Changes in 2.10BSD

April 20, 1987

Keith Bostic

Computer Systems Research Group
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720
seismo!keith; bostic@ucbvax.berkeley.edu

Casey Leedom

Department of Computer Science
California State University, Stanislaus
Turlock, California 95380
ucbvax!vangogh!casey; casey@vangogh.berkeley.edu

This document summarizes changes in PDP-11† UNIX‡ between the July 1983 2.9BSD release and the April 1987 2.10BSD distribution.

It is intended to provide sufficient information that those who maintain the kernel, have local modifications to install, or who have versions of 2.9BSD modified to run on other hardware should be able to determine how to integrate this version of the system into their environment. As always, the source code is the final source of information, and this document is intended primarily to point out those areas that have changed.

With the massive changes made to the system, both in organization and in content, it may take some time to understand how to carry over local software. The majority of this document is devoted to describing the contents of each important source file in the system. If you have local software to incorporate in the system you should first read this document completely, then study the source code to more fully understand the changes as they affect you.

Most of the changes between 2.9BSD and 2.10BSD fall into one of several categories. These are:

- bug fixes,
- performance improvements,
- addition of 4.3BSD system calls,
- removal of features no longer supported in the 4.3BSD release,
- new protocol and hardware support.

The major changes to the kernel are:

- the addition of networking,
- a complete rewrite of the user/kernel interface,
- the addition of numerous system calls,
- replacement of much of the high kernel with portions of the 4.3BSD kernel,

†DEC, PDP-11, QBUS, and UNIBUS are trademarks of Digital Equipment Corporation.
‡UNIX is a trademark of Bell Laboratories.

- the addition of the 4.3BSD tty and serial line drivers,
- the addition of inode, swap and text cacheing algorithms,
- restructuring the kernel into the 4.3BSD structure.

This document is not intended to be an introduction to the kernel, but assumes familiarity with prior versions of the kernel, particularly the 2.9BSD release. Other documents may be consulted for more complete discussions of the kernel and its subsystems. It cannot be too strongly emphasized that 2.10BSD much more closely resembles 4.3BSD than it does 2.9BSD. We have attempted to be, literally, bug-for-bug compatible with 4.3BSD. It is STRONGLY recommended that 4.3BSD manuals be consulted when using this system. 2.9BSD manuals are no longer correct. Online documentation is as complete and correct as time permitted.

This release is not supported, nor should it be considered an official Berkeley release. It was called 2.10BSD because 2.9BSD has clearly become overworked and System V was already taken. The "bugs" address supplied with this release (as well as with the 4BSD releases) will work for some unknown period of time; make sure that the "Index:" line of the bug report indicates that the release is "2.10BSD". See the *sendbug*(8) program for more details. All fixes that we make, or that are sent to us, will be posted on *USENET*, in the news group "comp.bugs.2bsd".

The authors gratefully acknowledge the contributions of many other people to the work described here. Major contributors include Gregory Travis and Jeff Johnson of the Institute for Social Research, and Steven Uitti of Purdue University. Cyrus Rahman of Duke University should hold some kind of record for being able to get the entire kernel rewritten with a single 10-line bug report. Much credit should also go to the authors of 4.2BSD and 4.3BSD from which we stole everything that wasn't nailed down and several things that were. (Just "diff" this document against *Changes to the Kernel in 4.2BSD* if you don't believe that!) We are also grateful for the invaluable guidance provided by Michael Karels, of the Computer Science Research Group, at Berkeley – although we felt that his suggestion that we "just buy a VAX", while perhaps more practical, was not entirely within the spirit of the project.

1. Organizational changes

The directory organization and file names are very different from 2.9BSD. The new directory layout breaks machine-specific and network-specific portions of the system out into separate directories. A new file, *machine*, is a symbolic link to a directory for the target machine, in this case, *pdp*. This allows a single set of sources to be shared between multiple machine types (by including header files as *../machine/file*). The directory naming conventions, as they relate to the network support, are intended to allow expansion in supporting multiple "protocol families". The following directories comprise the system sources for the PDP:

/sys/OTHERS	defunct PDP UNIBUS device drivers
/sys/autoconfig	PDP dependent autoconfiguration code
/sys/bootrom	PDP dependent boot rom code
/sys/conf	site configuration files and basic templates
/sys/h	machine independent include files
/sys/mdec	PDP dependent deadstart (block zero) code
/sys/net	network independent, but network related code
/sys/netimp	IMP support code
/sys/netinet	DARPA Internet code
/sys/netns	DARPA Internet code
/sys/netpup	PUP-1 support code
/sys/pdp	PDP specific mainline code
/sys/pdpdist	PDP distribution files
/sys/pdpif	PDP network interface code
/sys/pdpstand	PDP dependent stand-alone code
/sys/pdpuba	PDP UNIBUS device drivers and related code
/sys/pdpmba	PDP MASSBUS device drivers and related code
/sys/sys	machine independent system source files

/sys/vaxif	VAX network interface code
/sys/vaxuba	VAX UNIBUS device drivers and related code

Files indicated as *machine independent* are shared between all UNIX† systems. Files indicated as *machine dependent* are located in directories indicative of the machine on which they are used; the 2.10BSD release from Berkeley only contains support for the PDP. Files marked *network independent* form the "core" of the networking subsystem, and are shared among all network software; the 2.10BSD release only contains support for the DARPA Internet protocols IP, TCP, UDP, and ICMP.

1.1. /sys/h

Files residing here are intended to be machine independent. Consequently, the header files for device drivers, which were present in this directory in 2.9BSD, have been moved to other directories; e.g. */sys/pdpuba*, */sys/netinet*, etc. Many files which had been duplicated in */usr/include* are now present only in */sys/h*. Further, the 2.9BSD */usr/include/sys* directory is now, normally, a symbolic link to this directory. By having only a single copy of these files the "multiple update" problem no longer occurs. (It is still possible to have */usr/include/sys* be a copy of the */sys/h* for sites where it is not feasible to allow the general user community access to the system source code.) For further information, see the Makefile for *usr/src/include*.

In general, the include files for 2.10BSD have been totally reworked to be as close as possible to 4.3BSD. In many cases, they have even been renamed. This may cause problems when attempting to port local software, although it should facilitate porting software developed under any version of 4BSD.

The following files are new to /sys/h in 2.10BSD.

clist.h	contains the *cblock* structure definition previously found in tty.h
dk.h	contains the disk and tty monitoring information previously found in *systm.h*.
domain.h	describes the internal structure of a communications domain; part of the new ipc facilities
errno.h	had previously been only in */usr/include*; the file */usr/include/errno.h* is now a symbolic link to this file
exec.h	the kernel's definition of the exec structure, as well as the overlay structure and the magic number definitions
fs.h	replaces the old filsys.h description of the file system organization
gprof.h	describes various data structures used in profiling the kernel; see *gprof*(1) for details; not yet implemented under 2.10BSD
kernel.h	is an offshoot of systm.h and param.h; contains constants and definitions related to the logical UNIX "kernel"
mbuf.h	describes the memory management support used mostly by the networking; see "4.2BSD Networking Implementation Notes" for more information
msgbuf.h	describes the kernel message buffer
namei.h	defines various structures and manifest constants used in conjunctions with the *namei* routine
protosw.h	contains a description of the protocol switch table and related manifest constants and data structures use in communicating with routines located in the table
ptrace.h	contains definitions for process tracing
quota.h	contains definitions related to the 4.3BSD disk quota facilities, which are unimplemented in 2.10BSD. The file is included to allow easy porting of 4BSD programs.
resource.h	contains definitions used in the *getrusage*, *getrlimit*, and *getpriority* system calls (among others)

† UNIX is a trademark of Bell Laboratories.

socket.h contains user-visible definitions related to the new socket ipc facilities

socketvar.h contains implementation definitions for the new socket ipc facilities

syslog.h contains definitions for the system logging facility

tablet.h contains structures and definitions concerning the table line discipline

trace.h contains definitions and storage for file system buffer tracing points

ttychars.h contains definitions related to tty character handling; in particular, manifest constants for the system standard erase, kill, interrupt, quit, etc. characters are stored here (all the appropriate user programs use these manifest definitions)

ttydev.h contains definitions related to hardware specific portions of tty handling (such as baud rates); to be expanded in the future

uio.h contains definitions for users wishing to use the new scatter-gather i/o facilities; scatter-gather is unimplemented in 2.10BSD, although compatibility routines are available. These compatibility routines will behave exactly as a "correct" implementation would except for a few fairly unlikely situations.

un.h contains user-visible definitions related to the "unix" ipc domain

unpcb.h contains the definition of the protocol control block used in the "unix" ipc domain

vlimit.h contains definitions for kernel limits.

vm.h a quick way to include all of the vm header files

vmmac.h contains various macros concerning memory

vmmeter.h contains structures for monitoring various kernel functions which were in vm.h; several new items are metered: floating point simulation traps, number of calls to swapout and swapin (v_swpout/v_swpin); v_pgout/v_pgin count the number of I/O operations to and from the swap device (these use to be counted in v_swpout/v_swpin); synchronous software traps are no longer counted in v_intr; network soft interrupts are now counted in v_soft.

vmparam.h memory definitions

vmsystm.h various memory variables and structures

vtimes.h memory metering structures

wait.h contains definitions used in the *wait* and *wait3*(2) system calls; previously in */usr/include/wait.h*

The following files have undergone significant change:

buf.h several macros to replace rarely used routines and for dealing with buffer chains have been added. An new structure, *bufhd*, has been added. The *b_link* field has been removed from the buffer structure. Of particular interest may be a macro for the translation of a buffer address to a physical address. Many of the buffer flag values have been changed; note that the addition of any more flag values will require changing the flag word, *b_flags*, to a long.

callout.h the callout structure has been changed; kernel callouts are now implemented as a linked list

conf.h reflects changes made for the new *select*(2) system call; the character device table has a new member, *d_select*, which is passed a *dev_t* value and an FREAD (FWRITE) flag and returns 1 when data may be read (written), and 0 otherwise. The *bdevsw* structure has lost the *d_tab* entry and gained a flags field, *d_flags*. A new structure has also been added, *linesw*, to support the use of various line protocols.

dir.h is completely different since definitions for the user level interface routines described in *directory*(3) are included

file.h has a very different *file* structure definition and definitions for the new *open* and *flock* system calls; the symbolic definitions for many constants commonly supplied to *access* and

lseek, have been changed since 2.9BSD to bring them into line with 4.3BSD. A new member has been added, *f_type*. This member indicates the type of structure pointed to by *f_data*, and should be either DTYPE_INODE, DTYPE_PIPE, or DTYPE_SOCKET, constants found in *h/file.h*.

inode.h some of the standard macros have been renamed; the in-core inode structure has been rearranged to take advantage of areas of commonality between it and the dinode structure. A complete set of definitions for accessing various sub-fields have been added, and explicit references in the kernel have been removed. The access, modification and status change times have been added to the incore inode structure, as well as new fields for the advisory locking facilities and a back pointer to the file system super block information. A pointer to a text object has also been added. The dinode structure is now defined here. Also, various definitions for the *open*(2), *seek*(2), and *access*(2) calls have been added and/or changed.

ioctl.h has all request codes constructed from _IO, _IOR, _IOW, and _IOWR macros which encode whether the request requires data copied in, out, or in and out of the kernel address space; the size of the data parameter (in bytes) is also encoded in the request, allowing the *ioctl*() routine to perform all user-kernel address space copies

localopts.h is now generated by the script *config*, which is found in the *sys/conf* directory; it contains far fewer values than it did before, mostly as a result of the removal of most of the user defined options from 2.9BSD, as well as containing a few new ones.

param.h has had numerous items deleted from it; in particular, many definitions logically part of the "kernel" have been moved to *kernel.h*, and machine-dependent values and definitions previously found in *param.h* have been moved to *machine/machparam.h*; it now contains a manifest constant, NGROUPS, which defines the maximum size of the group access list.

proc.h has changed extensively as a result of the new signals, the different resource usage structure, and the new timers; in addition, new members are present to support the status information required by new system calls as well as to provide the linked lists and hashing required by new access methods.

signal.h reflects the new signal facilities; several new signals have been added; structures used in the *sigvec*(2) and *sigstack*(2) system calls, as well as signal handler invocations are defined here

stat.h has been updated to reflect the changes to the inode structure; in addition several new fields have been added. One of those fields, *st_block*, claims to be the number of blocks used by the file. It isn't really, it's the size of the file in bytes rounded up to a kilobyte boundary. This results in some programs, notably *du*(1), believing ridiculous file sizes.

systm.h has been trimmed back a bit as various items were moved to *kernel.h* and other include files

text.h two pointers have been added to the text structure to support LRU cacheing of text objects. The reference and loaded reference counters are now unsigned values.

time.h contains the definitions for the new time and interval timer facilities

tty.h reflects changes made to the internal structure of the terminal handler

types.h reflects several new types, and has been restructured

user.h has been extensively modified; members have been grouped and categorized to reflect the "4.2BSD System Manual" presentation; new members have been added and existing members changed to reflect the new groups facilities, changes to resource accounting and limiting, new timer facilities, and new signal facilities

1.2. /sys/sys

This directory contains the "mainstream" kernel code. Files in this directory are intended to be shared between 2.10BSD implementations on all machines. As there is little correspondence between the current files in this directory and those which were present in 2.9BSD, a general overview of each files's contents will be presented rather than a file-by-file comparison. This is also where we will discuss some of the major changes that have been made to various areas of the kernel.

Files in the *sys* directory are named with prefixes which indicate their placement in the internal system layering. The following table summarizes these naming conventions.

init_	system initialization
kern_	kernel (authentication, process management, etc.)
quota_	disk quotas
sys_	system calls and similar
tty_	terminal handling
ufs_	file system
uipc_	interprocess communication
vm_	memory

1.2.1. Initialization code

init_main.c contains system startup code

init_sysent.c contains the definition of the *sysent* table – the table of system calls supported by 2.10BSD.

1.2.2. Kernel-level support

kern_acct.c contains code used in per-process accounting

kern_clock.c contains code for clock processing; work was done here to minimize time spent in the *hardclock* routine.

kern_descrip.c contains code for management of descriptors; descriptor related system calls such as *dup* and *close* (the upper-most levels) are present here

kern_exec.c contains code for the *exec* system call. Argument and environment copies in exec are now done on a per-string rather than a per-character basis.

kern_exit.c contains code for the *exit* and *wait* system calls

kern_fork.c contains code for the *fork* and *vfork* system calls. The system now keeps a range of pids it can directly assign to new processes.

kern_mman.c contains code for memory management related calls

kern_proc.c contains code related to process management; in particular, support routines for process groups

kern_prot.c contains code related to access control and protection; the notions of user ID, group ID, and the group access list are implemented here

kern_resrce.c code related to resource accounting and limits; the *getrusage* and "get" and "set" resource limit system calls are found here

kern_rtp.c code supporting the "real-time" processing features of 2.10BSD; these features are essentially untested since 2.9BSD.

kern_sig.c the signal facilities; in particular, kernel level routines for posting and processing signals

kern_subr.c support routines for manipulating the kernel I/O structure: *uiomove*, *uiofmove*, *ureadc*, and *uwritec*

kern_synch.c code related to process synchronization and scheduling: *sleep* and *wakeup* among others. The scheduler no longer scans the process table once per second, it only considers runnable processes when recomputing priorities. Sleeping processes get their priority bump

when they get put back on the run queue.

kern_time.c code related to processing time, the handling of interval timers and time of day clock. Kernel timeouts are now implemented as a linked list. There are several minor differences in the treatment of the timers in 2.10BSD. All real-time timer calls are to one second resolution, not to a clock tick as offered in 4.3BSD. (Incidentally, while the *ualarm*(3) and *usleep*(3) calls are available, this one second resolution makes them useless on the 2.10BSD.) We also compute the real-time timer for all processes at the same time, as part of the cpu scheduler, rather than as a timeout, as is done in 4.3BSD. This was done to allow the timeout offset, *c_time*, to be stored in the kernel as a int, instead of a long, minimizing the number of long operations done at clock interrupt. If the kernel ever requires a timeout of greater than 9 minutes, it should be converted to a long, at which point the real-time timer calls should be done as timeouts. The other timers, which are necessarily processed at clock interrupt, are stored internally as ticks, not as seconds, but only one second resolution is offered by the system call interface to be consistent with the real time timer. The RLIMIT_CPU is also stored internally as clock ticks. The only noticeable effect of this is that very large values supplied with the RLIMIT_CPU option of *getrlimit*(2) and *setrlimit*(2) will automatically be converted to RLIM_INFINITY since the conversion of seconds to ticks would cause overflow. Also, *adjtime*(2) calls are executed immediately, not over a period of time, therefore, the *olddelta* return values for an *adjtime*(2) call will always be zero.

kern_xxx.c miscellaneous system facilities

syscalls.c list of available system calls

1.2.3. Disk quotas

quota_sys.c disk quota system call routines; quotas are unimplemented in 2.10BSD.

1.2.4. General subroutines

subr_prf.c *printf* and friends; also, code related to handling of the diagnostic message buffer

subr_rmap.c subroutines which manage resource maps. The kernel routines *malloc*(), *mfree*(), and *malloc3*() have been redone to various degrees for speed.

subr_xxx.c miscellaneous routines, including the *nonet*(2) routine. This routine has been supplied as a stub for the routines *socket*(2) and *socketpair*(2) on non-networking systems so that networking programs can behave rationally.

1.2.5. System level support

sys_generic.c code for the upper-most levels of the "generic" system calls: *read*, *write*, *ioctl*, and *select*; a "must read" file for the system guru trying to shake out 2.9BSD bad habits.

sys_inode.c code supporting the "generic" system calls of sys_generic.c as they apply to inodes; the guts of the byte stream file i/o interface. Inode locking, (the *flock*(2) call) is also implemented here.

sys_process.c code related to process tracing

sys_socket.c code supporting the "generic" system calls of sys_generic.c as they apply to sockets

1.2.6. Terminal handling

tty.c the terminal handler proper; both 2.9BSD and version 7 terminal interfaces have been merged into a single set of routines which are selected as line disciplines

tty_conf.c initialized data structures related to terminal handling;

tty_pty.c support for pseudo-terminals; actually two device drivers in one

tty_subr.c c-list support routines; these have been completely rewritten to be much faster, particularly in moving groups of characters around. The definition *CBSIZE* has also been

doubled, to 32, while the number of clists (*NCLISTS*) has been cut by about a third. The number of clists is also relative to the constant *MAXUSERS*, rather than being the same for all systems.

tty_tb.c two line disciplines for supporting RS232 interfaces to Genisco and Hitachi tablets. Untested in 2.10BSD.

tty_tty.c trivial support routines for "/dev/tty"

1.2.7. File system support

ufs_alloc.c code which handles allocation and deallocation of file system related resources: disk blocks, on-disk inodes, etc. 2.9BSD had a real problem here in that it did a linear search of the mount table every time it freed or allocated a block. The addition of a back-pointer to the associated file system to the incore inode eliminated this lookup.

ufs_bio.c block i/o support, the buffer cache proper. The 2.9BSD block i/o code has been completely ripped out and replaced with 4.3BSD code. The most visible effects of this were the removal of the *b_link* field from the buffer header structure, the removal of the *bunhash*() subroutine, and the reorganizing of the free queues into four separate queus. This will have some interesting side effects if you have local drivers that use the *b_link* field, or if you have drivers that misuse some of the remaining pointers in the buffer header structure. Note, the correct fix is not to put the *b_link* field back in, but to rewrite the driver's queueing mechanism. All current drivers have also been rewritten to supply the *BYTE* or *WORD* argument directly to *physio*(), eliminating the routine *physio1*().

ufs_bmap.c code which handles logical file system to logical disk block number mapping; understands structure of indirect blocks and files with holes; handles automatic extension of files on write

ufs_dsort.c sort routine implementing a prioritized seek sort algorithm for disk i/o operations

ufs_fio.c code handling file system specific issues of access control and protection

ufs_inode.c inode management routines; in-core inodes are now hashed and retained in an LRU cache, so that inodes do not have to be continually re-read from disk. The in-core inode also contains the access, modification, and creation times which make *stat*(2) calls much faster. There is an interesting relationship between texts and inodes, now, in that the number of inodes available for normal system use will be the number of inodes minus the number of text images cached by the system. Also, the inode now has a pointer to its text object, if any, which is used to implement shared text images.

ufs_mount.c code related to demountable file systems

ufs_nami.c the *namei* routine (and related support routines) – the routine that maps pathnames to inode numbers. The 2.9BSD *namei*() routine has been completely reworked. There are three major advantages to the new one: first, it reads the entire path in from user space at once rather than using multiple subroutine calls per character, second, it caches the current position in the directory so that each subsequent request from a process will continue searching at the place it stopped, and finally, it handles symbolic links correctly. *Namei*, and its associated routines, probably gave us more long-term trouble than any other part of the kernel. Be Very Careful if you modify anything here, especially in terms of side-effects for *mkdir*(2), *rmdir*(2), and *rename*(2).

ufs_subr.c miscellaneous subroutines

ufs_syscalls.c file system related system calls, everything from *open* to *unlink*; many new system calls are found here: *rename*, *mkdir*, *rmdir*, *truncate*, etc.

1.2.8. Interprocess communication

proto.c protocol configuration table and associated routines

sys_pipe.c the pipe open, read, and write routines

uipc_domain.c code implementing the "communication domain" concept; this file must be augmented to incorporate new domains

uipc_mbuf.c memory management routines for the ipc and network facilities; refer to the document "4.2BSD Networking Implementation Notes" for a detailed description of the routines in this file

mbuf11.c the same as above, only more PDP-11 dependent.

uipc_proto.c UNIX ipc communication domain configuration definitions; contains UNIX domain data structure initialization

uipc_socket.c top level socket support routines; these routines handle the interface to the protocol request routines, move data between user address space and socket data queues and understand the majority of the logic in process synchronization as it relates to the ipc facilities

uipc_socket2.c lower level socket support routines; provide nitty gritty bit twiddling of socket data structures; manage placement of data on socket data queues

uipc_syscalls.c user interface code to ipc system calls: *socket*, *bind*, *connect*, *accept*, etc.; concerned exclusively with system call argument passing and validation

uipc_usrreq.c UNIX ipc domain support; user request routine and supporting utility routines

1.2.9. Memory support

The code in the memory subsystem has changed very little from 2.9BSD; most changes made in these files were either to gain speed or to fix bugs.

vm_proc.c mainly code to manage memory allocation during process creation and destruction.

vm_sched.c the code for process 0, the scheduler, lives here; other routines which monitor and meter memory activity (used in implementing high level scheduling policies) also are present; the scheduler has been perturbed in minor ways to encourage the swapping of processes that have been in a sleep state for a significant amount of time.

vm_swap.c code to swap a process in or out of core

vm_swp.c code to handle swap i/o

vm_text.c code to handle shared text segments – the "text" table; this code has been completely redone to allow the cacheing of text images. The current system will maintain the core image of a process until the core or swap is required for another process. Neither will a swap image of a process be written until the image is forced out of core. Basically, all programs are now treated as "sticky". A new routine, *xuncore*(), has been added that frees up core space for the map allocation routines.

1.3. /sys/conf

This directory contains files used in configuring systems. The format of configuration files has changed a great deal.

:comm-to-bss script to move the *proc*, *inode* and *file* tables from comm space to bss.

:splfix.* scripts to implement the spl functions for various PDP-11 instruction sets.

GENERIC the generic configuration file; contains the standard user tuneable system parameters.

Make* the system Makefiles; they've been broken up into six parts since a single one cannot handle the dependencies generated for the kernel.

VAX.compile a directory containing a C preprocessor and C compiler that allows 2.10BSD to be compiled on another machine. If you're interested, the entire networking kernel compiles from scratch in about 5 minutes on a VAX 8600.

checksys.c source for the kernel size checking program

config the actual kernel configuration script

ioconf.c I/O configuration file

param.c various kernel parameters and tables

1.4. /sys/pdpuba

This directory contains UNIBUS device drivers, and their related include and autoconfiguration files. The include files have moved from */sys/h* and the autoconfiguration files from *sys/autoconfig* in an effort to isolate machine-dependent portions of the system. The following device drivers were not present in the 2.9BSD release.

ram.c a ram disk driver

rx.c a driver for the RX01/02 floppy disk

si.c a driver for the SI 9500 controller with CDC 9766 disks

ra.c a driver for the UDA controller

In addition to the above device drivers, many drivers present in 2.10BSD now sport corresponding include files which contain device register definitions. For example, the DH11 driver is now broken into three files: dh.c, dhreg.h, and dmreg.h.

The following device drivers have been significantly modified, or had bugs fixed in them, since the 2.9BSD release:

dh.c the 4.3BSD driver has been installed

dz.c the 4.3BSD driver has been installed

dhu.c the 4.3BSD driver has been installed

lpr.c the line printer driver has had some per-character i/o removed

xp.c the bad blocking has been fixed

Additionally, the *rl*, *ra* and *xp* drivers support the 22-bit QBUS. The tty drivers also do intelligent auto siloing, switching between siloing and interrupt per character based on input load.

In addition the *sys/OTHERS* directory has had several "new" device drivers added to it that may or may not work. It is extremely probable that they do not handle the new ioctl protocols and it is also likely that they do not correctly map buffers in and out of kernel space correctly. For more information regarding the installation of device drivers, see the file *sys/OTHERS/README*. This is a rambling, disjointed "must" for the driver hacker. You should also see this directory if you are having problems with a driver that's currently in place in 2.10BSD. There are several different versions of drivers, bug fixes etc. that we just didn't have time to install and/or test out. A great deal of work has been done on the *ra* and *xp* drivers. They are known to work, and work reliably. You should use them, if at all possible.

1.5. /sys/pdpmba

1.6. /sys/pdp

The following files are new in 2.10BSD:

clock.c the clock start and busy-delay routines

cons.c the console driver, originally *sys/dev/kl.c*

frame.h the PDP-11 calling frame definition

genassym.c a program to generate structure offset definitions for the assembly files, originally *sys/conf/genassym.c*

in_cksum.c checksum routine for the DARPA Internet protocols

kern_pdp.c PDP-11 dependent system calls

machparam.h machine-dependent portion of *sys/h/param.h*

mem.c the memory device driver

mscp.h definitions for the Mass Storage Control Protocol

psl.h definitions for the PDP-11 program status register

reg.h definitions for locating the users' registers, relative to R0 in the user call frame on the kernel stack

scb.s system control block file, containing the low core vector interrupt definitions, originally *confil.s*.

vmparam.h machine-dependent memory portion of *sys/h/param.h*

The file *mch.s* has been split up into several different files. The following files all contain code originally part of *mch.s*. The *libc_* files all contain code duplicated in the C library.

libc_bcmp.s the bcmp routine

libc_bcopy.s the bcopy routine

libc_bzero.s the bzero routine

libc_csv.s the csave and cretum routines

libc_ffs.s the find first set routine

libc_htonl.s the host to network long routine

libc_htons.s the host to network short routine

libc_insque.s the insert queue routine

libc_ldiv.s the long division routine

libc_lmul.s the long multiplication routine

libc_lrem.s the long remainder routine

libc_remque.s the remove queue routine

libc_strlen.s the string length routine

mch_backup.s routines to back up the user CPU state after an aborted instruction

mch_click.s routines to move click size areas

mch_copy.s routines to move various data sizes to and from user space

mch_dump.s the automatic tape dump routines

mch_dzpdma.s the DZ-11 pseudo-DMA interrupt routines

mch_profile.s system profiling routines; these are untested since 2.9BSD.

mch_start.s checkout, setup and startup routines

mch_trap.s interrupt interface code to C.

mch_vars.s variable definition code

mch_xxx.s various other routines that needed to be in assembly

DEFS.h definitions and common macros for all assembly files; emulates the C library DEFS.h file for the benefit of the *libc_* files.

1.7. /sys/autoconfig

The autoconfiguration directory is mostly as it was in 2.9BSD with the exception that the probe routines for the various devices have been broken out into either *sys/pdpuba* or one of the networking directories. To find a probe routine, look for a file named *xxauto*(), where xx is the standard device abbreviation.

2. Changes inside the kernel

1) Code which previously used the *iomove*, *passc*, or *cpass* routines will have to be modified to use the new *uiomove*, *ureadc*, and *uwritec* routines. These routines are faster than the 2.9BSD versions as well as having the added facilities of mapping in user space for large transfers and transferring odd lengths of data or data to odd addresses without the serious performance degradation seen in 2.9BSD.

2) The calling convention for the driver *ioctl* routine has changed. Any data copied in or out of the system is now done at the highest level, inside *ioctl*(). The third parameter to the driver *ioctl* routine is a data buffer passed by reference. Values to be returned by a driver must be copied into the associated buffer from which the system then copies into the user address space. It should also be noted that *ioctl*'s are 32-bit quantities, not 16-bit as in 2.9BSD, although the kernel still stores them in 16 bits internally. These changes are reflected in *ioctl.h*. Also, the *read*, *write*, and *ioctl* entry points in device drivers must return 0 or an error code from *<errno.h>*. This is a reflection of the effort to have kernel routines return errors rather than setting global variables to error values. When porting your local device drivers make sure that they correctly return these errors.

3) Assembly routines which make system calls will not port from 2.9BSD or earlier systems to 2.10BSD, as the system call protocol to the kernel has changed a great deal. Arguments are always passed into the kernel on the stack, as opposed to registers and/or data space. This change was made for a number of reasons, and triggered by the need for reliable signal delivery. The file *usr/include/syscall.h* details the new entrance values. The 2.9BSD protocol of using the high bit of the syscall value to indicate that the arguments are on the stack has also been removed. The *syslocal* (58) call has disappeared entirely as the *syslocal* table has been folded into the *sysent* table. The *indir* (0) call has also been deleted. If you have any assembly source which makes system calls directly, it is strongly recommended that you convert them to simply push their parameters onto the stack and call the standard library routines. **Obviously, there is no binary compatibility with previous versions of PDP UNIX.**

   There have been many changes in the system calls available to user programs, as approximately thirty new system calls have been added and five or so deleted. Use of a deleted system call is fairly easy to detect since the kernel delivers a SIGSYS signal to any program requesting a non-existent system call. The one exception to this rule is the use of a *socket*(2) or *socketpair*(2) call on a system without networking. Either of these calls will simply return an error so that networking programs can fail correctly.

4) On the PDP-11 only some of the *rusage* and *itimerval* structures are meaningful. In order to minimize the data space required by the user and proc structures in the kernel, the kernel has private copies of these structures. Only programs such as *ps*(1) and *vmstat*(1), that read through the proc structure, will need to know about the kernel's versions of these structures. Any unused members of these structures are zero'd out before being returned to a user process.

5) Both the kernel and application systems may now have up to NOVL (15) overlays. This feature requires modification of *ld(1)*, *adb(1)*, the C library, the kernel, *strip*(1), and *nm*(1) among other things. If you're not hearing what I'm saying, don't mess with it. Separate libraries are no longer required for overlaid and non-overlaid programs. It turned out that it cost approximately 3.89 micro-seconds per subroutine call on a PDP 11/44 to combine the two. Therefore, all programs should use the standard library (libc.a) for loading. The -V flag is ignored by *cc*(1) and *f77*(1); they automagically pass it to the assembler and the various passes of the C compiler. (-V wasn't hardwired into the assembler so that it would still be possible to run *as*(1) without the -V flag and get an executable without having to use the loader.) If you call the assembler directly, remember to pass it the -V flag. Also, the C library should not be loaded as an overlay, as many of its routines don't use the csv/cret protocol.

6) The process table has grown considerably since its original incarnation. Several parts of the kernel used a linear search of the entire table to locate a process, a group of processes, or group of processes in a certain state. In order to reduce the time spent examining the process table, several changes have been made. The first is to place all process table entries onto one of three linked lists, one each

for entries in use by existing processes (*allproc*), entries for zombie processes (*zombproc*), and free entries (*freeproc*). This allows the scheduler and other facilities that must examine all existing processes to limit their search to those entries actually in use. Searches for unused process id's are avoided by noting a range of usable process ID's when trying to generate a new, unique ID. Searches for individual processes are avoided by using linked lists of hashed pid's.

7) In order to avoid searching through the list of open files when the actual number in use is usually small, the index of the last used open file slot is maintained in the field *u.u_lastfile*. The routines that implement open and close or implicit close (*exit* and *exec*) maintain this field, and it is used whenever the open file array *u.u_ofile* is scanned.

8) The values for *nice* used in 2.9BSD and previous systems ranged from 0 though 39. Each use of these scheduling parameters was offset by the default, NZERO (20). This has been changed in 2.10BSD to use a range of -20 to 20, with NZERO redefined as zero.

9) There have been an large number of changes in the types defined and used by the system. The system file *sys/types.h* should be serve as the final description of these new types. Of particular interest are the new typedefs that have been added for user ID's and group ID's in the kernel and common data structures. These typedefs, *uid_t* and *gid_t*, are both of type *u_short*. This change from the previous usage (explicit *short* ints) allows user and group ID's greater than 32767 to work reasonably. Other changes include the conversion of the *label_t* type to a structure, the addition of a *quad* (64-bit) type, and the much more extensive use of such types as *daddr_t*, *memaddr*, *off_t*, and *caddr_t*.

10) The berknet driver didn't work in 2.9BSD, and we've done nothing to change that. In any case, it's been moved to the *sys/OTHERS* directory with all of the other defunct drivers. We have provided a version of the 4.3BSD driver since it will understand the other changes to the kernel made for 2.10BSD, although it won't work either. See the README. Enjoy.

11) The routines *schar* and *uchar* have gone away and have been replaced by *copyinstr* and *copyoutstr*, routines that copy a NULL terminated string of characters in or out of kernel space. Also available are *vcopyin*, *vcopyout*, and *copystr*. The first two routines copy odd numbers of bytes and data to and from odd addresses efficiently. *Copystr* copies a NULL terminated string of characters within kernel space.

12) Directories with more than 10 trailing empty slots will be automagically truncated whenever a new file is created within them. This creates a minor problem: if someone creates an entry in the *lost+found* directory, that directory will be cheerfully truncated to a very small length. Since we didn't have the time to really fix the problem, we changed *mklost+found*(8) to leave a special file in the very last slot of the *lost+found* directory. It would probably be a good idea for you to remake your *lost+found* directories, just in case. Of course, the correct fix is to port the 4.3BSD fsck; I'd really like a copy when you're done.

13) The "real-time" features of 2.9BSD have been left in place, and, should work as they always have, with one major exception. The *fmove*() routine has not been fixed to be interruptible. See the *copy*() routine for examples of what needs to be done to make it behave correctly. This, however, will be fairly difficult. We suggest that if you want to use CGL_RTP that you comment out the use of *uiofmove*() in *uiomove*().

18 Most of the conditional compilation defines in the 2.9BSD kernel have been removed because the features they controlled are now either standard or no longer supported in 4.3BSD. Other features have been grouped together and are now controlled by the same define.

The following *#defines* are new or remain largely unchanged since 2.9BSD. Note, no software outside of the kernel needs to be aware of the "VIRUS_VFORK" flag. The kernel automatically maps *vfork(2)* into *fork*(2) if the *vfork* call isn't available.

define name	feature	comment
CGL_RTP	support one "real-time" process	untested
DIAGNOSTIC	more stringent error checking	now more expensive to run
ENABLE34	Able's MM board	untested in quite some time
HZ	line clock frequency	renamed "LINEHZ"
MAXMEM	limit process memory	
NOKA5	only buffers and clists use KA5	
Q22	22-bit QBUS	
SMALL	smaller queues and hash tables	untested in quite some time
UCB_CLIST	map out clists	
UCB_FRCSWAP	force swap on forks/expands	
UCB_METER	various system statistics	expanded; expensive to run
UCB_NET	Berkeley's TCP/IP	a 4.3BSD interface
UCB_RUSAGE	resource accounting	
UNIBUS_MAP	18-bit Unibus mapping	
VIRUS_VFORK	the *vfork*(2) system call	

The following table lists *#defines* that are now a standard part of 2.10BSD.

define name	feature	comment
ACCT	keep accounting records	
DZ_PDMA	pseudo DMA for DZ boards	
INSECURE	retain set-id bits	now follows 4.3BSD behavior
MENLO_OVLY	support user text overlays	
MENLO_JCL	support job control	
MENLO_KOV	support kernel overlays	
OLDTTY	V7 tty line discipline	part of the 4.3BSD tty driver
PARITY	handle cache and memory parity traps	
TM_IOCTL	ioctl calls in the TM-11 driver	
TS_IOCTL	ioctl calls in the TS-11 driver	
UCB_AUTOBOOT	allow automatic rebooting	
UCB_BHASH	hash buffer headers	
UCB_DEVERR	translate device error messages	
UCB_ECC	correct soft ecc disk transfer errors	
UCB_FSFIX	update files in correct order	
UCB_IHASH	hash in-core inodes	
UCB_LOAD	compute Tenex style load average	
UCB_NKB	set file system block size	always set to "1"
UCB_NTTY	Berkeley line discipline	part of the 4.3BSD tty driver
UCB_RENICE	support *renice*(2) system call	
UCB_SCRIPT	scripts may specify interpreters	
UCB_SYMLINKS	support symbolic links	
UCB_UPRINTF	send error messages to users	
UCB_VHANGUP	revoke access to tty after logout	

The following table lists *#defines* that have been removed from 2.10BSD.

define name	feature	comment
DISKMON	keep statistics on the buffer cache	absorbed by UCB_METER
INTRLVE	interleave file systems across devices	not supported in 4.3BSD
MPX_FILS	multiplexed files	not supported in 4.3BSD
UCB_GRPMAST	"group" super-users	not supported in 4.3BSD
UCB_LOGIN	"login" system call	not supported in 4.3BSD
UCB_PGRP	V7 bug fix	supplanted by job-control
UCB_SUBM	"submit" system call	not supported in 4.3BSD
UNFAST	turn off inline macros	
UCB_QUOTAS	dynamic disk quota scheme	
TEXAS_AUTOBAUD	getty feature	part of ported 4.3BSD source

3. Changes in application programs.

The application directories have also been rearranged in an effort to mimic the 4.3BSD release. The following table indicates the old and new names as well of the functions of these directories. For more information, see the UNIX manual page for *hier*(7).

new name	old name	purpose
bin	cmd	commands found in /bin
etc	cmd	commands found in /etc
games	games	commands found in /usr/games
include	include	files found in /usr/include
lib	lib	commands found in /lib
local	local	commands found in /local
new	contrib	commands found in /usr/new
old	N/A	commands found in /usr/old
ucb	ucb	commands found in /usr/ucb
usr.bin	cmd	commands found in /usr/bin
usr.lib	lib	commands found in /usr/lib

Many new commands have been added, and several have been removed; whenever possible we've dropped the 4.3BSD source directories in place. As in the kernel, 4.3BSD support was usually the deciding factor as to a program's status. In many of the source directories there are either or both of the special directories "PORT" and "OLD". PORT contains copies of 4.3BSD source that we feel ought to be ported to 2.10BSD. OLD contains copies of 2.9BSD source that, while we have ported the 4.3BSD version of the source code, we're unsure enough of it that we wanted to provide a backup copy. The following paragraphs are a description of several things that have changed outside of the kernel.

1) Since the 2.10BSD C compiler still only recognizes seven significant characters in external names, several standard library names had to be changed to prevent name collisions. However, to prevent portability problems in your programs you should use the standard names. All known collisions in the standard include files or the C library have been handled either in the include file itself or in the include file *include/short_names.h*. This works because we're using the 4.3BSD C preprocessor, which has flexnames. Networking programs almost always need to include *short_names.h*. See *src/bin/mail.c* and *src/bin/passwd.c* for examples of long name work arounds. The C library itself is a port of the 4.3BSD C library.

2) Files ported from 4BSD systems that have more than MAXNAMLEN characters will port correctly as long as they are unique in those characters. For example, the kernel will translate an open call of the form:

```
open("/usr/man/man1/file_with_too_long_a_name",O_RDONLY,0);
```

as if it were:

```
open("/usr/man/man1/file_with_too_",O_RDONLY,0);
```

2) The new directory reading routines are now part of stdio. We found that it was usually easier to put the directory reading routines into the code than to handle both the old and the new structures. We haven't changed the actual file system. If you *have* to have the old directory stuff, i.e. you are reading the actual device, convert your code to use the following structure, also found in *h/dir.h*.

```
#define  MAXNAMLEN 14

struct   v7direct  {
        ino_t    d_ino;
        char     d_name[MAXNAMLEN];
};
```

This allows access to both the old and new directory structures. Make sure that you change all of the references, e.g. "struct direct" should be "struct v7direct". *Fsck*(8) and *dump*(8) are good examples of programs that have to do this. Since the define "DIRSIZ" has changed in 4BSD (in V7 it was 14, the length of a file/directory name) we've replaced all occurrences of "DIRSIZ" in 2.10BSD with the corresponding 4BSD define, "MAXNAMLEN".

3) The 2.10BSD stdio allocates space for file I/O buffering differently than 2.9BSD did. If you have programs that use buffered I/O and are complaining about bad input, or just dying horribly, and the input looks okay, start looking at the way it handles *sbrk*(2). If it calls *sbrk* with an argument of 0 and then uses that offset to figure out where the rest of its allocated objects are, you've probably found the problem. A simple work-around is to allocate the necessary stdio buffers as part of the program's data space and then use *setbuf*(3) to so inform stdio. See the C compiler for examples.

4) The 4.3BSD manual pages for *sigblock*(2), *sigpause*(2), and *sigsetmask*(2) are deceptive. They indicate that signal masks are integers, but, they must be able to hold 32 bits. Typically you'll see code like:

```
int omask;
omask = sigsetmask(0);
...
sigsetmask(omask);
```

which should be translated to:

```
long omask;
omask = sigsetmask(0L);
...
sigsetmask(omask);
```

In general, the fact that 4.3BSD thinks an "int" is 32 bits is the worst porting problem that you'll run into; finding "ints" that should be "longs" is an art. Routines that we look for as a matter of habit are any one of the *seek*(2) routines, *ftell*(3), *time*(2), the various *signal*(2) routines and the *truncate*(2) routines.

The functions *sigblock* and *sigsetmask* are defined as returning a long result in *h/signal.h*, which should ease some of the porting problems. The lint libraries have also been updated. In general, though, you'll have to scan any source you plan on porting for calls to *sigblock*, *sigpause*, or *sigsetmask* that take an int as a parameter or store their return value in an int.

To give an indication of the subtlety the *long/int* problem can take on, consider the following code fragment taken from /sys/sys/tty.c:

```
newflags = (tp->t_flags&0xffff0000) | (sg->sg_flags&0xffff));
```

where *newflags* and the fields *t_flags* and *sg_flags* are all longs. The problem occurs with "sg->sg_flags&0xffff". The intent is fairly obvious, but in long op int the int ("0xffff") is sign-extended to long ("0xffffffff") before the operation as per K&R. The resulting operation in this case is a no-op! The fix is fairly simple in this instance and yields the following:

```
newflags = (tp->t_flags&0xffff0000) | (sg->sg_flags&0xffffL));
```

This particular type of low word masking bit us no less than 33 times in the kernel. Other possibilities that are even more annoying:

```
#define  FLAG1    0x00000001
...
#define  FLAG16   0x00008000
...
#define  FLAG32   0x80000000
```

Using any of the above definitions to test or set flag values in a long quantity will work, except **FLAG16**.

5) The PDP-11 *setjmp*(2) implementation contains a subtle bug that occurs when a routine containing a *setjmp* has *register* variables. The bug sometimes causes those variables to be given invalid values when a longjmp is made back to the routine. This is probably impossible to fix in a reasonable manner, and it's much simpler to simply avoid register variables in such routines.

6) The optional '#' character is not supported by *printf*(3).

7) The DEC MXV11 bootstrap ROM, for the RL's, TU's, RX's, and RD's among others, *requires* that deadstart blocks begin with an 0240 and a branch. Noone seems to know why. This has already been implemented in *rauboot.s* and *rluboot.s* in *sys/mdec*.

8) To port the 4.3BSD *make*(1) program, several of its table sizes had to be reduced. Make is very unforgiving of Makefiles that are too large. If make drops core for no reason that you can think of, try reducing the size of the Makefile. Also, don't run make depend in the system application directories, make can't handle it.

9) Don't set csh history too high; it eventually runs out of space and logs you out.

10) The games directory is largely untested.

11) There's a new (and wonderful) compiler, that actually handles bit fields, identically named global structure elements, and lots of other stuff. It *doesn't* handle any of the old assignment operators, and, not only doesn't handle them, but produces bogus code. It is STRONGLY recommended that you read the file *src/lib/ccom/TEST/README*. It goes into the problems with this compiler in more depth and contains some other Extremely Important Information.

12) The 4.3BSD loader uses the *-L* flag to specify a list of library directories that should be searched for libraries specified with the *-l* option. The old meaning of the *-L* flag, the termination of overlaid text, is now the function of the *-Y* flag.

13 *readv*(2) and *writev*(2) are implemented as compatibility routines in the standard library and are semantically identical to the 4.3BSD *readv* and *writev* on all current descriptors, with the exception of sockets (SOCK_STREAM does work identically however). In other words, the fact that the compatibility routines implement *readv* and *writev* non-atomically as multiple *read*(2) and *write*(2) calls doesn't normally matter; disk files don't care, tape reads and writes via read/writev under 4.3BSD don't work atomically anyway, and the compatibility code for *readv*'s on a tty seems to work as advertised.

14) Any limit other than RLIMIT_CPU, RLIMIT_CORE, or RLIMIT_FSIZE will be ignored when passed to the kernel by *setrlimit*. A subsequent call by *getrlimit* will return the correct value, however. The three limits listed above are fully implemented in the kernel.

15) There is a define, "SEPFLAG", in many Makefiles, that governs compilation for separate and non-separate I/D machines. If you have a non-separate I/D machine, set it to "-n". If you have a separate I/D machine, set it to "-i". This define needs to be set in *Makefiles* in several directories, each of which is found in *usr/src*. These directories are *bin, etc, games, lib, local, old, ucb, usr.bin,* and *usr.lib*. Each of these *Makefiles* has been written to pass this flag down to its subdirectories. The default "SEPFLAG" value is "-i". You may also enter the definition "SEPFLAG=-i" in your shell environment, rather than explicitly within the various makefiles. See *make*(1) for more information.

17) The directory *usr/src/new* is a compendium of things that we moved in from the 4.3BSD directory *usr/src/new* and things we moved in from from the 2.9BSD *usr/src/local* and *usr/src/contrib* directories. They come with even less of a guarantee than the rest of the system. If that's possible.