Setting Up Version 1.0 of Unix/32V Operating System

Thomas B. London
John F. Reiser

The distribution tape can be used only on a DEC VAX-11/780 with RP06 or RM03 disks and with TE16 tape drives. The tape consists of some preliminary bootstrapping programs followed by one filesystem image and one tape archive image (see tar(1)); if needed, individual files can be extracted after the initial construction of the filesystems.

If you are set up to do it, it is a good idea immediately to make a copy of the tape to guard against disaster. The tape is 9-track 800 BPI and contains some 512-byte records followed by many 10240-byte records. There are interspersed tapemarks; end-of-tape is signalled by a double end-of-file.

The tape contains binary images of the system and all the user level programs, along with source and manual sections for them. There are about 2100 UNIX files altogether. The first two tape files contain binary images, along with other things needed to flesh out the filesystem enough so UNIX will run. The second tape file is to be put on one filesystem called the 'root filesystem'. The filesystem size required is about 9600 blocks. The third tape file has all of the source and documentation. Altogether it requires about 20,000 512-byte disk blocks.

Making a Disk From Tape

This description is an annotated version of the 'sysgen' manual page in section 8 of the UNIX Programmer's Manual.

Perform the following bootstrap procedure to obtain a disk with a root filesystem on it.

1. Mount the magtape on drive 0 at load point. [Make sure that the ring is not inserted.]
2. Mount a disk pack on drive 0.
3. Key in at 30000 and execute the following boot program: [You may enter in lower-case, the LSI-11 will echo in upper-case. The machine's printouts are shown in italic, explanatory comments are within ( ). Terminate each line you type by carriage return or line-feed.]

   > > > HALT
   > > > UNJAM
   > > > INIT
   > > > D 30000 20009FDE
   > > > D + D0512001
   > > > D + 3204A101
   > > > D + C113C08F
   > > > D + A1D40424
   > > > D + 008FD00C
   > > > D + C1800000
   > > > D + 8F320800
   > > > D + 10A1FE00
   > > > D + 00C139D0
   > > > D + 00000004
   > > > START 30000

   The tape should move and the CPU should halt at location 3002A. If it doesn't, you probably have entered the program incorrectly. Start over and check your typing.
4. Start the CPU with

>>> START 0

5. The console should type

   =

   If the disk pack is already formatted, skip to step 6. Otherwise, format the pack with

   (bring in standalone RP06 formatter)
   = rp6fmt
   rp6fmt : Format RP06 Disk

   MBA no. : 0 (format spindle on mba *unit :* 0 (format unit zero)
   (this procedure should take about 20 minutes)
   (some diagnostic messages may appear here)

   unit : -1 (exit from formatter)
   = (back at tape boot level)

6. Next, verify the readability of the pack via

   (bring in RP06 verifier)
   = rpread
   rpread : Read RP06 Disk

   disk unit : 0 (specify unit zero)
   start block : 0 (start at block zero)
   no. blocks : (default is entire pack)

   (this procedure should take about 10 minutes)
   (some diagnostic messages may appear here)
   # Data Check errors : nn (number of soft errors)
   # Other errors : xx (number of hard errors)
   disk unit : -1 (exit verifier)
   = (back to tape boot)

   If the number of 'Other errors' is not zero, consideration should be given to obtaining a clean pack before proceeding further.

7. Copy the magtape to disk by the following procedure.

(bring in the tape to disk program)
= tdcopy
tdcopy : TM03 tape-to-RP06 disk copy

tape MBA # : 1 (tape mba is normally 1)
tape unit # : 0 (tape unit is normally 0)
tape file offset : 1 (skip over tp tape file)
tape block offset : 0

disk MBA # : 0 (disk mba is normally 0)
disk unit : 0 (disk unit is normally 0)
disk block offset : 0 (start at block zero)

no. of input blocks : 480
10240 = tape block size

normal termination
480 input blocks read
9600 output blocks written
= (back at tape boot level)

You now have a UNIX root filesystem.

Booting UNIX

Since DEC does not provide a program on the console floppy which boots the VAX from a program located at block zero of a disk spindle, we provide one here.

If the console is not in 'LSI mode' (i.e. >>> prompt), type the 'CONTROL-p' key (i.e. hold the control key down while you hit the 'p' key). Perform the following sequence.

```
>>> HALT
>>> LINK          (save the following sequence on the floppy)
                  (the prompt should change to <<<)
<<< HALT
<<< UNJAM
<<< INIT
<<< D 30000 00009FDE        (boot pgm for MBA 0, drive 0)
<<< D + D0512001
<<< D + D004A101
<<< D + 0400C113
<<< D + 10008F32
<<< D + D40424C1
<<< D + 8FD00CA1
<<< D + 80000000
<<< D + 320800C1
<<< D + A1FE008F
<<< D + 28C1D410
<<< D + 14C1D404
<<< D + C139D004
<<< D + 00000400
<<< START 30000
<<< START 2
(to exit from linking mode type 'control-c')
<<< 'control-c'
>>>
```

You are now ready to boot UNIX (yea!). Each time it is necessary to boot (or reboot) UNIX, one simply follows the sequence

```
                (we should now be in 'LSI mode')
                (i.e. >>> prompt)
                (if not, it may be necessary to type 'control-p')
                (and 'HALT\r' i.e. HALT followed by return key)
>>> PERFORM     (this executes the commands saved in floppy)
                (link file)
                (the console should echo each command in the file)
file : unix             (load and execute /unix)
```

The machine should type the following:

```
real mem  = xxx
avail mem = yyy
#
```

The *mem* messages give the amount of real (physical) memory and the memory available to user programs in bytes. For example, if your machine has 512K bytes of memory, then xxx will be 524228, i.e. exactly 512K.

UNIX is now running, and the 'UNIX Programmer's manual' applies; references below of the form X(Y) mean the subsection named X in section Y of the manual. The '#' is the prompt from the Shell, and indicates you are the super-user. The user name of the super-user is 'root' if you should find yourself in multi-user mode and need to log in. There is no password provided for 'root'; provide one by using passwd(1). In the future, when you reboot from 'LSI mode' (i.e. >>> prompt), you can type just

```
>>> PERFORM   (let the LSI-11 boot the system)
file : unix    (as above)
```

You now need to make some special file entries in the dev directory. These specify what sort of disk you are running on, what sort of tape drive you have, and where the filesystems are. For simplicity, this recipe creates fixed device names. These names will be used below, and some of them are built into various programs, so they are most convenient. For example, 'rp0a' will be used for the name of the root filesystem, and 'rp0h' will be used for the name of the filesystem. Also, this sequence will put the user filesystem on the same disk drive as the root, which is not the best place if you have more than one drive. Thus the prescription below should be taken only as one example of where to put things. See also the section on 'Disk layout' below.

In any event, change to the dev directory (via cd /dev) and, if you like, examine and perhaps change the entries there (use rm(1) and mknod(1)). The file 'rp0a' refers to the root file system; 'swap' to the swap-space filesystem; 'rp0h' to the user filesystem. The devices 'rrp0a' and 'rrp0h' are the 'raw' versions of the disks. Also, 'mt0' is tape drive 0, at 800 BPI; 'rmt0' is the raw tape, on which large records can be read and written; 'rmt4' is raw tape with the quirk that it does not rewind on close, which permits multifile tapes to be handled.

The next thing to do is to extract the rest of the data from the tape. Comments are enclosed in ( ); don't type these. The number in the first command is the size of the filesystem.

```
        (in the following, xxx should be 322278 if
         you are using RP06's, 113280 if RM03's)
        (the following command creates an empty filesystem)
# /etc/mkfs /dev/rp0h xxx
isize = 65496       (this is the number of available inodes)
m/n = 3 500(freelist interleave parameters)
# /etc/mount /dev/rp0h /usr      (mount the usr filesystem)
# cd /usr      (make /usr the current directory)
# cp /dev/rmt4 /dev/null  (skip first tape file (tp format))
# cp /dev/rmt4 /dev/null  (skip second tape file (root))
# tar xbf 20 /dev/rmt0     (extract the usr filesystem)
# cd /          (back to root)
# /etc/umount /dev/rp0h  (unmount /usr)
```

All of the data on the tape has been extracted. The tape will rewind automatically.

You may at this point mount the source filesystem (mount(1)). To do this type the following:

```
/etc/mount /dev/rp0h /usr
```

The source and manual pages are now available in subdirectories of /usr.

The above mount command is only needed if you intend to play around with source on a single user system. The filesystem is mounted automatically when multi-user mode is entered, by a command in the file /etc/rc. (See 'Disk Layout' below).

Before UNIX is turned up completely, a few configuration dependent exercises must be performed. At this point, it would be wise to read all of the manuals (especially 'Regenerating System Software') and to augment this reading with hand to hand combat.

Reconfiguration

The UNIX system running is configured to run with the given disk and tape, a console, up to 1 megabyte of main memory, and 8 DZ11 lines. This is probably not the correct configuration. You will have to correct the configuration table to reflect the true state of your machine.

It is wise at this point to know how to recompile the system. Print the file /usr/src/sys/sys/makefile using the command cat /usr/src/sys/sys/makefile. This file is input to the program 'make(1)' which if invoked with 'make unix', will recompile all of the system source.

There are certain magic numbers and configuration parameters imbedded in various device drivers that you may want to change. The device addresses of each device are defined in each driver. In case you have any non-standard device addresses, just change the address and recompile. Also, if the devices's interrupt vector address(es) are not currently known to the system (this is likely), then the file /usr/src/sys/sys/univec.c must be modified appropriately: namely, the proper interrupt routine addresses must be placed in the table 'UNIvec'. Use the DZ11 as an example (as distributed, the DZ11 vectors are assumed to be at locations c0 and c4 (hexadecimal)).

The DZ11 driver is set to run 8 lines. This can be changed in dz.c

The DC11 driver is set to run 4 lines. This can be changed in dc.c.

The DH11 driver is set to handle 3 DH11's with a full complement of 48 lines. If you have less, or more, you may want to edit dh.c.

The DN11 driver will handle 4 DN's. Edit dn.c.

The DU11 driver can only handle a single DU. This cannot be easily changed.

The KL/DL driver is set up to run a single DL11-A, -B, and no DL11-E's. To change this, edit kl.c to have NKL11 reflect the total number of DL11-AB's and NDL11 to reflect the number of DL11-E's. So far as the driver is concerned, the difference between the devices is their address.

The disk and tape drivers (hp.c, ht.c) are set up to run 1 drive and should be changed if you have more. The disk driver (hp.c) has a partition table which you may want to experiment with.

After all the corrections have been made, use ‘make(1)’ to recompile the system (or recompile individually if you wish: use the makefile as a guide). If you compiled individually, say ‘make unix’ in the directory /usr/src/sys/sys. The final object file (unix) should be moved to the root, and then booted to try it out. It is best to name it /nunix so as not to destroy the working system until you’re sure it does work. See Boot Procedures(8) for a discussion of booting. Note: before taking the system down, always (!!) perform a sync(1) to force delayed output to the disk.

Special Files

Next you must put in special files for the new devices in the directory /dev using mknod(1). Print the configuration file /usr/src/sys/sys/conf.c. This is the major device switch of each device class (block and character). There is one line for each device configured in your system and a null line for place holding for those devices not configured. The essential block special files were installed above; for any new devices, the major device number is selected by counting the line number (from zero) of the device’s entry in the block configuration table. Thus the first entry in the table bdevsw would be major device zero. This number is also printed in the table along the right margin.

The minor device is the drive number, unit number or partition as described under each device in section 4. For tapes where the unit is dial selectable, a special file may be made for each possible selection. You can also add entries for other disk drives.

In reality, device names are arbitrary. It is usually convenient to have a system for deriving names, but it doesn’t have to be the one presented above.

Some further notes on minor device numbers. The hp driver uses the 0100 bit of the minor device number to indicate whether or not to interleave a filesystem across more than one physical device. See hp(4) for more detail. The ht driver uses the 04 bit to indicate whether or not to rewind the tape when it is closed. The 010 bit indicates the density of the tape on TE16 drives. Again, see ht(4).

The naming of character devices is similar to block devices. Here the names are even more arbitrary except that devices meant to be used for teletype access should (to avoid confusion, no other reason) be named /dev/ttyX, where X is some string (as in ‘00’ or ‘library’). The files console, mem, kmem, and null are already correctly configured.

The disk and magtape drivers provide a ‘raw’ interface to the device which provides direct transmission between the user’s core and the device and allows reading or writing large records. The raw device counts as a character device, and should have the name of the corresponding standard block special file with ‘r’ prepended. Thus the raw magtape files would be called /dev/rmtX. These special files should be made.

When all the special files have been created, care should be taken to change the access modes (chmod(1)) on these files to appropriate values.

Time Conversion

If your machine is not in the Eastern time zone, you must edit (ed(1)) the file /usr/src/sys/h/param.h to reflect your local time. The manifest ‘TIMEZONE’ should be changed to reflect the time difference between local time and GMT in minutes. For EST, this is 5*60; for PST it would be 8*60. Finally, there is a ‘DSTFLAG’ manifest; when it is 1 it causes the time to shift to Daylight Savings automatically between the last Sundays in April and October (or other algorithms in 1974 and 1975). Normally this will not have to be reset. When the needed changes are done, recompile and load the system using make(1) and install it. (As a general rule, when a system header file is changed, the entire system should be recompiled. As it happens, the only uses of these flags are in /usr/src/sys/sys/sys4.c, so if this is all that was changed it alone needs to be recompiled.)

You may also want to look at timezone(3) (/usr/src/libc/gen/timezone.c) to see if the name of your timezone is in its internal table. If needed, edit the changes in. After timezone.c has been edited it should be compiled and installed in its library. (See /usr/src/libc/Makefile). Then you should (at your leisure) recompile and reinstall all programs that use it (such as date(1)).

Disk Layout

If there are to be more filesystems mounted than just the root and /usr, use mkfs(1) to create any new filesystem and put its mounting in the file /etc/rc (see init(8) and mount(1)). (You might look at /etc/rc anyway to see what has been provided for you.)

There are two considerations in deciding how to adjust the arrangement of things on your disks: the most important is making sure there is adequate space for what is required; secondarily, throughput should be maximized. Swap space is a critical parameter. The system as distributed has 8778 blocks for swap space. This should be large enough for most sites. You may want to change these if local wisdom indicates otherwise.

The system as distributed has many of the binaries in /bin. Some of them should be moved to /usr/bin, leaving only the ones required for system maintenance (such as icheck, dcheck, cc, ed, tar, restor, etc.) and the most heavily used in /bin. This will speed things up a bit if you have only one disk, and also free up space on the root filesystem for temporary files. (See below).

Many common system programs (C, the editor, the assembler etc.) create intermediate files in the /tmp directory, so the filesystem where this is stored also should be made large enough to accommodate most high-water marks. If you leave the root filesystem as distributed (except as discussed above) there should be no problem. All the programs that create files in /tmp take care to delete them, but most are not immune to events like being hung up upon, and can leave dregs. The directory should be examined every so often and the old files deleted.

Exhaustion of user-file space is certain to occur now and then; the only mechanisms for controlling this phenomenon are occasional use of du(1), df(1), quot(1), threatening messages of the day, and personal letters.

The efficiency with which UNIX is able to use the CPU is largely dictated by the configuration of disk controllers. For general time-sharing applications, the best strategy is to try to split user files, the root directory (including the /tmp directory) and the swap area among three controllers.

Once you have decided how to make best use of your hardware, the question is how to initialize it. If you have the equipment, the best way to move a filesystem is to dump it (dump(1)) to magtape, use mkfs(1) to create the new filesystem, and restore (restor(1)) the tape. If for some reason you don't want to use magtape, dump accepts an argument telling where to put the dump; you might use another disk. Sometimes a filesystem has to be increased in logical size without copying. The super-block of the device has a word giving the highest address which can be allocated. For relatively small increases, this word can be patched using the debugger (adb(1)) and the free list reconstructed using icheck(1). The size should not be increased very greatly by this technique, however, since although the allocatable space will increase the maximum number of files will not (that is, the i-list size can't be changed). Read and understand the description given in filesystem(5) before playing around in this way. You may want to see section rp(4) for some suggestions on how to lay out the information on RP disks.

If you have to merge a filesystem into another, existing one, the best bet is to use tar(1). If you must shrink a filesystem, the best bet is to dump the original and restor it onto the new filesystem. However, this might not work if the i-list on the smaller filesystem is smaller than the maximum allocated inode on the larger. If this is the case, reconstruct the filesystem from scratch on another filesystem (perhaps using tar(1)) and then dump it. If you are playing with the root filesystem and only have one drive the procedure is more complicated. What you do is the following:

1. GET A SECOND PACK!!!!
2. Create an image of the new root filesystem using mkfs(1), dump(1), and restor(1).
3. Make a binary tape image of the new filesystem using dd(1).
4. Bring the system down and mount the new pack.
5. Retrieve the WECo distribution tape and perform steps 1 through 4 at the beginning of this document, then skip to step 7, substituting the desired filesystem size instead of 480 when asked for 'no. of input blocks'.

6. Boot(8) using the newly created disk filesystem.

New Users

Install new users by editing the password file /etc/passwd (passwd(5)). This procedure should be done before multi-user mode is entered (see init(8)). You'll have to make a current directory for each new user and change its owner to the newly installed name. Login as each user to make sure the password file is correctly edited. For example:

```
ed /etc/passwd
$a
joe::47:13::/usr/joe:
w
q
mkdir /usr/joe
chown joe /usr/joe
login joe
ls –la
login root
```

This will make a new login entry for joe, who should be encouraged to use passwd(1) to give himself a password. His default current directory is /usr/joe which has been created. The delivered password file has the user *bin* in it to be used as a prototype.

Multiple Users

If UNIX is to support simultaneous access from more than just the console terminal, the file /etc/ttys (ttys(5)) has to be edited. To add a new terminal be sure the device is configured and the special file exists, then set the first character of the appropriate line of /etc/ttys to 1 (or add a new line). Note that init.c will have to be recompiled if there are to be more than 100 terminals. Also note that if the special file is inaccessible when init tries to create a process for it, the system will thrash trying and retrying to open it.

File System Health

Periodically (say every day or so) and always after a crash, you should check all the filesystems for consistency (icheck, dcheck(1)). It is quite important to execute sync (8) before rebooting or taking the machine down. This is done automatically every 30 seconds by the update program (8) when a multiple-user system is running, but you should do it anyway to make sure.

Dumping of the filesystem should be done regularly, since once the system is going it is very easy to become complacent. Complete and incremental dumps are easily done with dump(1). Dumping of files by name is best done by tar(1) but the number of files is somewhat limited. Finally if there are enough drives entire disks can be copied using cp(1), or preferably with dd(1) using the raw special files and an appropriate block size.

Converting Sixth Edition Filesystems

The best way to convert filesystems from 6th edition (V6) to 7th edition (V7) format is to use tp(1) or tar(1).

Odds and Ends

The programs dump, icheck, quot, dcheck, ncheck, and df (source in /usr/source/cmd) should be changed to reflect your default mounted filesystem devices. Print the first few lines of these programs and the changes will be obvious. Tar should be changed to reflect your desired default tape drive.

Good Luck

Thomas B. London
John F. Reiser