Setting Up UNIX

R. C. Haight
T. J. Kowalski
M. J. Petrella
L. A. Wehr

1. INTRODUCTION

1.1 Prerequisites

Before attempting to generate a UNIX† system, you should understand that a considerable knowledge of the related documentation is required and assumed. In particular, you should be very familiar with the following documents:

- *The UNIX Time-Sharing System*
- *UNIX User's Manual*
- *C Reference Manual*
- *Administrative Advice for UNIX*

A complete set of pertinent documentation is contained in *Documents for UNIX*. Throughout this document, each reference of the form *name*(N), where N is a Arabic number, refers to entry *name* in Section N of the *UNIX User's Manual*.

You must have a basic understanding of the operation of the hardware. This includes the console panel, the tape drives, and the disk drives, all of which are assumed to have standard UNIX addresses and interrupt vectors. It is also assumed that the hardware works and has been completely installed. All appropriate DEC diagnostics and the Equipment Test Package should have been run to test the configuration, and you must have a detailed description of the hardware, including device addresses, interrupt vectors, and bus levels. This information is necessary to generate the UNIX system.

See Attachment 1 for a list of supported CPUs and devices.

1.2 Procedure

UNIX is distributed on a single, multi-file magnetic tape, recorded in 9-track format at 800 bpi. Distribution tapes will be marked either ''PDP-11'' or ''VAX''; be sure you have the correct tape for your machine.

The initial load program will copy a file system from tape (VAX: TE16; PDP-11: either a TU10 or a TU16) to disk (VAX: RP06; PDP-11: either an RP03, an RK05, an RL01, or an RP06). In this document, we consider RP04 and RP05 drives to be equivalent to RP06 drives; any differences will be noted explicitly. Once the *root* file system has been successfully loaded to disk, UNIX may be booted and the available utility programs may then be used to complete the installation.

The remaining files on the tape contain source text and supplemental commands. These files contain essential information to generate a new system that will match your particular hardware and software environment.

In order for any of the update procedures to work correctly, you *must* be *running* UNIX/TS or PWB/UNIX Edition 2.0. Older versions of UNIX cannot be correctly *updated* with a UNIX system. The *cpio*(1) program will not replace any file if its replacement has a modification time that is less than (i.e., earlier than) the modification time of the original file. This can be due to local modifications. Furthermore, certain administrative files (e.g., */etc/passwd*, */usr/lib/crontab*) are sent with a modification time of January 1, 1970 to ensure that they do not replace their counterparts during updates. Any file not copied will cause

---

† UNIX is a Trademark of Bell Laboratories.

cpio(1) to print a message to that effect. These messages should always be investigated to ensure that any files not copied were of that type. However, note that, depending on respective modification times, a locally-modified file may get updated, thus destroying the local modifications.

There are several difficulties that can arise when installing a UNIX system. One of the most common problems is running out of disk space when performing an update. Should this occur, the original contents of the file system should be restored from a backup copy and the contents of the update tape should be read into a spare file system using the *cpio*(1) program. Unwanted material can then be removed and the original file system can be updated from this new file system using the $-p$ option of *cpio*(1). Modification times of files should also be preserved using the $-m$ option of *cpio*(1).

This document is not strictly linear. Read it thoroughly, from start to finish, and then read it again. Also, remove the write-protect ring, if present, from the distribution tape to guard against accidental erasure.

2. LOAD PROCEDURES

2.1 Distribution Tape Format

The eight files are: a loader, a physical copy of the *root* file system, the *cpio*(1) program, a *cpio*(1) structured copy of the *root* file system, and three files (*cpio*) that make up */usr*. *Root* refers to the directory "/", which is the root of all the directory trees. The format of this tape is as follows:

record 0:	Tape boot loader–512-bytes;
record 1:	Tape boot loader–512-bytes;
remainder of file 1:	Initial load program–several 512-byte records;
end-of-file	
file 2:	*root* file system (physical)–several 5,120-byte records (blocking factor 10);
end-of-file	
file 3:	*cpio* program (latest version)–several 512-byte records;
end-of-file	
file 4:	*root* file system (structured in *cpio* format)–several 5,120-byte records (to be used *only* for updating an earlier UNIX);
end-of-file	
file 5:	*/usr* file system except *src* and *man* (*cpio*).
end-of-file	
file 6:	*/usr/src* part 1.
end-of-file	
file 7:	*/usr/src* part 2.
end-of-file	
file 8:	*/usr/man*.
end-of-file	

The *root* (/) file system contains the following directories:

bck:	Directory used to mount a backup file system for file restoring.
bin:	Public commands; most of what is described in Section 1 of the *UNIX User's Manual*.
dev:	Special files, all the devices on the system.
etc:	Administrative programs and tables.
lib:	Public libraries, parts of the assembler, C compiler.
mnt:	Directory used to mount a file system.
lost+found:	Directory used by *fsck*(1M) for disconnected files.
stand:	Stand-alone programs.

tmp:	Directory used for temporary files; should be cleaned at reboot.
usr:	Directory used to mount the */usr* file system; user directories often kept here also.

2.2 Initial Load of root

Mount the distribution tape on drive 0 and position it at the load point.

2.2.1 PDP-11

Bootstrap the tape by reading either record 0 or record 1 into memory starting at address 0 and start execution at address 0. This may be accomplished by using a standard DEC ROM bootstrap loader, a special ROM, or some manual procedure; see *romboot*(8), *tapeboot*(8), and *70boot*(8).

2.2.2 VAX

See "Installation Boot Procedures" under *vaxops*(8). This *UNIX User's Manual* entry describes initial tape booting, and modification of the console floppy disk to simplify UNIX administration.

2.3 Common to PDP-11 and VAX

The tape boot loader will then type "UNIX tape boot loader" on the console terminal and read in and execute the initial load program. The program will then type detailed instructions about the operation of the program on the console terminal. First, it will ask what type of CPU you have. Second, the program will ask what type of disk drive you have and which drive you plan to use for the copy. The disk controller used must be at the standard DEC address indicated by the program. However, other disk controllers on your system may be at non-standard addresses. You must mount a formatted, ECC flag-free pack on the drive you have indicated. If necessary, use the appropriate DEC diagnostic program to format the pack. Note that the pack will be written on. Third, the program will ask what type of tape drive you have and which drive contains the tape. Normally, this will be drive 0, but the program will work with other drives. Note that the tape is currently positioned correctly after the end-of-file between the initial load program and the *root* file system. When everything is ready, the program will copy the file system from the tape to disk and give instructions for booting UNIX. After the copy is complete and you have booted the basic version of UNIX, check (using *fsck*(1M)) the *root* file system and browse through it.

2.3.1 PDP-11/70 Only

The file */stand/mmtest* is a stand-alone memory mapping diagnostic program for the PDP-11/70. It should be booted and run (20 minutes) if you are not *absolutely* sure that DEC FCO (field change order) M8140-R002 has been applied to your PDP-11/70 CPU.

2.4 Update of root

It is very important that the system be running in *single-user* mode during the update phase. To update an already existing *root* file system, files three and four will be used. It is necessary to first make a copy of your *root* file system using *volcopy*(1M) and then update this copy. The copy should be made on a separate disk pack using the same section number as your *root* file system (always section 0). Also, after the update is completed, check if any of your local administrative files in the directory */etc* need modification. Most of these are mentioned in Section 4 below.

Mount the tape on drive 0 and position it at the load point. We assume that disk drive 1 is available for making the copy, and that the *root* file system is on */dev/rp0*. This shell procedure will be different for RK05 and RL01 disk drives. The following procedure will first make a copy of the *root* file system, and then update this copy. Note that */dev/mt4* refers to tape drive 0 but has the side effect of spacing forward to the next end-of-file (no rewind option). The *B* option of *cpio* specifies that input is in 5,120-byte records:

```
volcopy root /dev/rrp0 pkname1 /dev/rrp10 pkname2
mount /dev/rp10 /bck
#   The 2 echoes will move the tape to file 3
echo </dev/mt4
echo </dev/mt4
cp /dev/mt4 /bck/bin/cpio
chmod 755 /bck/bin/cpio
chown bin /bck/bin/cpio
cd /bck
/bck/bin/cpio −idmB </dev/rmt0
cd /
umount /dev/rp10
```

Pkname1 and *pkname2* are the volume names of the source and destination disk packs, respectively. If the new copy is satisfactory, shut down and halt the system, change disk packs, and reboot the system using the new *root*.

2.5 Files 5, 6, 7, and 8 (/usr) Format

File 5 contains the */usr* file system in *cpio*(1) format (5,120-byte records). The */usr* file system contains commands and files that must be available (mounted) when the system is in *multi-user* mode. The tape contains the following directories:

adm:	Miscellaneous administrative command and data files, including the connect-time accounting file *wtmp* and the process accounting file *pacct*.
bin:	Public commands; an overflow for */bin*.
dict:	Dictionaries for word processing programs.
games:	Various demonstration and instructional programs.
include:	Public C language *#include* files.
lib:	Archive libraries, including the text processing macros; also contains data files for various programs, such as *spell*(1) and *cron*(1M).
mail:	Mail directory.
man:	Source for the *UNIX User's Manual*; see *man*(1).
lost+found:	Directory used by *fsck*(1M) for disconnected files.
news:	Place for all the various system news.
pub:	Handy public information, e.g., table of ASCII characters.
spool:	Spool directory for daemons.
src:	Source for commands, libraries, the system, etc.
tmp:	Directory for temporary files; should be cleaned at reboot.

A table of contents of this tape may be obtained by using the *cpio*(1) options *−itB*. Also, after installation, files and directories deemed useless by the local administrator may be easily removed. Alternately, only parts of the tape may be extracted using the pattern matching capabilities of *cpio*(1).

2.6 Initial Load of /usr

Mount a file system (device) as */usr*. The ultimate size and location of this file system on a device is an administrative decision; initially, the following procedure will suffice:

```
#   The 4 echoes will move the tape to file 5
#   (mt4 is the no-rewind device)
echo </dev/mt4
echo </dev/mt4
echo </dev/mt4
echo </dev/mt4
cd /
mkfs /dev/rrp1 35000 7 418
#   The magic numbers "7 418" above refer to free-list ordering:
#   (rotational angle of 7 and 418 blocks/cylinder for RP04/5/6/s)
#   If you have RL01 drives, use mkfs /dev/rrl1 10240 13 20
#   If you have RK05 drives, use mkfs /dev/rrk1 4872 3 24
labelit /dev/rrp1 usr pkname
mount /dev/rp1 /usr
cd /usr
cpio −idmB </dev/rmt4  #  file 5: /usr except /src and /man
cd /usr/src
cpio −idmB </dev/rmt4  #  file 6: 1st part of /usr/src
cpio −idmB </dev/rmt4  #  file 7: last of /usr/src
cd /usr/man
cpio −idmB </dev/rmt0  #  file 8: /usr/man
#   If you have RL01 or RK05 drives, you will have to use separate
#   packs for files 5-8.
```

Pkname is the volume name of the pack (e.g., ‘‘p0001’’).

Because */usr* must be mounted when the system is in *multi-user* mode, the file */etc/rc* must be changed to include the command lines to mount and unmount the file systems in *single-user* and *multi-user* mode. These lines must be inserted at the appropriate places in */etc/rc*, as indicated by comments in the prototype file. Next the file */etc/checklist* should be changed to include */dev/rrp1*, */dev/rrl1*, or */dev/rrk1*; see also *fsck*(1M), *labelit*(1M), *mkfs*(1M), *mount*(1M), and *checklist*(5).

2.7 Update of /usr

It is advisable that the system be running in *single-user* mode during the update phase. It is also wise to first make a copy of your */usr* file system for backup purposes. Next, mount the distribution tape on drive 0 and position it at file 5. The */usr* file system *must* also be mounted. The following procedure will perform the update:

```
cd /usr
cpio −idmB </dev/rmt4
cpio −idmB </dev/rmt4
cpio −idmB </dev/rmt4
cpio −idmB </dev/rmt0
```

3. CONFIGURATION PLANNING

3.1 UNIX Configuration

The basic UNIX operating system supplied supports only the console, a disk controller (disk drive 0), and a tape controller (tape drive 0). The actual configuration of your system must be described by you. All of the UNIX operating system source code and object libraries are in */usr/src/uts*. All of the configuration information is kept in the directory */usr/src/uts/*/cf*. There are only two files that must be changed to reflect your system configuration, *low.s* and *conf.c*; the program *config*(1M) should be used to make these changes.

Config requires a *system description file* and produces the two needed files. The first part of the system

description file lists all of the hardware devices on your system. Next, various system information is listed. A brief explanation of this information follows; for more details of syntax and structure, see *config*(1M) and the associated *master*(5); TABLE I lists the values and sizes of the various parameters for the different CPUs.

TABLE I. UNIX Configuration Parameters

Item	PDP-11/34, /23		PDP-11/45, /70		VAX-11/780	
	Value	*Size*	*Value*	*Size*	*Value*	*Size*
nswap	1000	–	3000	–	9000	–
buffers	15-20	26†	25-60	26†	50-120‡	560
sabufs	4-6	538	10-15	538	–	–
inodes	30-50	74	100-250	74	100-250	76
files	30-50	8	100-250	8	100-250	12
mounts	3-5	8	8-16	8	8-16	16
coremap	50-100	4	50-100	4	–	–
swapmap	50-100	4	50-100	4	50-100	4
calls	15-30	6	30-60	6	30-60	12
procs	30-50	30	50-200	30	50-200	52
texts	10-15	12	25-50	12	25-50	16
clists	10-20	28	100-300	28	100-300	32
maxup	15	–	15	–	25	–

† Plus 512 bytes outside system space.
‡ 127 buffers is the system maximum.

- *root*–Specifies the device where the *root* file system is to be found. The device must be a block device with read/write capability because this device will be mounted read/write as "/". Thus, a tape can not be mounted as the *root*, but can be mounted as some read-only file system. Normally, *root* is disk drive 0, section 0.
- *pipe*–Specifies where pipes are to be allocated (must be a file system–the root file system is normally used).
- *dump*–Specifies the device to be used to dump memory after a system crash. Currently only the TU10 and TU16/TE16 tape drives are supported for this purpose.
- *swap*–Specifies the device and blocks that will be used for *swapping*. *Swplo* is the first block number used and *nswap* indicates how many blocks, starting at *swplo*, to use. Care must be taken that the swap area specified does not overlap any file system. For example, if section 0 is 8,000 blocks long, the *root* file system could occupy the first 6,000 blocks and *swap* the remaining 2,000 by specifying:

  ```
  root rp06 0
  swap rp06 0 6000 2000
  ```

  The VAX release is set up for a *root* of 7,000 blocks and a *swap* of 9,000 blocks.
- *buffers*–Specifies how many *system buffers* to allocate. Real time response improves as more buffers are allocated. UNIX buffers form a "data cache". Improvement in the hit rate of this cache tends to fall as the number of buffers is increased.
- *sabufs*–PDP-11 only: specifies how many *system addressable* buffers to allocate. One buffer is needed for every mounted file system. Certain I/O drivers need such buffers.
- *inodes*–Specifies how many *inode table* entries to allocate. Each entry represents a unique open inode. When the table overflows, the warning message "Inode table overflow" will be printed on the console. The table size should be increased if this happens regularly. The number of entries used depends on the number of active processes, texts, and mounts.
- *files*–Specifies how many *open-file table* entries to allocate. Each entry represents an open file. When the table overflows, the warning message "no file" will be printed on the console. The table

size should be increased if this happens regularly.

- *mounts*−Specifies how many *mount table* entries to allocate. Each entry represents a mounted file system. The *root* (*/*) will always be the first entry. When full, the *mount*(2) system call will return the error EBUSY.
- *coremap*−PDP-11 only: specifies how many entries to allocate to the *list of free memory*. Each entry represents a contiguous group of 64-byte blocks of free memory. When the list overflows, due to excessive fragmentation, the system will undoubtedly crash in an unpredictable manner. The number of entries used depends on the number of processes active, their sizes, and the amount of memory available.
- *swapmap*−Specifies how many entries to allocate to the *list of free swap blocks*. Exactly like the *coremap*, except it represents free blocks in the swap area, in 512-byte units.
- *calls*−Specifies how many *call-out table* entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler. The time unit is 1/60 of a second. The call-out table is used by the terminal handlers to provide terminal delays and by various other I/O handlers. When the table overflows, the system will crash and print the panic message ''Timeout table overflow'' on the console.
- *procs*−Specifies how many *process table* entries to allocate. Each entry represents an active process. The scheduler is always the first entry and *init*(8) is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user is in the range of 2-5. When full, the *fork*(2) system call will return the error EAGAIN.
- *texts*−Specifies how many *text table* entries to allocate. Each entry represents an active read-only text segment. Such programs are created by using the $-i$ or $-n$ option of the loader *ld*(1). When the table overflows, the warning message ''out of text'' is printed on the console.
- *clists*−Specifies how many *character list buffers* to allocate. Each buffer contains up to 24 bytes. The buffers are dynamically linked together to form input and output queues for the terminal lines and various other slow-speed devices. The average number of buffers needed per terminal line is in the range of 5-10. When full, input characters from terminals will be lost and not echoed.
- *maxup*−Specifies how many concurrent processes a user is allowed to run.
- *power*−Specifies whether to attempt restart after a power failure. A value 0 (default) indicates no restart, a value of 1 attempts power-fail restart. On restart, device drivers are called and process 1 (i.e., *init*) is sent a hangup signal; see *init*(8). VAX power-fail is provided for experimental use only in UNIX 3.0.

3.2 UNIX Generation

To generate a new UNIX operating system, make sure that the directories under */usr/src/uts* are up-to-date. Follow the procedure below:

```
cd /usr/src/uts
ed dfile
a
   [information as described above]
.
w
q
make −f uts.mk VER=ver SYS=sys CONFIG=dfile TYPE=type NODE=uucpname
```

Dfile is the complete path name or the path name relative to */usr/src/uts/*/cf* of the file containing the configuration information, *sys* is a system name, *ver* is normally *mmdd* (month and day). The resulting operating system executable file name is the result of concatenating *sys* and *ver* (i.e., ''utsa0513''). The *uucpname* is for network identification. *Type* is the CPU type: **i** is used for PDP-11/23 and /34, **vax** is used for VAX, and **id** is used for other CPUs. The procedure will compile *low.s* (*univec.c* on the VAX) and *conf.c*, and load them together with the object libraries into a file called *name*.

The PDP-11 system has a relatively small address space, so that if table sizes or the number of device types are too large, various error messages will result and the above procedure will only create an *a.out* file. In

particular, the maximum available data space is 49,152 bytes (57,344 bytes on the PDP-11/23 and the /34). The actual data space requested can be found by using *size*(1) on *a.out* and adding the *data*, *bss*, and, for PDP-11/23 and /34, text segment sizes. One then reduces the specified values for the various system entries until it all fits. The amount of space in the *bss* segment used for each entry is indicated in Section 3.1 above.

When you are satisfied with the new system, you can test it by the following procedure:

```
cd /usr/src/uts
cp sysver /        # sysver as above
cd /
rm /unix
ln /sysver /unix
sync
```

Then halt the processor and reboot the system. Note that this procedure results in two names for the operating system object, the generic */unix*, and the actual name, say */utsa0501*. An old system may be booted by referring to the actual name, but remember that many programs use the generic name */unix* to obtain the *name-list* of the system.

If the new system does not work, verify that the correct device addresses and interrupt vectors have been specified. If the *wrong* interrupt vector and the *correct* device address have been specified for a device, the operating system will print the error message "stray interrupt at XXX" when the device is accessed, where XXX is the correct interrupt vector. If the *wrong* device address is specified, the system will crash with a panic trap of type 0 (indicating a timeout) when the device is accessed.

3.3 Special Files

A special file must be made for every device on your system. Normally, all special files are located in the directory */dev*. Initially, this directory will contain:

console	console terminal
error	see *err*(4)
mem, kmem, null	see *mem*(4)
tty	see *tty*(4)
rp[0-7], rrp[0-7]	disk drive 0, sections 0-7
rl[0-1], rrl[0-1]	disk drives 0 and 1
rk[0-1], rrk[0-1]	disk drives 0 and 1
mt0, rmt0	tape drive 0 (800 bpi)
mt4, rmt4	tape drive 0 (800 bpi, no rewind).

There are two types of special files, block and character. This is indicated by the character *b* or *c* in the listing produced by *ls*(1) with the *−l* flag.

In addition, each special file has a major device number and a minor device number. The major device number refers to the device type and is used as an index into either the *bdevsw* or *cdevsw* table in the configuration file *conf.c*. The minor device number refers to a particular unit of the device type and is used only by the driver for that type. The *config* program with the **−t** option will list major device numbers.

The program *mknod*(1M) creates special files. For example, the following would create *part* of the initially-supplied */dev* directory:

```
cd /dev
mknod console c 0 0
mknod error c 20 0
mknod mem c 2 0; mknod kmem c 2 1; mknod null c 2 2
mknod tty c 13 0
mknod rp0 b 0 0; mknod rrp0 c 7 0
mknod mt0 b 1 0; mknod rmt0 c 6 0
mknod mt4 b 1 4; mknod rmt4 c 6 4
```

After the special files have been made, their access modes should be changed to appropriate values by *chmod*(1). For example:

```
cd /dev
chmod 622 console
chmod 444 error
chmod 644 mem kmem
chmod 666 null
chmod 666 tty
chmod 400 rp0 rrp0
chmod 666 mt0 rmt0
chmod 666 mt4 rmt4
```

Note that file names have no meaning to the *operating system* itself; only the major and minor device numbers are important. However, many *programs* expect that a particular file is a certain device. Thus, by convention, special files are named as follows:

block device	*conf.c*	*/dev*
RP03 disk	rp	rp*
RP04/5/6 disk	hp	rp*
RS03/4 fixed head disk	hs	rs*
RK05 disk	rk	rk*
RL01 disk	rl	rl*
TU10 tape	tm	mt*
TE/TU16 tape	ht	mt*

character device	*conf.c*	*/dev*
DL11 asynch. line	kl	tty*, console
DH11 asynch. line mux	dh	tty*
DMC11 sync. unit	dmc	dmc*
DZ11 asynch. line mux	dz	tty*
DN11 auto call unit	dn	dn*
DU11 synch. line	du	du*
KMC11 micro	kmc	kmc*
DZ11/KMC11 assist	dza,dzb	tty*
LP11 line printer	lp	lp*
RP03 disk	rp	rrp*
RP04/5/6 disk	hp	rrp*
RS03/4 fixed head disk	hs	rrs*
TU10 tape	tm	rmt*
TE/TU16 tape	ht	rmt*
error	err	error
memory	mm	mem, kmem, null
terminal	sy	tty

For those devices with a */dev* name ending in *, this character is replaced by a string of digits representing the *minor* device number. For example:

```
mknod /dev/mt1 b 1 1
mknod /dev/rp24 b 0 024
```

Note that for disks, an octal number scheme is maintained because each drive is split eight ways. Thus, */dev/rp24* refers to section 4 of physical drive 2. There is also a special file, */dev/swap*, that is used by the program *ps*(1). This file must reflect what device is used for swapping and must be readable. For example:

```
rm /dev/swap
mknod /dev/swap b 0 0
chmod 440 /dev/swap
```

3.4 File Systems

Each physical pack is split into eight logical sections. This split is defined in the operating system by a table with eight entries. Each table entry is two words long. The first specifies how many blocks are in the section, the second specifies the starting cylinder; see *hp*(4) (RP04/5/6) and *rp*(4) (RP03) for default cylinder and block assignments.

These values are described to the system in the header file */usr/include/sys/io.h* which may be changed by using the editor *ed*(1). After such a change, the system must be made again (see Section 3.2 above).

A file system starts at block 0 of a section of the disk and may be as large as the size of that section; if it is smaller than the size of a section, the remainder of that section is unused. Note that the sections themselves may overlap physical areas of the pack, but the file systems must *never* overlap.

The program *mkfs*(1M) is used to initialize a section of the disk to be a file system. Next, the program *labelit*(1M) is used to label the file system with its name and the name of the pack. Finally, the file system may be checked for consistency by using *fsck*(1M). The file system may then be mounted using *mount*(1M).

3.5 DZ11 software with KMC11 assist

KMC microprocessors may be used to control DZ11 asynchronous multiplexers, thus off-loading terminal-oriented functions from the main CPU. The software is distributed in two forms. The KMC11-A version does DMA output of data, character translations, tab expansions, etc. The KMC11-B version does these output functions in addition to doing DMA input of data. Each KMC11 can control up to four DZ11 multiplexers for a total of thirty-two asynchronous lines. Each system can support up to four KMC11 microprocessors. Up to sixty-four DZ11 lines can be controlled by KMC11 microprocessors.

3.5.1 Installation

1. Generate a system (see Section 3.2 above) by including each DZ11 to be controlled by a KMC11 in the configuration file. For example:

   ```
   #  For the KMC11-A
   dza X Y Z

   #  For the KMC11-B
   dzb X Y Z
   ```

   where X is the vector address, Y is the UNIBUS address, and Z is the bus request priority. Also include the KMC11 microprocessors in the configuration file:

   ```
   kmc X Y Z
   ```

2. Install the KMC11 microcode in */lib*:

   ```
   #  For the KMC11-A
   cd /usr/src/uts/*/up/dza/dzkload
   /lib/cpp dza.s | kas -o /lib/dzkmc.o

   #  For the KMC11-B
   cd /usr/src/uts/*/up/dzb/dzkload
   /lib/cpp main.s | kasb -o /lib/dzkmc.o
   ```

3. Copy *dzkload* into */etc*:

   ```
   #  For the KMC11-A
   cp /usr/src/uts/*/up/dza/dzkload /etc

   #  For the KMC11-B
   cp /usr/src/uts/*/up/dzb/dzkload /etc
   ```

4. Update */etc/rc* to execute *dzkload* for multi-user and power-fail *init* states. Each KMC11 used to control a DZ11 must be loaded with microcode. For each KMC11 used to control a DZ11 include:

   /etc/dzkload /dev/kmc?

   where **?** is the minor device number of that KMC11.

5. Special files (see Section 3.3 above) must be created for each KMC11 and DZ11 line:

   ```
   #  Example
   /etc/mknod /dev/kmc? c X ?
   /etc/mknod /dev/tty?? c Y Z
   ```

   X is the major device number of the KMC11 and **?** is the minor device number of the KMC11 controlling the DZ11 multiplexers, i.e., the KMC11 loaded with microcode in step 4. Y is the major device number of the *dza/dzb* device as is supplied by *config*(1M). Z is the minor device number for a particular line on a DZ11. This number is composed of three fields. The low-order three bits are the line number relative to a DZ11. The next three bits contain the minor device number of the DZ11 controlling these lines. Note that this number is the absolute DZ11 number, not the number relative to the KMC11. The high-order two bits are the minor device number of the KMC11 controlling this DZ11. For example:

   mknod /dev/tty?? c Y 0241

   specifies the second line (0 through 7) on the fifth DZ11 to be controlled by the KMC11 with minor device number 2. The DZ11 number is specified by the order of appearance in the configuration file. The first four DZ11 multiplexers must be associated with one KMC11 and the next four must be associated with another KMC11. The order in which the KMC11 microprocessors are specified is not significant.

4. ADMINISTRATIVE FILES

4.1 /etc/motd

This file contains the *message-of-the-day*. It is printed by */etc/profile* after every successful *login*.

4.2 /etc/rc

On the transition between init states, */etc/init* invokes */bin/sh* to run */etc/rc* (must have executable modes). So that */etc/rc* can properly handle the removal of temporary files and the mounting and unmounting of file systems, it is invoked with three arguments: new state, number of times this state has been entered, previous state. When the system is initially booted, */etc/rc* is invoked with arguments "1 0 1"; when state two(multi-user) is subsequently entered, the arguments are "2 0 1".

Daemons may be invoked either by */etc/rc* or by including lines for them in */etc/inittab*.

The */etc/rc* file is also used to initialize KMC11 microprocessors (see */etc/dzkload* and*/etc/vpmload* below).

This file must be edited to suit local conditions; see *init*(8).

4.3 /etc/inittab

This file indicates to */etc/init* which processes to create in each init state. By convention, state 1 is *single-user* and state 2 is *multi-user*. For example, the following line creates the single-user environment:

```
1:co:c:env HOME=/ PATH=/bin:/etc:/usr/bin /bin/sh </dev/console\
        >/dev/console 2>/dev/console
1:xx:k:/etc/getty console ! 0
```

This indicates that for state 1 a process with the arbitrary unique identifier *co* should be created. The program invoked for this process should be the shell and when it exits it should be reinvoked (*c* flag).

To attach a *login* process to the console in the multi-user state, add the line:

2**:**co**:**c**:**/etc/getty console 4

and for line */dev/tty00* for use by 300/150/110 baud terminals, add the following line:

2**:**00**:**c**:**/etc/getty tty00 0 60

The arguments to *getty* are the device, speed table, and number of seconds to allow before hanging up the line.

Again, this file must be edited for local conditions; see *getty*(8), *init*(8), and *inittab*(5).

4.4 /etc/dzkload

This file is invoked as a command by */etc/rc*. It contains instructions for initializing a KMC11 microprocessor which is to function as a controller for one or more DZ11 communications multiplexers (see Section 3.5 above). This file must be edited to suit the configuration.

4.5 /etc/passwd

This file is used to describe each *user* to the system. You must add a new line for each new user. Each line has seven fields separated by colons:

1. Login name: normally 1-6 characters, first character alphabetic, the remainder alphanumeric, no upper-case characters.
2. Encrypted password: initially null, filled in by *passwd*(1). The encrypted password contains 13 bytes, while the actual password is limited to a maximum of 8 bytes. The encrypted password may be followed by a comma and up to 4 more bytes of password "age" information.
3. User ID: a number between 0 and 65,535; 0 indicates the super-user. These other IDs are reserved:

   bin**::**2**:** software administration;
   sys**::**3**:** system operation;
   adm**::**4**:** system administration;
   uucp**::**5**:** UNIX-to-UNIX file copy;
   rje**::**68**:** remote job entry administration;
   games**::**196**:** miscellaneous; never a real user.

4. Group ID: the default is group 1 (one).
5. Accounting information: this field is used by various accounting programs. It usually contains the user name, department number, and account number.
6. Login directory: full path-name (keep them reasonably short).
7. Program name: if null, */bin/sh* is invoked after a successful login. If present, the named program is invoked in place of */bin/sh*.

For example:

ghh**::**38**:**1**:**6824-G.H.Hurtz(4357)**:**/usr/ghh**:**
grk**::**44**:**1**:**6510-S.P.LeName(4466)**:**/usr/grk**:**/bin/rsh

See also *passwd*(5), *login*(1), *passwd*(1).

4.6 /etc/group

This file is used to describe each *group* to the system. You must add a new line for each new group. Each line has four fields separated by colons:

1. Group name: normally 1-6 characters, first character alphabetic, rest alphanumeric, no upper-case characters.
2. Encrypted password: initially null, filled in by *passwd*(1). The encrypted password contains 13 bytes, while the actual password is limited to a maximum of 8 bytes.
3. Group ID: a number between 0 and 65,535.
4. Login names: list of all *login* names in the group, separated by commas.

We strongly discourage group passwords. See also *group*(5).

4.7 /etc/profile

When the shell is executed and is the leader of a process group, as is the case when it is invoked by *login*, it will read and execute the commands in */etc/profile* before executing commands in the user's *.profile* file. This allows the system administrator to set up a standard environment for all users (e.g., executing *umask*, setting shell variables, etc.) and take care of other housekeeping details (such as *news −n*).

4.8 /etc/checklist

This file contains a list of default devices to be checked for consistency by the *fsck*(1M) program. The devices normally correspond to those mounted when the system is in *multi-user* mode. For example, a sample checklist would be:

```
/dev/rp0
/dev/rrp1
```

4.9 /etc/shutdown

This file contains procedures to gracefully shut down the system in preparation for file-save or scheduled down-time.

4.10 /etc/filesave.?

This file contains the detailed procedures for the local file-save.

4.11 /usr/adm/pacct

This file contains the process accounting information; see *acct*(1M).

4.12 /usr/adm/wtmp

This file is the log of login processes.

5. REGENERATING SYSTEM SOFTWARE

System source is issued under the directory */usr/src*. The sub-directories are named *cmd* (commands), *lib* (libraries), *uts* (the operating system), *head* (header files), and *util* (utilities); see *mk*(8) for details on how to remake system software.

A couple of anomalies: the accounting routine that deals with holidays and the prime/non-prime time split must be recompiled at the end of each year (it is currently correct for BTL-Murray Hill in 1980). The file is */usr/src/cmd/acct/lib/pnpsplit.c*. A reminder is sent to */usr/adm/acct/nite/log*, the standard place for such messages, starting a week before year-end and continuing until *pnpsplit.c* is recompiled.

5.1 PDP-11 Command Regeneration

The distributed object code has been compiled for machines without separate "I/D" space and without floating-point hardware. If your system has separate I/D space (i.e., is a PDP-11/70 or PDP-11/45), you should *mkcmd* adb, awk, bs, cc, cpio, dc, du, dump, efl, f77, fgrep, find, fsck, lex, make, mkfs, nm, pcat, restor, spell, spline, tbl, tplot, troff, units, unpack, uucp, volcopy, and yacc. If your configuration has an FP11-[ABC] floating-point processor (or the compatible 11/23 chip), you should *mkcmd* acct, adb, awk, bs, cc, pack, spline, tplot, typo, and units. If your configuration has both separate I/D space and floating point, you should *mkcmd* acct, adb, awk, bs, cc, cpio, dc, du, dump, efl, f77, fgrep, find, fsck, lex, make, mkfs, nm, pack, pcat, restor, spell, spline, tbl, tplot, troff, typo, units, unpack, uucp, volcopy, and yacc.

6. FILE SYSTEM CONVERSION TO UNIX (VAX)

Procedures have been developed for converting UNIX file systems from PDP-11/UNIX (including UNIX/TS, PWB Edition 2.0, and Research Version 7) to VAX UNIX. Direct conversion from other systems (i.e., Version 6-based or UNIX/RT) is also possible, but the administrator should get help.

The following UNIX commands are referenced in this section: *cpio*(1), *find*(1), *fsck*(1M), *fscv*(1M), *mkdir*(1), *mkfs*(1M), *mount*(1M), and *umount*(1M). The reader is assumed to be familiar with them.

Unless you have repealed Murphy's Law, you should allow plenty of time for the conversion. As a lower

bound, it takes about two hours to convert each 65K of file system space.

6.1 Preliminaries

Obviously, the new system should be generated and decently tested before conversion is attempted. Source for local commands and libraries should be moved to the VAX and compiled and tested. Your users may also reasonably require time to develop conversion programs for data files that contain binary information.

6.1.1 The Old System

The file systems should be pruned of marginal files. The following shell sequence will get rid of all executables:

```
#   For each user file system:
find /usr-fs-list -type f -print | xargs file | \
     sed -n -e 's/\([^:]*\):.*executable/\1/p' >/usr/tmp/exfiles
#   You may want to look this file over before the next step.
xargs rm -f</usr/tmp/exfiles
rm /usr/tmp/exfiles
```

6.1.2 Spare Packs

Do not convert without spare packs—that is courting disaster. It is best to keep the old packs for several days, and to make backup tapes as well.

6.2 Converting the Hard Way

Using *find* and *cpio* takes much longer, but you will have optimized converted user file systems when you are finished (compacted *inodes* and directories, file and free-list blocks arranged for fastest access).

6.2.1 Copying from the Old System

The following steps should be executed by the super-user on an idle, stand-alone (old) system:

```
#   For each user file system:
cd /file-system-name
find . -cpio /dev/rmt1
```

The *-cpio* option of *find* produces ten-block records on physical tape in *cpio* format. Unless there are a great many linked files, a 1,600-bpi, 2,400-foot reel should hold about 65K file system blocks. If you have larger file systems, the easiest (fastest, safest) thing to do is to use a raw disk pack in place of the tape (i.e., **/dev/rrp??** in place of the **/dev/rmt1** above). In our tests, multi-reel *find/cpio* tapes have worked. *Find* can also be used to pick up *parts* of file systems that can be combined later as described below.

6.2.2 Under the New System

Re-create each file system as follows:

```
mkdir /file-system-name
mkfs /dev/rrp? blocks:inodes 7 418
#   The magic numbers "7 418" above refer to free-list ordering:
#   (rotational angle of 7 and 418 blocks/cylinder for RP04/5/6s)
labelit /dev/rrp? file-system-name pack-num
mount /dev/rp? /file-system-name
cd /file-system-name
#   Mount tape created during step 3
cpio -idmB </dev/rmt1
#   If you are combining the smaller file systems,
#   you may copy-in more than one tape per new file system
#   (but make sure that first-level directory names are unique)
```

After the tapes have been copied in, the new file systems should be unmounted and checked (using *fsck*(1M)).

6.3 Converting the Easy Way

The *fscv*(1M) command has been provided for fast conversion between PDP-11 and VAX file systems.

Note that *fscv* will not convert "special files" (user file systems only). *Fscv* source will compile and run on either system. It was designed primarily for use in computer labs where there are a mixture of PDP-11 and VAX systems.

Example 1:

```
#  Converting PDP-11 to VAX:
#  Make sure that file system cylinder boundaries agree!
fscv −v /dev/rrp21 /dev/rrp31
```

Example 2:

```
#  Converting "in place" to the PDP-11:
#  Anyone who does this without making a copy first deserves
#  whatever bad (plenty) that can happen!
fscv −p /dev/rrp12 /dev/rrp12
```

See the *fscv*(1M) manual entry.

6.4 A Final Precaution

It is only sensible to do another complete file system backup under the new operating system (using another set of tapes or packs).

ATTACHMENT 1

Processors and Peripherals Supported by UNIX

The following table summarizes the hardware configurations supported by UNIX:

CPU with *floating point*	PDP-11/23	PDP-11/34	PDP-11/45	PDP-11/70	VAX-11/780
Memory	256KB ECC MOS	256KB ECC MOS	256KB ECC MOS or core	0.5-1MB ECC MOS or core	1-4MB ECC MOS
Disk drives:					
RL01 (2 required)	F	F	S	N	–
RK05 (2 required)	–	O	O	–	–
RP03	–	–	O	–	–
RP04	–	O (UNIBUS)	O (UNIBUS)	O (MASSBUS)	–
RP06	–	S (UNIBUS)	F (UNIBUS)	F (MASSBUS)	F (MASSBUS)
Fixed-head disks:					
RF11	–	–	O	–	–
RS03,4	–	–	N (UNIBUS)	N (MASSBUS)	–
Tape drives:					
TU10	–	O	O	–	–
TU16	–	O	O	O	–
TE16	–	F	F	F	F
TU45	–	–	–	S	S
Line printer:					
LP11†	–	N	N	N	N
Asynch. interfaces:					
DLV11	S	–	–	–	–
DL11-E	–	S	S	S	–
DZ11	–	S	S	S	S
KMC11B/DZ11‡	–	F	F	F	F
DH11	–	O	O	O	–
Synch. interfaces:					
KMC11B/DMC line unit	–	S	S	S	S
DMC11	–	S	S	S	S
DU11	–	O	O	O	–
DQS11B	–	–	O	O	–
Auto-call unit:					
DN11AA/DA	–	S	S	S	S
Parallel interface:					
DRV11	S	–	–	–	–
DR11C	–	S	S	S	–
DR11B	–	S	S	S	–
DA11B (use DMCs)	–	S	S	S	–

Key: F → First choice.
S → Supported.
O → Driver provided, but device is obsolete.
N → Driver provided, but device is not recommended.
– → No support.

† Use a printer on an RS232 interface.

‡ 4 DZ11s (32 lines) per KMC.

June 1980