp* is NFS, NFS comma-separated options can be added after the *fstyp*. The available NFS options follow:

- soft** Returns error if the server does not respond.
- rsiz=*n*** Sets the read-buffer size to *n* bytes.
- wsiz=*n*** Sets the write-buffer size to *n* bytes.
- timeo=*n*** Sets the initial NFS timeout to *n* tenths of a second.
- retrans=*n*** Sets the number of NFS retransmissions to *n*.
- port=*n*** Sets the server IP port number to *n*.
- special* Indicates the block special device to be mounted on *directory*. If *fstyp* is NFS, *special* should be of the form *hostname:/pathname*.
- resource* Indicates the remote resource name to be mounted on *directory*.
- directory* Indicates the directory mount point for *special* or *resource*. (The directory must already exist.)

The *umount* command announces to the system that the file system previously mounted *special* or *resource* is to be made unavailable. If invoked with an incomplete argument list, *umount* searches */etc/fstab* for the missing arguments.

Note that *mount* can be used by any user to list mounted file systems and resources. Only the superuser can mount and unmount file systems.

## FILES

- /etc/mnttab* mount table
- /etc/fstab* file system table

## SEE ALSO

*adv*(1M), *fuser*(1M), *mountd*(1M), *nfsd*(1M), *nsquery*(1M), *rfstart*(1M), *rmntstat*(1M), *setmnt*(1M), *showmount*(1M), *unadv*(1M), *mount*(2), *umount*(2), *fstab*(4), *mnttab*(4).

*S/Series CTIX Administrator's Guide.*

**DIAGNOSTICS**

If the *mount(2)* system call fails, *mount* prints an appropriate diagnostic. The *mount* command issues a warning if the file system to be mounted is currently mounted under another name. A remote resource mount fails if the resource is not available, or if it is advertised read-only and not mounted with *-r*, or if Remote File Sharing is not running.

The *umount* command fails if *special* or *resource* is not mounted or if it is busy. *special* or *resource* is busy if it contains an open file or some user's working directory. In such a case, use *fuser(1M)* to list and kill processes using *special* or *resource*.

—

—

—

**NAME**

*occ* - old C compiler

**SYNOPSIS**

*occ* [ options ] files

**DESCRIPTION**

The *occ* command is the interface to the pre-6.3 CTIX C Compilation System. The compilation tools consist of a preprocessor, compiler, optimizer, assembler, and link editor. The *occ* command processes the supplied options and then executes the various tools with the proper arguments. The *occ* command accepts several types of files as arguments.

Files whose names end with *.c* are taken to be C source programs and may be preprocessed, compiled, optimized, assembled, and link edited. The compilation process may be stopped after the completion of any pass if the appropriate options are supplied. If the compilation process runs through the assembler, then an object program is produced and is left in the file whose name is that of the source with *.o* substituted for *.c*. However, the *.o* file is normally deleted if a single C program is compiled and then immediately link edited. In the same way, files whose names end in *.s* are taken to be assembly source programs, and may be assembled and link edited; files whose names end in *.i* are taken to be preprocessed C source programs and may be compiled, optimized, assembled, and link edited. Files whose names do not end in *.c*, *.s*, or *.i* are handed to the link editor.

Since the *occ* command usually creates files in the current directory during the compilation process, it is necessary to run the *occ* command in a directory in which a file can be created. The following options are interpreted by *occ*.

- #**
- ##**
- ###** These options cause *occ* to display each command that it would generate if it were to execute, but to fully execute only in the case of **-#**. Thus, **-#** specifies execution in verbose mode; **-##** specifies verbose mode (what *occ* would do if it were to execute) and checks permissions on all necessary files, but does not compile; and **-###** specifies verbose mode (what *occ* would do if it were to execute), but does nothing.
- c** Suppresses the link editing phase of the compilation, and does not remove any produced object files.
- g** Causes the compiler to generate additional information needed for the *sdb* (1).

**-o *outfile***

Produces an output object file by the name *outfile*. The name of the default file is **a.out**. This is a link editor option.

**-p** Arranges for the compiler to produce code that counts the number of times each routine is called; also, if link editing takes place, profiled versions of *libc.a* and *libm.a* (with **-lm** option) are linked and *monitor(3C)* is automatically called. A **mon.out** file will then be produced at normal termination of execution of the object program. An execution profile can then be generated by *prof(1)*.

**-w** Tells the linker (*ld*) not to print warnings about symbols that are partially matched. This option is meaningful only when the **-T** option is also specified.

**-Bstring****-t/p02a/**

These options will be removed in the next release. Use the **-Y** option.

**-E** Runs only *ocpp* on the named C programs, and sends the result to the standard output.

**-H** Prints out on *stderr* the pathname of each file included during the current compilation.

**-O** Compiles with optimization. This option will not have any effect on *.s* files.

**-P** Runs only *ocpp* on the named C programs and leaves the result in corresponding files suffixed *.i*. This option is passed to *ocpp*.

**-S** Compiles and does not assemble the named C programs; leaves the assembler-language output in corresponding files suffixed *.s*.

**-T** Truncates variable names to eight characters. Tells the loader to match eight character names (same as **-G** in the loader).

**-Wc, arg1[, arg2...]**

Hands off the argument[s] *argi* to pass *c* where *c* is one of **[p02a]** indicating the preprocessor, compiler, optimizer, assembler, or link editor, respectively. For example: **-Wa,-m** passes **-m** to the assembler.

**-Y** [**p02a**lSILUc], *dirname* | *processor*

Specifies a new pathname, *dirname*, for the locations of the tools and directories designated in the first argument; or selects a processor type, *processor*, for which to generate code. [**p02a**lSILUc] represents:

**p** preprocessor  
**0** compiler  
**2** optimizer  
**a** assembler  
**l** link editor  
**S** directory containing the start-up routines  
**I** default include directory searched by *ocpp*  
**L** first default library directory searched by *old*  
**U** second default library directory searched by *old*  
**c** selects the processor type, specified by the second argument, for which to generate code: **68020**, **68010**, and **68881**. For example, **-Y c,68020** selects the 68020 processor with software floating-point instructions. Note that **68881** implies **68020**.

If the location of a tool is being specified, then the new pathname for the tool will be *dirname/tool*. If more than one **-Y** option is applied to any one tool or directory, then the last occurrence holds.

The *occ* command also recognizes **-C**, **-D**, **-H**, **-I**, and **-U** and passes these options and their arguments directly to the preprocessor without using the **-W** option. Similarly, the *occ* command recognizes **-l**, **-m**, **-o**, **-r**, **-s**, **-t**, **-u**, **-w**, **-x**, **-z**, **-F**, **-G**, **-L**, **-M**, **-N**, **-V**, and **-Z** and passes these options and their arguments directly to the loader. See the *cpp*(1) and *ld*(1) man pages for descriptions.

Other arguments are taken to be C-compatible object programs, typically produced by an earlier *occ* run, or perhaps libraries of C-compatible routines, and are passed directly to the link editor. These programs, together with the results of any compilations specified, are link edited (in the order given) to produce an executable program with name **a.out** unless the **-o** option of the link editor is used.

The *occ* command uses a filename *prefixcc*, so the prefix is parsed off the command and used to call the tools, that is, *prefixtool*. For example, **OLDcc** will call **OLDcpp**, **OLDcomp**, **OLDoptim**, **OLDas**, and **OLDld** and will link **OLDcrt1.o**. Therefore, you **MUST** be careful when moving the *occ* command around. The prefix will apply to the preprocessor, compiler, optimizer, assembler, link editor, and the start-up routines.

The C compiler uses one of three code generators for the 68010, 68020, and 68020/68881. There are several ways to select a particular code generator, but the selection is normally done using one of two basic mechanisms.

The first is to specify the processor on the *occ* command line, for example, by using the **-Y** option. (An equivalent mechanism is provided by the *gcc(1)* command, and also by the *cc1sw(1)*, *cc2sw*, or *cc2fp* commands.) The **-Y** option has additional arguments that allow you to specify pathnames of default libraries, include files, and tools as described earlier.

The second mechanism is to use the CENVIRON shell variable. Note that the first mechanism, specifying the processor and/or search path of libraries and include files, overrides the CENVIRON and any other shell variable settings.

The CENVIRON variable has the following syntax:

```
CPU=xxxxx,FPU=yyyyy
```

where CPU indicates the central processor to generate for and FPU indicates the style of floating-point math to use. *xxxxx* may be 68010 or 68020, and *yyyyy* may be 68881 or SOFTWARE. The FPU parameter may be omitted; the default is SOFTWARE. The CENVIRON variable should always be set to the appropriate values in the **.profile** or **.cshrc** files or in the makefile. [See *hinv(1M)*.]

The C compiler interprets two shell variables that, along with the CENVIRON variable, allow cross-compilation for any CTIX machine:

LIBROOT	This variable is a path that is prepended to normal library names when searching for a library. See also <i>ld(1)</i> .
INCROOT	This variable is a path that is prepended to the <b>/usr/include</b> and <b>/usr/include/sys</b> directories during include file searches. See also <i>cpp(1)</i> .

The C language standard was extended to allow arbitrary length variable names. The option pair “**-Wp,-T -W0,-XT**” will cause *occ* to truncate arbitrary length variable names to eight characters.

## FILES

<i>file.c</i>	C source file
<i>file.i</i>	preprocessed C source file
<i>file.o</i>	object file
<i>file.s</i>	assembly language file
<i>a.out</i>	link edited output
<i>LIBDIR/*ocrt1.o</i>	start-up routine
<i>LIBDIR/ocrt.n.o</i>	start-up routine



<i>TMPDIR</i> /*	temporary files
<i>LIBDIR</i> /ocpp	preprocessor, <i>cpp</i> (1)
<i>LIBDIR</i> /occom	68010 compiler
<i>LIBDIR</i> /occom20	68020 compiler
<i>LIBDIR</i> /occom20.81	68020/68881 compiler
<i>LIBDIR</i> /ooptim	optimizer
<i>BINDIR</i> /oas	assembler, <i>as</i> (1)
<i>BINDIR</i> /old	link editor, <i>ld</i> (1)
<i>LIBDIR</i> /libc.a	standard C library
<i>LIBDIR</i> /libc_s.a	standard C shared library

*LIBDIR* is usually /lib

*BINDIR* is usually /bin

*TMPDIR* is usually /tmp but can be redefined by setting the environment variable *TMPDIR* [see *tempnam*( ) in *tempnam*(3S)].

#### SEE ALSO

*as*(1), *ld*(1), *cc1sw*(1), *cpp*(1), *gcc*(1), *lint*(1), *prof*(1), *sdb*(1), *tempnam*(3S).

*Programmer's Guide: CTIX Supplement.*

Kernighan, B.W., and Ritchie, D.M., *The C Programming Language*, Prentice-Hall, 1978.

#### CAVEATS

*occ* will complain if it encounters inconsistencies between the processor selected and default libraries or include files.

Sometimes the range for a branch does not fit inside a word; in this case, an error message is printed. For suggested workarounds, see the section called "Span-Dependent Optimization" in Chapter 14 of the *Programmer's Guide: CTIX Supplement*.

#### DIAGNOSTICS

The diagnostics produced by the C compiler are sometimes cryptic. Occasional messages may be produced by the assembler or link editor.

#### NOTES

By default, the return value from a compiled C program is completely random. The only two guaranteed ways to return a specific value is to explicitly call *exit*(2) or to leave the function *main*( ) with a "return expression;" construct.

—

—

—

**NAME**

passmgmt - password files management

**SYNOPSIS**

**passmgmt -a** options name  
**passmgmt -m** options name  
**passmgmt -d** name

**DESCRIPTION**

The *passmgmt* command updates information in the password files. This command works with both */etc/passwd* and */etc/shadow*. If there is no shadow password file, the changes done by *passmgmt* will go in */etc/passwd*.

The *passmgmt -a* form of the command adds an entry for user *name* to the login password files. This command does not create any directory for the new user and the new login remains locked (with the string **\*LK\*** in the *password* field) until the *passwd(1)* command is executed to set the password.

The *passmgmt -m* form of the command modifies the entry for user *name* in the login password files. The name field in the */etc/shadow* entry and all the fields (except the password field) in the */etc/passwd* entry can be modified by this command. Only fields entered on the command line are modified. If there is no */etc/shadow* file, all modifications are made in */etc/passwd*.

The *passmgmt -d* command deletes the entry for user *name* from the login password files. It does not remove any files the user owns on the system; they must be removed manually.

The login name of the user, *name*, must be unique.

The following options are available:

- c** *comment* A short description of the login. It is limited to a maximum of 128 characters and defaults to an empty field. If the comment is more than one word, it must be enclosed in single or double quotation marks.
- h** *homedir* Home directory of *name*. It is limited to a maximum of 256 characters and defaults to */u/name*.
- u** *uid* UID of the *name*. This number must range from 0 to the maximum value for the system. It defaults to the next available UID greater than 100. Without the **-o** option, it enforces the uniqueness of a UID.
- o** This option allows a UID to be nonunique. It is used only with the **-u** option.

- g *gid***      GID of the *name*. This number must range from 0 to the maximum value for the system. The default is 1.
  - s *shell***      Login shell for *name*. It should be the full pathname of the program to be executed when the user logs in. The maximum length of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as **/bin/sh**.
  - l *logname***      This option changes the *name* to *logname* for the **-m** option only.
- The total size of each login entry, whether existing or new, is limited to a maximum of 511 bytes in the password files.

**FILES**

/etc/passwd  
 /etc/shadow  
 /etc/opasswd  
 /etc/oshadow

**SEE ALSO**

passwd(1), passwd(4).

**DIAGNOSTICS**

The *passmgmt* command exits with one of the following values:

- 0      Success.
- 1      Permission denied.
- 2      Invalid command syntax. Usage message of the *passmgmt* command is displayed.
- 3      Invalid argument provided to option.
- 4      UID in use.
- 5      Inconsistent password files (for example, *name* is in the **/etc/passwd** file and not in the **/etc/shadow** file, or vice versa).
- 6      Unexpected failure. Password files unchanged.
- 7      Unexpected failure. Password file(s) missing.
- 8      Password file(s) busy. Try again later.
- 9      *name* does not exist (if **-m** or **-d** is specified), already exists (if **-a** is specified), or *logname* already exists (if **-m -l** is specified).

**NOTE**

Do not use a colon or carriage return; these characters are interpreted as field separators.

**NAME**

`passwd` - change login password

**SYNOPSIS**

`passwd` [ *name* ]

`passwd -s` [ *name* ]

`passwd -l` [ *-f* ] [ *-x max* ] [ *-n min* ] *name*

`passwd -d` [ *-f* ] [ *-x max* ] [ *-n min* ] *name*

`passwd -s` [ *-a* ]

**DESCRIPTION**

The *passwd* command changes, sets, or lists attributes of a password associated with the login *name*. Ordinary users can change only the password that corresponds to their login *name*; the superuser can additionally set or change passwords and attributes associated with any login *name*.

When used to change a password, *passwd* prompts ordinary users for their old password, if any; it then prompts for the new password twice. When the old password is entered, *passwd* checks to see if the old password has “aged” sufficiently. Password “aging” is the amount of time (usually a number of days) that must elapse between password changes. If aging is insufficient, *passwd* terminates; see *passwd*(4).

Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated (at most, two more times).

Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.
- Each password must contain at least two alphabetic (upper- and lowercase) characters and at least one numeric or special character.
- Each password must differ from the user’s login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.
- New passwords must differ from the old by at least three characters. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

One whose effective user ID is zero is called a superuser; see *id(1)* and *su(1)*. Superusers can change any password; therefore, *passwd* does not prompt superusers for the old password. Superusers are not forced to comply with password aging and password construction requirements. A superuser can create a null password by entering a carriage return in response to the prompt for a new password.

Any user can use the *-s* option to show password attributes for the login *name*.

The format of the display is as follows:

```
name status mm/dd/yy min max
```

or, if password aging information is not present,

```
name status
```

where:

- name* Is the login ID of the user.
- status* Is the password status of *name*: PS stands for passworded or locked, LK stands for locked, and NP stands for no password.
- mm/dd/yy* Is the date the password was last changed for *name*. Note that all password aging dates are determined by using Greenwich Mean Time and, therefore, may differ by as much as a day in other time zones.
- min* Is the minimum number of days required between password changes for *name*.
- max* Is the maximum number of days the password is valid for *name*.

Only a superuser can use the following options:

- l* Locks the password entry for *name*.
- d* Deletes the password for *name*. The login *name* is not prompted for a password.
- n* Sets the minimum field for *name*. The *min* field contains the minimum number of days between password changes for *name*. If *min* is greater than *max*, the user cannot change the password. Always use this option with the *-x* option, unless *max* is set to -1 (aging disabled), in which case, *min* need not be set.
- x* Sets the maximum field for *name*. The *max* field contains the number of days the password is valid for *name*. The aging for

*name* is disabled immediately if *max* is set to -1. If *max* is set to 0, the user must change the password at the next login session, and aging is disabled.

- a Shows password attributes for all entries. Use only with the -s option; *name* must not be provided.
- f Forces the user to change the password at the next login by expiring the password for *name*.

## FILES

/etc/passwd  
 /etc/shadow  
 /etc/opasswd  
 /etc/oshadow

## SEE ALSO

id(1M), login(1), passmgmt(1M), pwconv(1M), su(1M), crypt(3C), passwd(4).

## DIAGNOSTICS

The *passwd* command exits with one of the following values:

- 0 Success.
- 1 Permission denied.
- 2 Invalid combination of options (incorrect syntax).
- 3 Unexpected failure. Password file unchanged.
- 4 Unexpected failure. Password file(s) missing.
- 5 Password file(s) busy. Try again later.
- 6 Invalid argument to option.

## WARNINGS

If the optional */etc/shadow* file exists, *passwd* uses that file instead of */etc/passwd* to obtain password information. Because the two files store password aging information in different ways, the output from the *passwd* options can differ.

—

—

—



**NAME**

profiler: *prfld*, *prfstat*, *prfdc*, *prfsnap*, *prfpr* - operating system profiler

**SYNOPSIS**

*/etc/prfld* [ *namelist* ]  
*/etc/prfstat on*  
*/etc/prfstat off*  
*/etc/prfdc* file [ *period* [ *off\_hour* ] ]  
*/etc/prfsnap* file  
*/etc/prfpr* file [ *cutoff* [ *namelist* ] ]  
*S/4040*, *S/640*, *S/480*, *S/280*, and *S/80* only:  
*/etc/prfstat* time

**DESCRIPTION**

The *prfld*, *prfstat*, *prfdc*, *prfsnap*, and *prfpr* programs form a system of programs to facilitate an activity study of the CTIX operating system. A kernel configured with kernel profiling must be used: the **pfr** driver may be loaded with *lddrv*(1M).

The *prfld* program is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from *namelist*.

The *prfstat* program is used to enable or disable the sampling mechanism. Profiler overhead is less than one percent as calculated for 500 text addresses. Note that *prfstat* also reveals the number of text addresses being measured.

Addresses are sampled every clock tick (definition for *Hz* is given in *param.h*). *S/4040*, *S/640*, *S/480*, *S/280*, and *S/80* systems allow sampling every *time* microsecond: the lower limit is 100 microsecond intervals.

The *prfdc* and *prfsnap* programs perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. The *prfdc* program stores the counters into *file* every *period* minutes and turns off at *off\_hour* (valid values for *off\_hour* are 0-24). The *prfsnap* program collects data at the time of invocation only, appending the counter values to *file*.

The *prfpr* program formats the data collected by *prfdc* or *prfsnap*. Each text address is converted to the nearest text symbol (as found in *namelist*) and is printed if the percent activity for that range is greater than *cutoff*.

**FILES**

/dev/prf interface to profile data and text addresses

/etc/lldrv/unix.exec default for namelist file

**SEE ALSO**

prf(7).

**NAME**

`pwconv` - install and update `/etc/shadow` with information from `/etc/passwd`

**SYNOPSIS**

`pwconv`

**DESCRIPTION**

The `pwconv` command creates and updates `/etc/shadow` with information from `/etc/passwd`. If the `/etc/shadow` file does not exist, `pwconv` creates it with information from `/etc/passwd`. The command populates `/etc/shadow` with the user's login name, password, and password aging information. If password aging information does not exist in `/etc/passwd` for a given user, none is added to `/etc/shadow`; however, the "last changed" information is always updated.

If the `/etc/shadow` file does exist, the following tasks are performed:

- Entries in the `/etc/passwd` file but not in the `/etc/shadow` file are added to the `/etc/shadow` file.
- Entries in the `/etc/shadow` file but not in the `/etc/passwd` file are removed from `/etc/shadow`.
- Password attributes (such as password and aging information) that exist in an `/etc/passwd` entry are moved to the corresponding entry in `/etc/shadow`.

The `pwconv` program is a privileged system command that can be executed only by the superuser. The `passwd` command should be used to add or change password aging information or passwords.

**FILES**

`/etc/passwd`  
`/etc/shadow`  
`/etc/opasswd`  
`/etc/oshadow`

**WARNING**

If malformed lines are encountered in `/etc/passwd` or `/etc/shadow`, these files may be truncated to the last good entry.

**SEE ALSO**

`passwd(1M)`, `passmgmt(1M)`, `pwunconv(1M)`.

**DIAGNOSTICS**

The *pwconv* command exits with one of the following values:

- 0 Success.
- 1 Permission denied.
- 2 Invalid command syntax.
- 3 Unexpected failure. Conversion not done.
- 4 Unexpected failure. Password file(s) missing.
- 5 Password file(s) busy. Try again later.

## NAME

sar - system activity reporter

## SYNOPSIS

**sar** [-ubdycwaqvmprDSAC] [-o file] t [ n ]

**sar** [-ubdycwaqvmprDSAC] [-s time] [-e time] [-i sec] [-f file]

## DESCRIPTION

In the first instance, *sar* samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second instance, with no sampling interval specified, **sar** extracts data from a previously recorded *file*, either the one specified by the **-f** option, or by default, the standard system activity daily data file */usr/adm/sa/sadd* for the current day *dd*. The starting and ending times of the report can be bounded by use of the **-s** and **-e time** arguments of the form *hh[:mm[:ss]]*. The **-i** option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by the following options:

**-u** Reports CPU utilization (the default):

**%usr, %sys, %wio, %idle**

Portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle. When used with **-D**, **%sys** is split into percentage of time servicing requests from remote machines (**%sys remote**) and all other system time (**%sys local**).

**-b** Reports buffer activity:

**bread/s, bwrit/s** Transfers per second of data between system buffers and disk or other block devices.

**lread/s, lwrit/s** Accesses of system buffers.

**%rcache, %wcache**

Cache-hit ratios: that is, (1-bread/lread) as a percentage.

**pread/s, pwrit/s** Transfers through raw (physical) device mechanism. When used with **-D**, buffer caching is reported for locally-mounted remote resources.

- d Reports activity for each block device; for example, disk or tape drive. When data is displayed, the device specification *dsk*- is generally used to represent a disk drive. The following activity data is reported:
  - %busy, avque** Portion of time device was busy servicing a transfer request, and average number of requests outstanding during that time.
  - r+w/s, blks/s** Number of data transfers from or to device, and number of bytes transferred in 512-byte units.
  - await, avserv** Average time in milliseconds that transfer requests wait idly on queue, and average time to be serviced (which for disks includes seek, rotational latency, and data transfer times). RS-422 activity is also reported in this section.
- y Reports TTY device activity:
  - rawch/s, canch/s, outch/s** Input character rate, input character rate processed by canon, and output character rate.
  - rcvin/s, xmtin/s, madmin/s** Receive, transmit, and modem interrupt rates.
- c Reports system calls:
  - scall/s** System calls of all types.
  - sread/s, swrit/s, fork/s, exec/s** Specific system calls.
  - rchar/s, wchar/s** Characters transferred by read and write system calls. When used with **-D**, the system calls are split into strictly local calls, remote outgoing (client) calls, and remote incoming (server) calls.
- w Reports system swapping and switching activity:
  - swpin/s, swpot/s, bswin/s, bswot/s** Number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs).
  - pswch/s** Process switches.
- a Reports use of file access system routines: **iget/s, namei/s,** and **dirblk/s**.

- q** Reports average queue length while occupied, and percentage of time occupied:
- runq-sz, %runocc** Run queue of processes in memory and runnable.
- swpq-sz, %swpocc** Swap queue of processes swapped out but ready to run.
- v** Reports status of process, i-node, and file tables:
- proc-sz, inod-sz, file-sz, lock-sz, fhdr-sz** Entries/size for each table, evaluated once at sampling point.
- ov** Overflows that occur between sampling points for each table.
- m** Reports message and semaphore activities:
- msg/s, sema/s** Primitives per second.
- p** Reports paging activities:
- vflt/s** Address translation page faults (valid page not in memory).
- pflt/s** Page faults from protection errors (illegal access to page) or copy-on-writes.
- pgfil/s** vflt/s satisfied by page-in from file system.
- rclm/s** Valid pages reclaimed for free list.
- r** Reports unused memory pages and disk blocks:
- freemem** Average pages available to user processes.
- freeswap** Disk blocks available for process swapping.
- D** Reports Remote File Sharing (RFS) activity. When used in combination with **-u**, **-b** or **-c**, it causes **sar** to produce the remote file sharing version of the corresponding report. **-Du** is assumed when only **-D** is specified.
- S** Reports server and request queue status:
- serv/lo-hi** Average number of Remote File Sharing servers on the system.
- request %busy** Percentage of time receive descriptors are on the request queue.

- request avg lgth** Average number of receive descriptors waiting for service when queue is occupied.
- server %avail** Percentage of time there are idle servers.
- server avg avail** Average number of idle servers when idle ones exist.
- A Reports all data. Equivalent to **-udqbwcaymprSDC**.
- C Reports Remote File Sharing (RFS) buffer caching overhead:
- snd-inv/s** Number of invalidation messages per second sent by your machine as a server.
- snd-msg/s** Total outgoing RFS messages sent per second.
- rcv-inv/s** Number of invalidation messages received from the remote server.
- rcv-msg/s** Total number of incoming RFS messages received per second.
- dis-bread/s** Number of buffer reads that would be eligible for caching if caching is not disabled. (Indicates the penalty of running uncached.)
- blk-inv/s** Number of buffers removed from the client cache.

## EXAMPLES

The following command reports today's CPU activity so far:

```
sar
```

The following command reports CPU activity over a period of 10 minutes and saves the data:

```
sar -o temp 60 10
```

The following command reports disk and tape activity saved from a previous *sar* (like that shown above) in which data was saved:

```
sar -d -f temp
```

## FILES

*/usr/adm/sa/sadd* daily data file, where *dd* are digits representing the day of the month.

## SEE ALSO

sag(1G), sar(1M).  
*S/Series CTIX Administrator's Guide.*



**NAME**

sdb - symbolic debugger

**SYNOPSIS**

**sdb** [ **-w** ] [ **-W** ] [ *objfil* [ *corfil* [ *directory-list* ] ] ]

**DESCRIPTION**

The *sdb* command calls a symbolic debugger that can be used with C programs. It can be used to examine their object files and core files and to provide a controlled environment for their execution.

*objfil* is an executable program file that has been compiled with the **-g** (debug) option. If *objfil* has not been compiled with the **-g** option, the symbolic capabilities of *sdb* are limited, but the file can still be examined and the program debugged. The default for *objfil* is **a.out**. *corfil* is assumed to be a core image file produced after executing *objfil*; the default for *corfil* is **core**. The core file need not be present. A **-** in place of *corfil* forces *sdb* to ignore any core image file. The colon-separated list of directories (*directory-list*) is used to locate the source files used to build *objfil*.

It is useful to know that at any time there is a *current line* and *current file*. If *corfil* exists, they are initially set to the line and file containing the source statement at which the process terminated. Otherwise, they are set to the first line in *main()*. The current line and file can be changed by using the source file examination commands.

Initially, *sdb* has an asterisk character (**\***) prompt, which indicates that *sdb* is ready for the user to enter the first command. If the **S**, **s**, **I**, or **i** command is used, the prompt corresponds to the command letter (for example, **S** when the **S** command is used).

By default, warnings are provided if the source files used in producing *objfil* cannot be found, or are newer than *objfil*. This checking feature and the accompanying warnings may be disabled by the **-W** flag.

Names of variables are written just as they are in C. *sdb* does not truncate names. Variables local to a procedure may be accessed using the form *procedure:variable*. If no procedure name is given, the procedure containing the current line is used by default.

It is also possible to refer to structure members as *variable.member*, pointers to structure members as *variable->member*, and array elements as *variable[number]*. Pointers may be dereferenced by using the form *pointer[0]*. Combinations of these forms may also be used. A number may be used in place of a structure variable name. In this case, the number is viewed as the address of the structure, and the template used for the structure is that of the last

structure referenced by *sdb*. An unqualified structure variable may also be used with various commands. Generally, *sdb* will interpret a structure as a set of variables. Thus, *sdb* will display the values of all the elements of a structure when it is requested to display a structure. An exception to this interpretation occurs when displaying variable addresses. An entire structure does have an address, and it is this value *sdb* displays, not the addresses of individual elements.

Elements of a multidimensional array may be referenced as *variable [number][number]...*, or as *variable [number,number,...]*. In place of *number*, the form *number;number* may be used to indicate a range of values, \* may be used to indicate all legitimate values for that subscript, or subscripts may be omitted entirely if they are the last subscripts and the full range of values is desired. As with structures, *sdb* displays all the values of an array or of the section of an array if trailing subscripts are omitted. It displays only the address of the array itself or of the section specified by the user if subscripts are omitted.

A particular instance of a variable on the stack may be referenced by using the form *procedure:variable,number*. All the variations mentioned in naming variables may be used. *Number* is the occurrence of the specified procedure on the stack, counting the top, or most current, as the first. If no procedure is specified, the procedure currently executing is used by default.

It is also possible to specify a variable by its address. All forms of integer constants that are valid in C may be used, so that addresses may be input in decimal, octal, or hexadecimal.

Line numbers in the source program are referred to as *file-name:number* or *procedure:number*. In either case, the number is relative to the beginning of the file. If no procedure or filename is given, the current file is used by default. If no number is given, the first line of the named procedure or file is used.

While a process is running under *sdb*, all addresses refer to the executing program; otherwise, they refer to *objfil* or *corfil*. An initial argument of *-w* permits overwriting locations in *objfil*.

### Addresses

The address in a file associated with a written address is determined by a mapping associated with that file. Each mapping is represented by two triples

(*b1*, *e1*, *f1*) and (*b2*, *e2*, *f2*) and the *file address* corresponding to a written *address* is calculated as follows:

$$b1 \leq \text{address} < e1$$

$$\text{file address} = \text{address} + f1 - b1$$

otherwise,

$$b2 \leq \text{address} < e2$$

$$\text{file address} = \text{address} + f2 - b2.$$

Otherwise, the requested *address* is not legal. In some cases (for example, for programs with separated I and D space), the two segments for a file may overlap.

The initial setting of both mappings is suitable for normal **a.out** and **core** files. If either file is not of the kind expected, then for that file, *b1* is set to 0, *e1* is set to the maximum file size, and *f1* is set to 0; in this way, the whole file can be examined with no address translation.

In order for *sdb* to be used on large files, all appropriate values are kept as signed 32-bit integers.

## Commands

The commands for examining data in the program are as follows:

**t** Prints a stack trace of the terminated or halted program.

**T** Prints the top line of the stack trace.

*variable/clm*

Prints the value of *variable* according to length *l* and format *m*. A numeric count *c* indicates that a region of memory, beginning at the address implied by *variable*, is to be displayed. The length specifiers are as follows:

<b>b</b>	one byte
<b>h</b>	two bytes (half word)
<b>l</b>	four bytes (long word)

Legal values for *m* are as follows:

<b>c</b>	character
<b>d</b>	decimal
<b>u</b>	decimal, unsigned

<b>o</b>	octal
<b>x</b>	hexadecimal
<b>f</b>	32-bit single precision floating point
<b>g</b>	64-bit double precision floating point
<b>s</b>	assume <i>variable</i> is a string pointer and print characters starting at the address pointed to by the variable
<b>a</b>	print characters starting at the variable's address; this format may not be used with register variables
<b>p</b>	pointer to procedure
<b>i</b>	disassemble machine-language instruction with addresses printed numerically and symbolically
<b>I</b>	disassemble machine-language instruction with addresses just printed numerically

Length specifiers are only effective with the **c**, **d**, **u**, **o**, and **x** formats. Any of the specifiers, **c**, **l**, and **m**, may be omitted. If all are omitted, *sdb* chooses a length and a format suitable for the variable's type as declared in the program. If **m** is specified, then this format is used for displaying the variable. A length specifier determines the output length of the value to be displayed, sometimes resulting in truncation. A count specifier **c** tells *sdb* to display that many units of memory, beginning at the address of *variable*. The number of bytes in one such unit of memory is determined by the length specifier **l**, or if no length is given, by the size associated with the *variable*. If a count specifier is used for the **s** or **a** command, then that many characters are printed. Otherwise, successive characters are printed until either a null byte is reached, or 128 characters are printed. The last variable may be redisplayed with the command *J*.

The *sh*(1) metacharacters **\*** and **?** may be used within procedure and variable names, providing a limited form of pattern matching. If no procedure name is given, variables local to the current procedure and global variables are matched; if a procedure name is specified, then only variables local to that procedure are matched. To match only global variables, the form *:pattern* is used.

*linenumber?lm*  
*variable:?lm*

Prints the value at the address from **a.out** or **I** space given by *linenumber* or *variable* (procedure name), according to the format *lm*. The default format is 'i'.

*variable=lm*  
*linenumber=lm*  
*number=lm*

Prints the address of *variable* or *linenumber*, or the value of *number*, in the format specified by *lm*. If no format is given, then *lx* is used. The last variant of this command provides a convenient way to convert between decimal, octal, and hexadecimal.

*variable!value*

Sets *variable* to the given *value*. The value may be a number, a character constant, or a variable. The value must be well defined; expressions that produce more than one value, such as structures, are not allowed. Character constants are denoted *'character*. Numbers are viewed as integers unless a decimal point or exponent is used. In this case, they are treated as having the type *double*. Registers are viewed as integers. The *variable* may be an expression that indicates more than one variable, such as an array or structure name. If the address of a variable is given, it is regarded as the address of a variable of type *int*. C conventions are used in any type conversions necessary to perform the indicated assignment.

- f Prints the 68881/68040 floating-point registers.
- x Prints the machine registers and the current machine-language instruction.
- X Prints the current machine-language instruction.

The commands for examining source files are as follows:

e *procedure*  
 e *file-name*  
 e *directory/*  
 e *directory file-name*

The first two forms set the current file to the file containing *procedure* or to *file-name*. The current line is set to the first line in the named procedure or file. Source files are assumed to be in *directory*. The default is the current working directory. The latter two forms change the value of *directory*. If no procedure, filename, or directory is given, the current procedure name and filename are reported.

*/regular expression/*

Searches forward from the current line for a line containing a string matching *regular expression* as in *ed(1)*. The trailing */* may be deleted.

*?regular expression?*

Searches backward from the current line for a line containing a string matching *regular expression* as in *ed(1)*. The trailing *?* may be deleted.

**p** Prints the current line.

**z** Prints the current line followed by the next nine lines. Sets the current line to the last line printed.

**w** Window. Prints the 10 lines around the current line.

*number*

Sets the current line to the given line number. Prints the new current line.

*count+*

Advances the current line by *count* lines. Prints the new current line.

*count-*

Retreats the current line by *count* lines. Prints the new current line.

The commands for controlling the execution of the source program are as follows:

*count r args**count R*

Runs the program with the given arguments. The **r** command with no arguments reuses the previous arguments to the program while the **R** command runs the program with no arguments. An argument beginning with **<** or **>** causes redirection for the standard input or output, respectively. If *count* is given, it specifies the number of breakpoints to be ignored.

*linenumber c count**linenumber C count*

Continues after a breakpoint or interrupt. If *count* is given, the program will stop when *count* breakpoints have been encountered. The signal that caused the program to stop is reactivated with the **C** command and ignored with the **c** command. If a line number is specified, a temporary breakpoint is placed at the line and execution is continued. This temporary breakpoint is deleted when the command finishes.

*linenumber g count*

Continues after a breakpoint with execution resumed at the given line. If *count* is given, it specifies the number of breakpoints to be ignored.

*s count*

**S** *count*

Single-steps the program through *count* lines. If no count is given, then the program is run for one line. **S** is equivalent to **s** except it steps through procedure calls.

**i**

**I** Single-steps by one machine-language instruction. The signal that caused the program to stop is reactivated with the **I** command and ignored with the **i** command.

*variable***\$m count**

*address***:m count**

Single-steps (as with **s**) until the specified location is modified with a new value. If *count* is omitted, it is effectively infinity. *variable* must be accessible from the current procedure. Since this command is done by software, it can be very slow.

*level v*

Toggles verbose mode, for use when single-stepping with **S**, **s**, or **m**. If *level* is omitted, then just the current source file and/or subroutine name is printed when either changes. If *level* is 1 or greater, each **C** source line is printed before it is executed; if *level* is 2 or greater, each assembler statement is also printed. A **v** turns verbose mode off if it is on for any level.

**k** Kills the program being debugged.

*procedure*(arg1,arg2,...)

*procedure*(arg1,arg2,...)/*m*

Executes the named procedure with the given arguments. Arguments can be integer, character, or string constants or names of variables accessible from the current procedure. The second form causes the value returned by the procedure to be printed according to format *m*. If no format is given, it defaults to **d**. This facility is only available if the program was loaded with the **-g** option.

*linenumber b commands*

Sets a breakpoint at the given line. If a procedure name without a line number is given (for example, "proc:'), a breakpoint is placed at the first line in the procedure even if it was not compiled with the **-g** option. If no *linenumber* is given, a breakpoint is placed at the current line. If no *commands* are given, execution stops just before the breakpoint and control is returned to *sdb*. Otherwise, the *commands* are executed when the breakpoint is encountered and execution continues. Multiple

commands are specified by separating them with semicolons. If **k** is used as a command to execute at a breakpoint, control returns to *sdb*, instead of continuing execution.

**B** Prints a list of the currently active breakpoints.

*linenumber d*

Deletes a breakpoint at the given line. If no *linenumber* is given, then the breakpoints are deleted interactively. Each breakpoint location is printed and a line is read from the standard input. If the line begins with a **y** or **d**, then the breakpoint is deleted.

**D** Delete all breakpoints.

**I** Print the last executed line.

*linenumber a*

Announces. If *linenumber* is of the form *proc:number*, the command effectively does a *linenumber b I*. If *linenumber* is of the form *proc:*, the command effectively does a *proc: b T*.

Miscellaneous commands:

*!command*

The command is interpreted by *sh*(1).

**new-line**

Performs the previous command again.

**end-of-file character**

Scrolls and prints the next 10 lines of instructions, source, or data depending on which was printed last. The end-of-file character is usually CONTROL-D.

*< filename*

Reads commands from *filename* until the end of file is reached, and then continues to accept commands from standard input. When *sdb* is told to display a variable by a command in such a file, the variable name is displayed along with the value. This command may not be nested; **<** may not appear as a command in a file.

**M** Prints the address maps.



**M** [?/] [\*] *b e f*

Records new values for the address map. The arguments ? and / specify the text and data maps, respectively. The first segment (*b1, e1, f1*) is changed unless \* is specified; in which case, the second segment (*b2, e2, f2*) of the mapping is changed. If fewer than three values are given, the remaining map parameters are left unchanged.

**"** *string*

Prints the given string. The C escape sequences of the form *\character* are recognized, where *character* is a nonnumeric character.

**q** Exits the debugger.

The following commands also exist and are intended only for debugging the debugger:

**V** Prints the version number.**Q** Prints a list of procedures and files being debugged.**Y** Toggles debug output.

*sdb* may be instructed to monitor a given memory location and stop the program when the value at that location changes in any given way. For example:

```
> if x <= 123
```

The above example instructs *sdb* to monitor the value at location *x*. When the user gives the command to continue (*c*), *sdb* checks the value of *x* at every source line executed and stops the program if the given condition becomes true. Note that this construct slows the real-time execution of a program.

The syntax of the *if* command is as follows:

**if** Shows a list of the current data breakpoints; assigns a number to each.

**if var** Monitors the value of *var* and stops the program if the value changes. A variable name may be used for *var*, as well as a constant address. Comparisons are done as either 4-byte signed or 4-byte unsigned, depending on the data type. To perform a 1-byte or 2-byte comparison, an optional length value may accompany *var*. An example of a 2-byte comparison is

```
if x,2 = 0xff
```

**if var rel value**

Compares the value of *var* to the constant given and stops the program if the condition is true. The values of *rel* may be =, ==, <, <=, >, >=, or !=.

- off *n*** Disables or turns off a data breakpoint without removing it from the list.
- on *n*** Enables a breakpoint that was turned off.
- out *n*** Removes a breakpoint from the list.

Conditional breakpoints are used in a manner similar to data breakpoints, except that the user specifies a place in the program at which *sdb* should stop to check the data values. For example,

```
mysub:99 b if xyz = 123
```

The above example instructs *sdb* to check the value of *xyz* every time the program arrives at line 99 of subroutine *mysub*. If the condition is true, then execution stops there, as with a normal breakpoint. This type of breakpoint does not monitor the value *xyz* at every line of code, as the data breakpoint does.

## FILES

a.out  
core

## SEE ALSO

cc(1), sh(1), a.out(4), core(4), syms(4).

## WARNINGS

When *sdb* prints the value of an external variable for which there is no debugging information, a warning is printed before the value. The size is assumed to be **int** (integer).

Data stored in text sections are indistinguishable from functions.

Line number information in optimized functions is unreliable, and some information may be missing.

## BUGS

If a procedure is called when the program is *not* stopped at a breakpoint (such as when a core image is being debugged), all variables are initialized before the procedure is started. This makes it impossible to use a procedure that formats data from a core image.

When setting a breakpoint at a procedure, *sdb* will inconsistently produce the incorrect line number. Recompiling the source program will correct this problem.

**NAME**

*serstat* - display serial port error statistics

**SYNOPSIS**

*serstat*

**DESCRIPTION**

The *serstat* command reports error status information about groups of serial tty ports. The command supports IOP16 ports. The command does not currently support IOP and RIOP ports. When first invoked, *serstat* finds the four ports with the largest number of total errors logged and displays the logged errors.

The command then runs in “automatic” mode, in which it scans all serial ports for any change of status. As port status changes, *serstat* updates the display to ensure that the four ports with the largest number of errors logged are displayed at all times. Ports with fewer errors logged are replaced as other ports with more errors logged are displayed. A message at the bottom of the screen indicates which port has most recently changed.

The *serstat* program can also be run in “scan” mode and “continuous” mode. In scan mode, *serstat* scans sequential groups of ports every 3 seconds and displays the errors. In continuous mode, *serstat* continues to scan and update the currently-displayed ports only.

To exit *serstat*, generate a keyboard interrupt.

Once *serstat* is running, use any of the following one-character commands:

- r** Redraws the screen. No mode change.
- a** Redraws the screen. Starts automatic mode.
- m** Redraws the screen with ports having the most errors. Starts automatic mode.
- s** Redraws the screen. Starts scan mode.
- c** Redraws the screen. Starts continuous mode.

The program displays data from the following status structure maintained by the serial driver in the kernel:

```

struct sererrstat {
    uint    se_ttyhog;        /* tty input hog status achieved (ttin) */
    uint    se_iflushed;     /* hogs input queues discarded (ttin) */
    uint    se_idropped;    /* input char(s) dropped (ttin, serrint) */
    uint    se_norbuf;      /* no receive buffer available (serrint) */
    uint    se_othrottle;   /* output throttled, low clists (T_HIWATER) */
    uint    se_oflushed;    /* hogs output queue discarded (ttxput) */
}

```

```

uint    se_odropped;    /* output char(s) dropped (serxsend, sersend) */
uint    se_notbuf;      /* no transmit buffer available (ttout) */
uint    se_rxorun;     /* receiver overrun (serrint) */
uint    se_exstat;     /* external status change (sertint) */
uint    se_pe;         /* parity errors (serrint) */
uint    se_frame;     /* CRC/framing error (serrint) */
};

```

All fields are incremented once per event occurrence except **se\_idropped** and **se\_odropped**. These two fields try to keep track of the number of characters dropped for that particular error event instead of the number of times that error event occurred. Note that 256 characters or more can be lost when the input queue is flushed, but the only record of this event is a single increment to the **se\_iflushed** field.

The field **se\_exstat** counts the number of external status changes occurring on a port. A break condition or change in the Carrier Detect or Clear To Send lines increments this number. These are not normally error conditions, but may be of interest.

#### SEE ALSO

termio(7).

#### NOTE

This utility is intended for diagnostic use by qualified system administrators; it is not a basic user command.

#### BUGS

Scan mode displays ports that might not be present in the system.

If run on a system without an IOP or IOP16, **serstat** reports a read error and terminates.

## NAME

*tio* - tape io filter

## SYNOPSIS

```
tio -r tape_device [ -b blocksize ]
tio -w tape_device [ -b blocksize ]
```

## DESCRIPTION

*tio* reads from or writes to a tape device asynchronously, which results in high throughput tape streaming. If the **-r** option is used, *tio* reads from *tape\_device* and writes to standard output; if the **-w** option is used, *tio* reads from standard input and writes to *tape\_device*. The block size specified with **tio -r** must be the same as the block size specified with **tio -w** when the tape is made.

When end-of-tape is reached, *tio* prompts the user to choose between continuing (by inserting a new tape) or exiting. The user may select either the same tape device or a new tape device by pressing RETURN when the drive is ready.

*tio* accepts block sizes only in multiples of 256, and fills with NULLS any incomplete last block. The **-b** flag can be used to select a particular block size. The number specifying the block size can end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate multiplication. For example, 128 (incomplete block, filled to 256 bytes), 256w (512 bytes), 512x4 (2048 bytes), 64k (64\*1024 bytes), or 128b (128\*512 bytes). The default is 65536 bytes.

Although *tio* has been optimized to support tape streaming, the user may get only partial streaming, depending on the archiving software and tape drives used. For example, *cpio*(1) is usually too slow for *tio*, especially when there are a lot of small files. On the contrary, the Cipher 990 caching tape drive is too fast for *tio* to stream.

For the quarter-inch cartridge tape drive, the user can expect an increase in performance of about 100 percent using *cpio*(1) and *tio*. For the half-inch drives (Cipher 880), the user can expect a 50 percent performance gain if *cpio*(1) is used, and about a 200 percent performance gain if *dd*(1M) is used.

## EXAMPLES

```
find . -print | cpio -ocQ | tio -w /dev/rmt0
```

```
tio -r /dev/rmt0 | cpio -icQt
```

```
dd if=/dev/rdisk/c0d1s1 bs=10k | tio -w /dev/rmt/c1d0h
```

**TIO(1)**

**TIO(1)**

**FILES**

/dev/rmt\*, /dev/rmt/\*