SETTING UP MERT FOR THE FIRST TIME

1. Introduction and Overview

Installation of MERT Release 0 consists of two distinct phases:

A. If this is your first installation of MERT, you'll have to follow the instructions of this section to set up a bare-bones MERT system to be used for generating your customized operating system. Also, you will want to store away safely and conveniently the distributed software which is needed for the generation process. At the end of Phase A, you will have a MERT system up and running, and you will be in a position similar to the existing MERT installation that wants to use the distributed files to convert the existing system to an updated customized MERT operating system.

B. With a MERT system up and running and the distributed files on hand, MERT software can now be (re-)generated for your installation. Thus, if you are an existing installation, you can skip to Section 20 *Generating MERT Software*.

For reference purposes, here is a listing of the paragraphs of this section:

- 1. Introduction and Overview
- 2. Distribution Tapes
- 3. Stand-Alone Copy of File System Tape onto a Disk
- 4. Booting MERT for a First Installation
- 5. The Root File System
- 6. Planning the Layout of Your Installation
- 7. Special Files (Hardware Names)
- 8. Multiple Users (Time-Sharing Use)
- 9. The Source Distribution Tape
- 10. The Manual File System
- 11. Changing the Disk Layout
- 12. Moving File Systems Around
- 13. Installing New Users
- 14. File System Maintenance
- 15. Odds and Ends

2. Distribution Tapes

MERT Release 0 software is distributed on two magnetic tapes. One tape has a *tp*-formatted leader followed by three (MERT) file system images *root, usr,* and *man,* and the other is a *cpio*-formatted tape that contains the source code for the entire system. If you are set up to do it, make a copy of these tapes to guard against disaster.

The tape marked *File Systems* contains 12100 512-byte records followed by a single file mark. The first 100 blocks contain stand-alone programs (mostly various boots) in *tp*-format

(see tp[V]), followed by three MERT file systems. Each file system is exactly 4000 blocks long on the tape so that each also fits on an RK disk.

The first file system is the root file system. It contains all of the MERT operating system and the commands in binary form. See 6. *The Root File System* below for a more complete description.

The other file systems are *usr* and *man*. Usr has been separated from the root file system because of the RK file size restriction; it should be permanently mounted.

The *man* file system contains all of the contents of the MERT Release 0 Manual. It is likely that some of the manual pages have changed since the manual went to press. These changes are indicated in the cover letter.

The *cpio*-formatted tape labelled *Source* contains the source code and shells to make(compile) the entire system. Briefly, the source tape contains :

MERT operating system source User command source Library source Boot program source Bourne shell scripts to make the system

More details on the makeup of the source tape and ways to extract and compile various directories will be given below under 9. *The Source Distribution Tape*.

3. Stand-Alone Copy of File System Tape onto a Disk

Perform the following bootstrap procedure to obtain a disk with the binaries (root file system).

- A. Mount magtape on drive 0 at load point.
- B. Mount formatted disk pack on drive 0.
- C. Key in and execute at 100000

TU10	TU16
012700	Use the DEC ROM or other
172526	means to load block 1
010040	(i.e. second block) at 800 BPI
012740	into location 0 and transfer
060003	to 0.
000777	

The tape should move and the CPU loop. (The TU10 code is *not* the DEC bulk ROM for tape; it reads block 0, not block 1.)

- D. Halt and restart the CPU at 0. The tape should rewind. The console should type ' $\stackrel{\sim}{=}$.
- E. Copy the magtape to disk by the following. This assumes TU10 and RK05; see F below for other devices. The machine's printouts are shown in italic (the '=' signs should be considered italic). Terminate each line you type by carriage return or line-feed. (Actually, the first two responses to copy don't seem to need it).

= copv'p' for rp03; 'k' for rk;'4' for rp04 k 'm' for tml1; 'u' for tu16;'c' for tc11 m disk offset 0 tape offset 100 (See F below) count 1 (The tape should move) = copy'p' for rp03;'f' for rf; 'k' for rk;'4' for rp04 k 'm' for tml1; 'u' for tu16;'c' for tc11 m disk offset 1 tape offset (See 7 below) 101 count 3999 (The tape moves lots more)

To explain: the *copy* program copies tape to disk with the given offsets and counts. Its first use copies a bootstrap program to disk block 0; the second use copies the file system itself onto the disk. You may get back to '=' level by starting at 157000.

- F. If you have TU16 tape say 'u' instead of 'm' in the above example. If you have an RP03 disk, say 'p' instead of 'k', and use a 99 instead of 100 tape offset. If you have an RP04 disk, use '4' instead of 'k', and use a 98 instead of 100 tape offset. The different offsets load bootstrap programs appropriate to the disk they will live on.
- G. This procedure generates the root disk; the usr disk may be generated on another RK pack by using a tape offset of 4101 instead of 101. The man source disk is at offset 8101 instead of 101. Unless you have only a single RK drive, it is probably wise to wait on generating these disks. Better tools are available using-MERT-UNIX itself. For instance, the UNIX command dd-I may be used to extract both the *usr* and *man* file systems. As an example, the *usr* file system image, which begins at block 4100 on the tape, may be copied using the command:

dd if =/dev/mt0 of =/dev/rk1 count =4000 skip =4100

Likewise, the command:

dd if =/dev/mt0 of =/dev/rk2 count =4000 skip =8100

may be used to extract the manual file system man. Note, that both of these file systems should be mounted respectively on /usr and /man of the root file system.

4. Booting MERT For a First Installation

Once the MERT root disk is obtained, the system is booted by keying in (if you don't have a ROM) and executing one of the following programs at 100000. These programs correspond to the DEC bulk ROMs for disks, since they read in and execute block 0 at location

0.

RK05	RP03	RP04
012700	012700	Use the DEC ROM or other
177414	176726	means to load block 0 into
005040	005040	location 0 and transfer
005040	005040	to 0.
010040	005040	
012740	010040	
000005	012740	
105710	000005	
002376	105710	
005007	002376	
	005007	

Now follow the indicated dialog, where '=' and '#' are prompts:

= rkmert (or 'rpmert' or 'hpmert') n memory partitions: mem = xxxxx; yyyyyy login: root #

The *memory partition* message gives the memory starting address (xxxxxx) and memory allocated to each partition (yyyyyy) in 64 byte units in octal. There are *n* memory partitions (a maximum of 3). Most of the MERT software will run with 3200 (for 52K words), but some things require much more.

MERT is now running, and the *MERT Programmer's Manual* and the UNIX Programmer's Manual apply; in references below UNIX manual pages are suffixed with a Roman numeral (e.g. ed-I), MERT manual pages have a lower case letter to denote the section (e.g. *pcp-e*). The '#' is the prompt from the Bourne Shell, and indicates you are logged in as the superuser. The only valid user names are 'root' and 'bin'. The root is the superuser and is the owner of nearly every file in the file system. At this point, it would be wise to read all of the manuals and to augment this reading with hand-to-hand combat.

Notice that the currently running version of MERT is provisional only in the sense that it only should be used to generate a MERT operating system for your installation. Explicitly, the booted version of MERT ignores the floating point hardware (if you have any) on your system, using software emulation instead. (This way, the distributed version will run on systems with or without floating point hardware.) Further, all the executable commands distributed with this system are compiled for a floating point hardware system (which most people have). Some of these commands would fail with a core dump under this operating system. Stated differently, you have to remake either the operating system or the commands setting the floating point flag according to whether you have floating point hardware or not. In addition, you also need to remake your MERT system to accommodate for your hardware configuration.

Before starting to adapt your system to the hardware you are running on, you should familiarize yourself with the distributed system. Continue reading this document and, on the way, inspect the distributed root system. To guide you during this effort, we'll first give a description of the root file system.

5. The Root File System

The root file system disk on which you are running is configured with 4000 blocks for the file system and 872 blocks for swap space. The root file system contains all of the binary object files necessary to run the MERT system. (Included in the root file system is a shell script *extract.sh* which can be used to extract various parts of the source distribution tape onto the directory */src*. This is explained in paragraph 9. *The Source Distribution Tape.*)

On the tape we find

rkmert boot image for RK05 disk rpmert boot image for RP03 disk hpmert boot image for RP04 disk

and the following directories:

bin	UNIX user binary object programs
cdmp	space for core dumps of processes terminated abnormally
crp	available for mounting of file system
dev	character, block and record device file names
etc	miscellaneous programs
lib	library files and passes of the C compiler
mnt	available for mounting of file system
man	available for mounting of the manual file system
mrt	miscellaneous MERT files
prc	process images
usr	available for mounting of usr file system
tmp	temporary files
src	available for copying directories from the cpio-formatted distribution tape

You might examine each one of these directories and see what there is. The following directories bear some more discussion.

5.1. dev

This directory contains the names of all of the special device files. There should be one file name for each logical device in your system. Each major device may have a number of logical channels (minor devices) associated with it. See the discussion in 7. Special Files for a detailed description, of how you can generate any special files that you need.

5.2. prc

The load images of all of the processes which may be run on the MERT system, both kernel and supervisor-user, are contained in this directory. The UNIX and 'pkill' processes are the only supervisor-user processes distributed with the system. The names of the character, block and record device driver processes are related to the major device numbers of the devices as assigned in the '/dev' directory. See the discussion below on Special Files for details.

5.3. cdmp

This directory is reserved for producing core dumps of the processes which are terminated abnormally. The name of the 'core dump' file is the same as the name of the original process image in the '/prc' directory or any other directory. For example, if a break-point trap is planted in '/prc/cdb' and that break-point is executed, a core dump will be produced by the process manager in '/cdmp/cdb'. the core image may then be debugged by:

PA-1C600-01 Section 19 Issue 1, October 1977 AT&TCo SPCS

adb /prc/cdb /cdmp/cdb

5.4. mrt

Three MERT files are contained in this directory:

kprc	process idchar to name translation
krn.sym	kernel symbol table
syslib	system library.

The 'krn.sym' and 'kprc' files are used by the *ps-I* command to determine the memory address of the DCT (Dispatcher Control Table) and the identity of each kernel process, respectively. Thus it is important that these files correspond to the currently running system.

The system library 'syslib' is the one that is normally included in the boot image. It is also the one which is referenced by some character device drivers which use the system library. Note that the same version number must be used in both cases to guarantee that the system will boot properly.

5.5. tmp

This directory is use by UNIX user programs such as the editor, assembler and C compiler for temporary intermediate files. The contents of this directory should be looked at periodically and the old files deleted. There should be 1000 or so free blocks on the file system which contains this directory.

5.6. usr

The file system mounted on this directory contains some of the frequently used MERT and UNIX user commands in *lusr/bin* as well as user library routines in *lusr/lib* and C data structures in *lusr/include*.

5.7. src

This directory is available for copying various directories from the source distribution tape using the shell script *extract.sh*. If the entire cpio tape is copied, this directory will contain the following subdirectories:

mertsrc	MERT operating system source
cmd	user command source
lib	library source
mdec	boot programs source
makefile	aids to remake cmd

6. Planning the Layout for Your Installation

To prepare yourself for the system generation process described in the next section, you have to create and mount several file systems, and you have to edit several installation-dependent files and directories. (An existing installation which is merely upgrading should already have these and needs only some editing to comply with the new standards.) An overview of the required steps is given here; detailed instructions follow in the paragraphs below.

At least two file systems need to be accessed: */usr* and */src*. Since there are to be more file systems mounted than just the root, you must use mkfs-VIII to create the new file systems and make an entry in the shell procedure /etc/rc if they are to be mounted automatically on each

reboot (see init-VIII and mount-VIII). (You might look at /etc/rc anyway to see what has been provided for you.)

The *lusr* file system is needed because it contains the required libraries in *lusr/lib* and a good deal of the commands in *lusr/bin* as well as data structures in*lusr/include*. 4000 blocks are more than sufficient for the supplied files, but it is customary, to include your users' file space in this file system, so you may want to make it larger.

The *lsrc* file system is needed only for the system generation process, so it can be unmounted thereafter. You will need about 6500 blocks; if you don't have that much space (RK system), you will have to do it in steps as explained later.

The file system *lman* is only needed if you want to provide on-line documentation (man-I command) or want to reproduce the manuals.

The root system you are running on can not acces any other disk space than the one occupied by itself. To access additional file systems, such as the other two distributed ones, you must perform four tasks:

1. The unused portions of your disk(s) must be made known to your system by giving them filenames with associated major/minor device numbers. Use the *mknod-VIII* command. In Tables 7.1 and 7.2 below, you can find the name of your disk (Table 7.1), the pertinent major device number from Table 7.2, and finally in Section IV the correct names for the subsections of your disk. As an example, for the RP04 disk Table 7.1 gives name hp0 and hints to manual page HP-IV); there you find that the disk subsections are known as hp0, hp1, hp2, ..., hp7, and each of these names is by convention associated with a certain section on the disk. The connection, though, is made through the *minor device* number, which is the same as the "disk" or partition number in the table. You'll find the major device number for fIhpfR disks is standardized in Table 7.2 to be 5. On manual page HP(IV) you find that we could choose, e.g., hp1 to be the name *ldev/hp1* for a special file denoting a file of size up to 65535 blocks. The command

letc/mknod /dev/hp1 b 5 1

would create the special file entry. (Claim the remaining disk space by making appropriate use of mknod.) Make sure that you do not choose overlapping disk areas for file systems or swap space.

At this point there exist two alternatives: Already formatted file system images can be copied into such spaces, as will be done in step 2A. below with the *lusr* and *lman* file systems. Alternatively, we format the new space into a file system and move files into it one by one. This is done for the Source tape in steps 2B. through 4 below.

2A. Copy the already formatted MERT file systems *lusr* or *lman* into the just named disk spaces, by executing the following copy command as explained in paragraph 3.G:

dd if =/dev/mt0 of =/dev/hp1 count =4000 skip =4100

This will copy the *lusr* file system supplied on tape into the special file *ldevlhp1*. If you want to access the third supplied file system *lman* you have to perform steps 1. and 2A. accordingly. Now you can mount these file systems with *letclmount*-VIII on appropriate mount points, e.g. *lusr* and *lman* have been provided in the root file system. You don't

Bell Telephone Laboratories, Incorporated - 8 - PROGRAM APPLICATION INSTRUCTION

need to do the following steps in this case.

- 2B. Format the newly named subsections of your disk to look like empty file systems to be filled with the supplied files. Use *mkfs*-VIII with the just created names in */dev* and appropriate sizes (at least 6500 blocks for */src*, 4000 for the others).
- 3. Now you use *letc/mount* to mount one of these empty file systems on one of the provided mount points in your root file system; e.g. *letc/mount ldev/hp1 /src*. (see *mount-*VIII).
- 4. The mounted empty file system could now be filled ; in the above example, execution of the shell *extract.sh all* (see paragraph 9. *The Source Distribution Tape*) will fill the file system with all the supplied source from the Source Distribution Tape.

Notice that the disk drivers supplied with MERT Release 0 comply with the definitions given in the appropriate device description pages in Section IV of the UNIX Manual.

If you have several devices, you may want to create *ltmp* for scratch space on a separate device (see 11. *Changing the Disk Layout* for tuning hints).

Now you have to modify a number of files and directories to define your configuration. First you will make entries for each device on your system in directory *ldev*, using the *mknod* command, and give each a major and minor device number (See paragraph 7.Special Files). Then, in directory *lprc*, each major device will get an entry matching the major device number in its name, to allow the system to access the device with the correct driver software.

Finally, you need to make some changes in certain files of the *lsrc* file system, which, of course, needs to be mounted first. A directory *sysgen* contains the files *sys.b* with specifications for the operating system compilation, and *header.s* with global options such as an indication of the availability of a floating point processor on the system. Then *conf.d* has to be edited, to indicate for the compilation of these drivers how many minor devices are available for each selected major device. Editing instructions for these files as well as certain hints on adapting the driver software are given in the next section (20. Generating MERT Software).

Before you proceed, you should mount the supplied file systems and have a look at them. Some more explanations on their contents are given in paragraphs 9. and 10. of this section.

7. Special Files (Hardware Names)

The MERT system is configured to run on an 11/45 CPU with the disk you booted on as the only block device and the control console (KL) as the only character device. You must put in all of the special files in the directory '/dev' using *mknod-VIII* to enable the loading of all of the disk and tape drivers. The block special devices are put in first by executing the following generic command for each disk or tape drive. (Note that some of these files already exist in the directory '/dev'. Examine each file with *ls-I* with the -1 flag to see if the file should be removed.)

/etc/mknod /dev/NAME b MAJOR MINOR

The NAME is selected from the following list:

Table 7.1: Abbreviations for Block and Record Devices			
NAME	DEVICE	SEE SECTION	
tap0	TU56 DECtape -		
rk0	RK03 RK05 moving head disk	RK(IV)	
mt0	TU10 TU16 magtape	TM,HT(IV)	
rp0	RP03 moving head disk	RP(IV)	
hs0	RS03 RS04 fixed head disk	HS(IS)	
hp0	RP0/5/6 moving head disk	HP(IV)	

The MAJOR device number should be selected in accordance with the following table, which matches the UNIX Generic 3 standard (its *conf.c* file). Special device drivers which are not distributed with the system should be made using one of the available (AVA) entries in Table 7.2. The reserved entries are for future device drivers and should not be used.

Table 7.2: Standard Assignment of Major Device No., Kernel Process Name, and Device Type

#	Process	Device	#	Process	Device
0	/prc/bda	rk	0	/prc/cda	kl
1	/prc/bdb	rp	1	/prc/cdb	RES
2	/prc/bdc	rf	2	/prc/cdc	kp
3	/prc/bdd	tm	3	/prc/cdd	dc
4	/prc/bde	tc	4	/prc/cde	dh,dm
5	/prc/bdf	hp	5	/prc/cdf	dp
6	/prc/bdg	ht	6	/prc/cdg	RES
7	/prc/bdh	hs	7	/prc/cdh	dn
8	/prc/bdi	AVA	8	/prc/cdi	mem
9	/prc/bdj	AVA	9	/prc/cdj	RES
10	/prc/bdk	AVA	10	/prc/cdk	RES
11	/prc/bdl	AVA	11	/prc/cdl	RES
12	/prc/bdm	AVA	12	/prc/cdm	RES
13	/prc/bdn	AVA	13	/prc/cdn	RES
14	/prc/bdo	AVA	14	/prc/cdo	RES
15	/prc/bdp	AVA	15	/prc/cdp	RES
16	/prc/bdq	AVA	16	/prc/cdq	RES
17	/prc/bdr	AVA	17	/prc/cdr	AVA
18	/prc/bds	AVA	18	/prc/cds	AVA
19	/prc/bdt	AVA	19	/prc/cdt	AVA
20	/prc/bdu	AVA	20	/prc/cdu	AVA
21	/prc/bdw	AVA	21	/prc/cdw	AVA
22	/prc/bdx	AVA	22	/prc/cdx	AVA
23	/prc/bdy	AVA	23	/prc/cdy	AVA
24	/prc/bdz	AVA	24	/prc/cdz	AVA

RES = reserved AVA = available

The MINOR device is the drive number, unit number or partition as described under each device in section IV. The last digit of the name (all given as 0 in the first table above) should reflect the minor device number. For tapes where the unit is dial-selectable, a special file may be made for each possible selection.

The boot image which you have received is configured with the swap device equal to the root device number. Remember that the magtape devices are record type devices. Therefore you must use 'r' rather than 'b' for the device type character.

The same goes for the character devices. Here the names are arbitrary except that devices meant to be used for teletype access should be named /dev/ttyX, where X is any character. The files tty8 (console), mem, kmem, null, kmemd and smem are already correctly configured.

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

The second special directory which must be reconstructed is '/prc'. This directory contains load images of all of the special processes which may be run, both kernel processes and supervisor-user processes. The kernel device driver processes are named according to the convention, '/prc/cdX' for character device drivers and '/prc/bdX' for block device drivers. Follow the convention established by the above table. Record device drivers such as magtape are included in block device drivers. Here X corresponds to the major device number as specified above in the '/dev' directory. Thus the control console device driver whose major device number is 0, is named '/prc/cda' and a character device driver whose major device number is 3, is named '/prc/cdd', and so on. The same is true for block device drivers; record device drivers are named the same as block device drivers. The '/prc' directory contains images of most of the common disk drivers and all of the tape drivers. These device drivers were constructed assuming the standard DEC device addresses:

FILE DEVICE ADDRESS hsprc RS03, RS04 fixed head disk RK03, RK05 moving head disk rkprc RP03 moving head disk rpprc RP04 moving head disk hpprc tfprc Telefile moving head disk tcprc **TU56** DECtape tmprc TU10 magtape htprc TU16 magtape

For the device drivers which are not in the '/prc' directory, you must first extract the MERT Source file systems as discussed under Section 20. Generating MERT Software and edit it accordingly.

8. Multiple Users (Time-Sharing Use)

If UNIX is to support simultaneous access from more than just the console terminal, we must edit one more file: *letc/ttys* (see *ttys*-V). For historical reasons *tty8* is the name of the console typewriter. To add new typewriters be sure the device is configured in *conf.d* and the special file exists in *dev*, 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 changes made to the *conf.d* file parameters. 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.

To summarize, each terminal on the system shows its presence in the software in three places:

A. special file entry in *ldev*.

B. line in *letclttys* for *init* to start it.

C. inclusion in minor device count in conf.d.

Corrections like steps A. and B. can be done without recompilation; a reboot will make them effective.

Further adaptation of the system to your installation requires the availability of the system source, which we will now make accessible.

9. The Source Distribution Tape

You should now be in a position to extract the various directories that constitute */src.* As indicated earlier, there is a (Bourne) shell script in root, *extract.sh* that has been provided for this purpose. It might be worthwhile at this point to examine the shell before proceeding further. The shell script is nothing more than a case statement whose entries execute various preformatted *cpio* commands. Familiarize yourself with the *cpio-I* command first.

Again, because of RK size constraints, it was necessary to divide the source directory. The size of of the cpio tape (*/src*) is around 12000 blocks. If you are not bound by RK size restrictions and can create a file system of this size, then create and mount it on */src* and execute the following command:

extract.sh all

This command extracts all the subdirectories from the tape that make up *lsrc*.

If you are an RK installation, or simply don't have the space, or don't care to have all of the source on-line at the same time, the shell *extract.sh* has been set up to extract subsets of the Source tape whose sizes are under 4000 blocks. The directory */src* contains the following subdirectories:

mertsrc	the MERT operating system source
cmd	the source code for all the commands
lib	the source code for all the libraries
mdec	the source code for the various boot programs
makefile	shell scripts to compile and install the user commands

It might be added that the other subdirectories also contain compilation shell scripts. More on that later.

To extract the source for the MERT operating system, create a file system of 4000 blocks and mount it on */src* and then execute the following command:

extract.sh mert

This command extracts all files and subdirectories that comprise *lsrc/mertsrc*. A complete description of this directory is given in Section 20 Generating MERT Software

The source code for the commands occupies more than 4000 blocks (6300 blocks to be exact). Therefore, the extraction of the directory *cmd* must be done in two parts. The command:

extract.sh cmd1

extracts all the files and subdirectories whose first letter falls in the interval a through 1. This command also extracts the directory *makefile* which can be used to remake and install anyone of the extracted commands. Likewise, the command:

Bell Telephone Laboratories, Incorporated - 12 - PROGRAM APPLICATION INSTRUCTION

extract.sh cmd2

extracts the remainder of the commands from the source tape, that is those files and subdirectories whose first letter falls in the interval m through z, and the directory makefile.

The organization of the directory *cmd* is similar to that of UNIX research, which we have adopted. Any command which is made (compiled) by processing a single source file exists as a **file** under the directory *cmd*. Any command that requires more than one source file exists as a **directory** under *cmd*. The shell *install.sh*, under the directory *makefile*, examines the name of each file or directory in *cmd*. If the name of the file ends in *.[clsy]* the appropriate processor (e.g. *yacc*) is called to remake the command. If the name of the command does not end with a **.[csly]*, the shell executes a file called *makefile* in the command's directory. It is up to the creator of the command to handcraft the makefile details. The reader is urged to examine the shell scripts under *makefile* and then a command that requires a directory (i.e. *cc*).

As an example, to extract the library and boot program source, you need only execute:

extract.sh libmdec

lsrc/lib contains the following subdirectories:

libc t	he c library source
liba t	he assembler library source
libS t	he standard I/O library source
libp t	he portable I/O library source
lib7 t	he UNIX version 7 system calls source
liby t	he yacc library source
libl t	he lex library source

lsrc/mdec contains all of the source for the MERT stand-alone boot and utility programs. The file *dboot.s* contains the source for all of the one block disk boot programs. The *tboot.s* file contains the source for all of the one-block tape boot programs. The boot programs for DECtape and for magtape must be in *lusr/mdec/tboot* and in *lusr/mdec/mboot*, respectively, for the *tp-I* program to work correctly.

Finally, if you wish to extract something other than the above, the any option has been provided

extract.sh any [pattern]

This extracts the files and/or directories that match the pattern and copies them into the directory *src.* To summarize: the successive execution of the commands:

extract.sh mert extract.sh cmd1 extract.sh cmd2 extract.sh libmdec

is equivalent to:

extract.sh all

After each extraction you are urged to perform a:

du -a

which lists all files under the current directory and their lengths. This allows you to check for the presence of zero-length files. (It is always good practice to do an additional check after executing so many file I/O operations.)

The other directories *lib, mdec*, and *mertsrc* also have shell scripts to recompile them. Most of the shell scripts use the *make-I* command, which is quite useful in recompiling a command (or an operating system) that consists of many parts. Throughout the directories *lib, mdec* and *cmd*, you will find many files called *make* or *makefile* or *Makefile*. These are *make* descriptor files and are processed by the command *make*. There are also shells with make-like names that are involved with the making or recompilation process. These shell files usually end with the suffix *.sh*. Not all of the compilation shells have been converted yet, hence the mixture of make descriptor files and shell scripts. These shell scripts or make descriptor files should be consulted whenever you need to recompile (e.g. to generate a non-floating point version). Please note that the various makeshells are set up to install the compiled commands etc. in */bin* and */usr/bin* directly. You should therefore save the old copies firstas an insurance against failure. Refer to Section 20 *Generating MERT Software* for further details.

10. The Manual File System

The third file system on the tape contains the manual pages and some other documentation. If you want to keep the manual on-line, you have to keep this file system mounted on *lman*. Make sure the *man* command (which is actually a shell file in *lusr/bin*) has as its first line a *chdir* command pointing to the mount point of the *lman* file system. You may want to remove some of the less suitable on-line documentation (like this section) or make *man* part of the root file system. The file system starts at block 8100 on the tape.

If you want to print the pages which have changed since the printing of the MERT Release 0 Manual, list the directory *changes*. These changes are also listed in the cover letter.

The file system contains directories man[0-8] for UNIX and man[a-g] for MERT manual sections, respectively. Directories man0 and manm0 contain ancillary files for the production of the UNIX and MERT Programmer's Manual portions of this manual. Directory manr0 holds ancillary files for the remaining sections like this one.

11. Changing the Disk Layout

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. The RK disk (or its image) as distributed has 4000 blocks for file storage, and the remainder of the disk (872 blocks) is set aside for swap space. In some systems, which allow many simultaneous processes, this amount of swap space is not quite enough, so a larger amount must be used for this purpose; a space large enough so that running out of swap space is unlikely should be chosen. Running out of swap space may cause random programs to produce a core dump. (This will be fixed in future releases of MERT).

Many common system programs (C, the editor, the assembler etc.) create intermediate files in the /tmp directory, so the file system where this is stored also should be made large enough to accommodate most high-water marks. In an idle state, we have about 1000 free blocks on the file system where /tmp resides, and hit the bottom every few days or so. (This causes a momentary disruption, but not a crash.) All the programs that create files in /tmp try to take care to delete them, but most are not immune to events like being hung up upon, and can leave files around. 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-l, threatening messages of the day and personal letters.

The efficiency with which MERT is able to use the CPU is somewhat 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. If only one controller is available with a number of drives, one should at least split the files among the different drives if simultaneous seeks are allowed by the disk driver.

12. Moving File Systems Around

Once you have decided how to make best use of your hardware, the question is how to rearrange the files to take advantage of it. If you have the equipment, the best way to move a file system is to dump it (dump-VIII) to magtape, use mkfs-VIII to create the new file system, and restore the tape. If you don't have magtape, dump accepts an argument telling where to put the dump; you might use another disk or DECtape. Alternatively, you can use the cpio-I command to move various subtrees about.

If you have only an RP disk, see rp-IV for some suggestions on how to lay out the information on it. The file systems distributed on tape, containing the binary, the source, and the manuals, are each only 4000 blocks long. Perhaps the simplest way to integrate the latter two into a large file system is to extract the tape into the upper part of the RP, dump it, and restore it into an empty, non-overlapping file system structure. If you have to merge a file system into another, existing one, the best bet is to use ncheck-VIII to get a list of names, then edit this list into a sequence of mkdir and cp commands which will serve as input to the Shell. (But notice that owner information is lost.)

13. Installing New Users

Install new users by editing the password file *letclpasswd* (passwd-V) when logged in as superuser. You'll have to make current directories for the new users and change their owners 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::10:1::/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. His default current directory is /usr/joe which has been created. The delivered password file has a number of users in it to be used as prototypes.

14. File System Maintenance

Periodically (say every day or so) and always after a crash, you should check all the file systems for consistency (fsck, icheck, dcheck-VIII). It is quite important to execute sync (VIII) before rebooting or taking the machine down. This is done automatically every 30 seconds by the update program (VIII) when a multiple-user system is running, but you should do it anyway to make sure; and you should not take the machine down while users are entering data.

Dumping of the file system 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 the

dump command (VIII) but restoration of individual files is painful. Dumping of files by name is best done by tp (I) but the number of files is limited. Finally if there are enough drives entire disks can be copied using cp-I, or preferably with the *pcp-e* program which copies tracks at a time. Note that there is no stand-alone program with MERT that will restore any of these formats. Unless some action has been taken to prevent destruction of a running version of MERT, you can find yourself stranded even though you have backup.

15. Odds and Ends

The programs dump, recdmn, fsck, icheck, dcheck, ncheck, and df (source in '/src/mert/kfs') should be changed to reflect your default mounted file system devices. Print the first few lines of these programs and the changes will be obvious.

Finally, be sure to read the introductory sections of this manual: Section 1 for this Release 0, and Sections 2 and 11 for the UNIX and MERT Programmers' Manuals, respectively.

Bell Telephone Laboratories, Incorporated - 16 - PROGRAM APPLICATION INSTRUCTION

PA-1C600-01 Section 19 Issue 1, October 1977 AT&TCo SPCS

This page has been left blank intentionally.