Installing and Using Research Unix Version 7
In the SimH PDP-11/45 and 11/70 Emulators

by Will Senn

Created December 7, 2015
Last Updated November 4, 2017

This is document revision 1.6. This note is intended to document the process of running Unix v7 in a PDP-11/45 emulated environment. It is based on the blog entry `https://decuser.blogspot.com/2015/12/installing-and-using-research-unix.html`. Another entry documents the process of running Unix v6[1]. This document assumes you are able to use the command-line skillfully.

1 Changelog

`[1.6 - 20171026] Fixed a typo and added clarification about 0 vs O in STTY command`

`[1.5 - 20171014] Added section describing multiple sessions and made some minor edits`

`[1.4 - 20171014] Added Appendix C - a notes section`

`[1.3 - 20171014] Added binmode to perl script to support windows and updated tested environments`

`[1.2 - 20171012] Minor wording corrections`

`[1.1 - 20171011] Created PDF version`

2 Preliminaries

2.1 Cross references

- Setting up Unix - Seventh Edition
  `http://www.tuhs.org/Archive/Documentation/PUPS/Setup/v7_setup.html`
- The Unix Heritage Society
  `http://www.tuhs.org/`
- The PDP Unix Preservation Society
  `http://minnie.tuhs.org/PUPS/`
- The Computer History Simulation Project
  `http://simh.trailing-edge.com/`
- Hellwig Geisse's package, notes, and utilities (mktape and exfs)
  `https://homepages.thm.de/~hg53/pdp11-unix/`
- Warren Toomey's work on restoration and archiving
  `http://minnie.tuhs.org/`

This note follows the approach laid out in the blog entry, *Installing and Using Research Unix Version 6 in SimH PDP-11/40 Emulator*[1]. It is written for anyone wishing to bring a v7 instance into life on modern hardware. It does not delve deeply into administrative details or installing additional software. At the end of the exercise, the reader will have enough knowledge at hand to bring a working system up and to explore it.

As alluded to in the previous entry, the so-called original bits that will be used are binary copies of original tapes that folks have donated to The Unix Heritage Society, TUHS. They are available to the public.

[1] `http://decuser.blogspot.com/2015/11/installing-and-using-research-unix.html`

2.2 Prerequisites

- The license - I am using the Caldera Unix Enthusiasts license available at `http://www.tuhs.org/Archive/Caldera-license.pdf`
- A working Host - I have used Mac OS X Mavericks through MacOS High Sierra 10.9-10.13, as well as FreeBSD 10.2 and 11. I feel confident that it would work with Linux as well, but I haven't tested it. Basically, any host capable of running SimH should work.
- SimH PDP-11 Emulator - I have used versions 3+, this note is based on using the latest code as of 20171011, version 4.0-0 Beta, available at `https://github.com/simh/simh`. See Appendix A for basic build instructions.
- A distribution tape image - I am using a tape constructed from Keith Bostic's tape records which appear to be closest to the bits of the original distribution. Available at `http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/`

2.3 Tested Environments

These notes are based on having a working SimH environment and a C-compiler installed, such as clang or gcc. The configurations that have been tested directly are:

- MacOS 10.13 High Sierra running on a MacBook Pro i7 Quad Core with 16GB RAM and XCode and Homebrew installed
- Mac OS X 10.11.1 El Capitan and MacOS 10.12.x Sierra running on a MacBook Pro i7 Quad Core with 16GB RAM and XCode and Homebrew (for gcc) installed
- FreeBSD 10.2 and 11 running on a Dell Optiplex 755 Core 2 Quad with 8GB RAM with gcc48 installed
- Windows 8.1 Enterprise running virtually in Linux Mint 18.2 on a HP Elite Desk 800 i5-6500 3.2 GHz with 16 GB RAM and Git bash with Unix tools installed
- Raspbian 2017-09-07-stretch running on a Raspberry Pi2 Model B

3 Installation

3.1 Preparing the tape image

This section describes creating a workspace, downloading the tape records, and constructing a bootable tape image.

1. Create a working directory (I use retro-workarea/v7 in my sandboxes folder).

```
mkdir -p ~/sandboxes/retro-workarea/v7
cd ~/sandboxes/retro-workarea/v7
```

2. Get a copy of Keith Bostic's tape records.

```
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f0.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f1.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f2.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f3.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f4.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f5.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/f6.gz
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/filelist
```

3. Unpack the six tape sections.

```
gunzip f?.gz
```

4. Create the tape image[2] for use with SimH.

   The SimH simulator will emulate a TU10 MagTape controller. We need to create a suitable tape image to load into the simulated TU10. We do this by combining each of the tape records along with their lengths and appropriate tape marks into a single file, named v7.tap.

   Download the mktape.pl[3] script.

```
curl -O http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/mktape.pl
```

5. Make the script runnable and run it.

```
chmod u+x mktape.pl
./mktape.pl
f0: 8192 bytes = 16 records (blocksize 512 bytes)
f1: 7168 bytes = 14 records (blocksize 512 bytes)
f2: 512 bytes = 1 records (blocksize 512 bytes)
f3: 11264 bytes = 22 records (blocksize 512 bytes)
f4: 11264 bytes = 22 records (blocksize 512 bytes)
f5: 2068480 bytes = 202 records (blocksize 10240 bytes)
f6: 9594880 bytes = 937 records (blocksize 10240 bytes)
```

6. To confirm that the correct number of records and block sizes were used, take a look at the filelist file we obtained with the tape sections and compare with what the mktape.pl script reported.

```
cat filelist
file 0: block size 512: 16 records
file 0: eof after 16 records: 8192 bytes
file 1: block size 512: 14 records
file 1: eof after 14 records: 7168 bytes
file 2: block size 512: 1 records
file 2: eof after 1 records: 512 bytes
file 3: block size 512: 22 records
file 3: eof after 22 records: 11264 bytes
file 4: block size 512: 22 records
file 4: eof after 22 records: 11264 bytes
file 5: block size 10240: 202 records
file 5: eof after 202 records: 2068480 bytes
file 6: block size 10240: 937 records
file 6: eof after 937 records: 9594880 bytes
file 7: block size 63:
```

7. To confirm that this tape is actually byte identical to the author's version, use openssl to check the sha1 digest.

```
openssl sha1 v7.tap
SHA1(v7.tap)= e6188335c0c9a3e3fbdc9c29615f940233722432
```

   With a working distribution tape image, we are ready to fire up the simulator and install v7 onto the RP06 diskpack.

3.2 Create a SimH ini file for the first boot

In the v6 blog entry, I showed both a manual and and ini file method of booting the simulation. In this note, I only show the ini method. However, I will describe the contents of the file. Just be aware that any commands in the ini file can be entered into the simulator interactively, if desired.

Below is the ini file that will be used during the installation of v7. After the system is installed, a modified version of the ini file will be used for subsequent boots.

[2]Or download it at `http://www.tuhs.org/Archive/Distributions/Research/Keith_Bostic_v7/v7.tap.gz`

[3]The source code for the script is in Appendix B.

The file sets the cpu and allows it to idle. A virtual RP06 moving head disk pack is attached to the simulator and associated with a file on the host as rp0, another is added as rp1, then a virtual TU10 magtape is added and associated with the distribution file we compiled on the host as tm0.

This ini file gives us the following initial configuration: A PDP-11/45 with two identical, empty, RP06 disk packs, and a TU10 magtape with our v7 distribution tape ready to load.

We use cat again to create the tape.ini file.

```
cat > tape.ini <<"EOF"
set cpu 11/45
set cpu idle
set rp0 rp06
att rp0 rp06-0.disk
set rp1 rp06
att rp1 rp06-1.disk
att tm0 v7.tap
boot tm0
EOF
```

3.3 Boot from the tape image

I highly recommend following along in *Setting up Unix - Seventh Edition* `http://www.tuhs.org/Archive/Documentation/PUPS/Setup/v7_setup.html` as you work through the rest of this document. *Setting up Unix* is authoritative for installing Unix v7. With the work we have done up to this point, if you were to leave out the boot tm0 command from the ini file, the instructions can be followed verbatim (talk about durable documentation). However, I will be explaining as we go, so you may want to wait until you have worked along with these instructions before trying to rely solely on the official document.

1. Start the SimH PDP Simulator with the distribution tape image.

   To run the PDP-11 simulator, we simply execute the simulator and pass the ini file to it.

   ```
   pdp11 tape.ini
   ```

2. Boot the distribution bootblock and let SimH create bad block tables for the RP06es.

   The simulator will start, process our ini file, and boot the tape. The simulator will ask if we want to overwrite the last tracks of each rp06. I believe that this is asking permission to write a bad block table on the last track. Answer y and press enter for both RP0 and RP1.

   ```
   PDP-11 simulator V4.0-0 Beta        git commit id: 247bd8d5
   Disabling XQ
   RP0: creating new file
   Overwrite last track? [N]y
   RP1: creating new file
   Overwrite last track? [N]y
   Boot
   :
   ```

   We don't have to key in a boot routine, because SimH has a built in tape rom that works fine. However, if we wanted to we could key in the TU10 boot routine as described in *Setting up Unix* manually and run it.

3.4 Create a filesystem on the first RP06

The simulator has started from the tape image and displays the word Boot followed by a carriage return and a colon prompt. At this point, no operating system is present. However a tape is loaded and a standalone program called *tm* is aslo loaded and available. If we run *tm*, it will run a program directly from the tape, indexed by the tape controller and file on the tape. It is a zero based index. We want to run the 4th file on the tape, which is a standalone version of *mkfs* in order to create a filesystem on the RP06 diskpack. The target device name is *hp* for the rp06. This time the index refers to the controller and the partition. We want to create a filesystem on the first controller's first partition. This will prepare our disk for a root filesystem.

```
: tm(0,3)
file sys size: 5000
file system: hp(0,0)
isize = 1600
m/n = 3 500
Exit called
Boot
:
```

3.5 Restore the root filesystem from tape

Again the Boot and : prompt will be displayed. Now that a filesystem is prepared, we will use another program from the tape, *restor*, to populate it. The standalone program *restor* will take a tape file as input and a disk device as output. The sixth file on the tape is a dump of *rp0* (a root filesystem). The same conventions as above apply regarding the indexes. Press return after *Last chance before scribbling on disk.* to proceed.

```
: tm(0,4)
Tape? tm(0,5)
Disk? hp(0,0)
Last chance before scribbling on disk.
End of tape
```

3.6 Boot the root filesystem

At this point, a root filesystem is available. We can boot Unix from the root. Using the indexing scheme described above, we can load and run the hptmunix (hp and tm drivers are included) kernel from the root filesystem.

```
: hp(0,0)hptmunix
mem = 177344
#
```

The system comes up single user. It also comes up with UPPERCASE characters and is slow to print output to the console, which is annoying and weird. We could enter multi-user mode and it would fix this and other annoyances, but because of our remaining low level tasks, it is simpler and more effective to stay in single user mode. In order to fix the annoying console issues while we complete the installation, we will use stty to set lowercase and to add no delays to newlines or carriage returns. Make sure that you type the number `0`, not the letter `O`, in the STTY command below. This will make our console snappy to respond and allow us to use both upper and lower case letters in the terminal.

```
# STTY -LCASE NL0 CR0
```

3.7 Remove unused kernels and clean up

Typing hp(0,0)hptmunix is a pain, let's shorten it to hp(0,0)unix and get rid of unused kernels in the root filesystem.

```
# mv hptmunix unix
# rm hp*ix
# rm rp*ix
# ls *ix
unix
```

3.8 Create device files

In order to use Unix to complete the installation, we will need to create a number of special files to represent our hardware devices. Special files serve as interfaces between the user and the underlying devices. They are the magic that allows the Unix system to present nearly all hardware to the user as simple files. The special file abstraction maps a filename to a memory vector that points to a limited set of common I/O operations (read, write, getchar, putchar, etc). The device drivers that implements these I/O operations are either part of the operating system, as is the case for all of our devices, or are supplied as add-ons. The makefile in /dev contains sections for the most common devices and serves as a template for us to determine what devices we need to instantiate.

```
The rp06 section in that file looks like this:
rp06:
        /etc/mknod rp0 b 6 0
        /etc/mknod swap b 6 1
        /etc/mknod rp3 b 6 7
        /etc/mknod rrp0 c 14 0
        /etc/mknod rrp3 c 14 7
        chmod go-w rp0 swap rp3 rrp0 rrp3
```

The syntax for mknod from man with edits is: `/etc/mknod name [c][b] major minor`

The first argument is the name of the entry. The second is b if the special file is block-type (buffered, block sized I/O, slower) or c if it is character-type (unbuffered, byte sized I/O, faster, aka raw). The last two arguments are numbers specifying the major device type and minor device (unit, drive, line-number).

For our purposes, we obtain the major device number from the makefile (6 for block-type rp and 14 for character-type rp). The minor device is a number represented by a byte that combines the block device index and the partition.

In order to know what partitions to use requires a bit of detective work, but I will just tell you that partition 7 is the largest available partition for us to use on the disk. The reference for the curious is HP(4), where the original authors describe the rp06 partition scheme.

So, here are the devices we will need to create special files for:

- The first rp06 partition 0 is root, partition 1 is swap, we will leave the rest unused for the time being
- The second rp06, partition 7 will be used for /usr
- The magtape tu10.

We will manually create block and character devices for each rp06, but we will use make to create the tape device.

According to the makefile, the rp06 major devices are 6 and 14, respectively for block and character devices. The minor numbers are:

- for the first drive, drive 0's first partition, it is 00000000, decimal 0
- for the first drive's second partition, it is 00000001, decimal 1
- although, we aren't using it, the first drive's seventh partition would be 00000111, decimal 7
- for the second drive, drive 1's seventh partition, it is 00001111, decimal 15

1. Create nodes for RP06 devices.

   Using this information results in the following commands, which we run to create the special files for the rp06 and give them appropriate permissions.

   ```
   # cd /dev
   # /etc/mknod rp0 b 6 0
   # /etc/mknod swap b 6 1
   # /etc/mknod rp3 b 6 15
   # /etc/mknod rrp0 c 14 0
   # /etc/mknod rrp3 c 14 15
   # chmod go-w rp0 swap rp3 rrp0 rrp3
   ```

   Now, rp0 refers to disk 0, partition 0 (the root device), swap refers to disk 0, partition 1, rp3 refers to disk 1, partition 6 (/usr), rrp0 refers to the raw disk 0, partition 0, and rrp3 refers to the raw disk 1, partition 6.

2. Create nodes for TU10 device.

   We use make to create the tape special files (regular device, rewinding device, and non-rewinding device) and set appropriate permissions.

```
# make tm
/etc/mknod mt0 b 3 0
/etc/mknod rmt0 c 12 0
/etc/mknod nrmt0 c 12 128
chmod go+w mt0 rmt0 nrmt0
```

   At this point you should have the following special files in /dev:

```
# ls -l
total 2
crw--w--w- 1 root    0,  0 Dec 31 19:02 console
crw-r--r-- 1 bin     8,  1 Jan 10 15:40 kmem
-rw-rw-r-- 1 bin       775 Jan 10 15:26 makefile
crw-r--r-- 1 bin     8,  0 Jan 10 15:39 mem
brw-rw-rw- 1 root    3,  0 Dec 31 19:02 mt0
crw-rw-rw- 1 root   12,128 Dec 31 19:02 nrmt0
crw-rw-rw- 1 bin     8,  2 Jan 23 16:35 null
crw-rw-rw- 1 root   12,  0 Dec 31 19:02 rmt0
brw-r--r-- 1 root    6,  0 Dec 31 19:02 rp0
brw-r--r-- 1 root    6, 15 Dec 31 19:02 rp3
crw-r--r-- 1 root   14,  0 Dec 31 19:02 rrp0
crw-r--r-- 1 root   14, 15 Dec 31 19:02 rrp3
brw-r--r-- 1 root    6,  1 Dec 31 19:02 swap
crw-rw-rw- 1 bin    17,  0 Jan 10 15:40 tty
```

3.9 Create a filesystem on the second RP06

With the devices attached and working and the special files representing them available, it is time to create a filesystem for the /usr partition and copy files from tape into the filesystem. We will use mkfs to create the filesystem and icheck to check the result.

```
# cd /
# etc/mkfs /dev/rp3 322278
isize = 65496
m/n = 3 500
# icheck /dev/rp3
/dev/rp3:
files      2 (r=1,d=1,b=0,c=0)
used       1 (i=0,ii=0,iii=0,d=1)
free  314088
missing    0
```

3.10 Restore the /usr filesytem from the distribution tape

We will use dd to move the tape to the appropriate starting point (skipping 6 files and setting the tape to point at the seventh file, a dump of rp3). Then we will use restor to restor the files in that tape file. Note that we are reading from nrmt0, the non-rewinding tape device.

```
# dd if=/dev/nrmt0 of=/dev/null bs=20b files=6
202+80 records in
202+75 records out
# restor rf /dev/rmt0 /dev/rp3
last chance before scribbling on /dev/rp3. [press enter]
end of tape
```

3.11 Mount /usr and copy the boot block on to the first RP06

Finally, copy a boot block from /usr to the first block of our first disks' root partition.

```
# /etc/mount /dev/rp3 /usr
# dd if=/usr/mdec/hpuboot of=/dev/rp0 count=1
0+1 records in
0+1 records out
```

3.12 Shut down gracefully

The installation is complete. It is a good idea to ensure that the superblocks of our filesystems are written (sync'ed) to disk before we virtually turn the power off.

```
# sync
# sync
# sync
# sync
```

Halt the Unix system by typing CTRL-E.

```
# CTRL-E
Simulation stopped, PC: 002306 (MOV (SP)+,177776)

Exit the simulator by tying q at the sim prompt:
sim> q
Goodbye
```

4 Post-installation

Next, we will create an ini file that is appropriate for normal booting. We'll also upgrade the system to a beefier PDP-11 in the process, log in as root and dmr and test a thing or two before calling it a day.

4.1 Create a normal single session boot SimH ini file

The following ini file is similar to the initial boot file, with the following changes. First, it includes comments and creates a PDP-11/70 with 2 Megs of memory, without requiring any addition changes or configurations to perform an upgrade. The second change is the removal of the tape, this is optional. The last change is that we boot directly from the rp06 instead of from tape.

```
cat > nboot.ini << "EOF"
echo
echo After Disabling XQ is displayed type in boot
echo and at the : prompt type in hp(0,0)unix
echo
set cpu 11/70
set cpu 2M
set cpu idle
set rp0 rp06
att rp0 rp06-0.disk
set rp1 rp06
att rp1 rp06-1.disk
boot rp0
EOF
```

4.2 Boot normally from the newly installed system

When running, note that the simulator does not initially provide a recognizable prompt, just type boot and the familiar colon prompt should appear.

```
pdp11 nboot.ini

PDP-11 simulator V4.0-0 Beta        git commit id: 247bd8d5

After Disabling XQ is displayed type in boot
and at the : prompt type in hp(0,0)unix

Disabling XQ
boot
Boot
: hp(0,0)unix
mem = 2020544
#
```

4.3 Log into multi-user mode as root

Rather than continuing on to single user mode, it is just a matter of pressing CTRL-D to obtain a properly configured multi-user mode console. root is the username and password.

```
# CTRL-D

# RESTRICTED RIGHTS: USE, DUPLICATION, OR DISCLOSURE
IS SUBJECT TO RESTRICTIONS STATED IN YOUR CONTRACT WITH
WESTERN ELECTRIC COMPANY, INC.
WED DEC 31 20:02:48 EST 1969

login: root
Password:
You have mail.
#
```

4.4 Read mail

Read mail to assure yourself that something (mail) works :)! When you're satisfied quit mail by typing `x` to keep message or `q` to not keep message.

```
# mail
From bin Thu Jan 11 19:28:15 1979
Secret mail has arrived.

? x
```

4.5 Create a .profile with sane TTY defaults

After you create the root .profile, logout by typing `^d`.

```
echo 'stty erase "^h" kill "^u" nl0 cr0'> .profile
# ^d
```

4.6 Become dmr and say hello, world[4]

In honor of the user dmr, login to the system as dmr and compile this most well known program as it was originally written, straight from the seminal work *The C Programming Language* by Brian W. Kernighan and dmr himself, Dennis M. Ritchie.

1. Login as dmr with no password.

   ```
   login: dmr
   $
   ```

2. Create a .profile for the dmr account, log out, and back in again.

   ```
   echo 'stty erase "^h" kill "^u" nl0 cr0'> .profile
   $ ^D
   login: dmr

   $
   ```

3. Create hello.c

   ```
   $ cat > hello.c <<"EOF"
   main()
   {
       printf("hello, world\n");
   }
   EOF
   ```

4. Compile hello.

   ```
   $ cc hello.c
   ```

5. Run hello.

   ```
   a.out
   hello, world
   $
   ```

[4]Suggestion by Warren Young

4.7 Shutdown

The system is now operational for single sessions. To exit, write the superblock and halt Unix, then exit the simulation.

```
$ sync
$ sync
$ sync
$ CTRL-E
Simulation stopped, PC: 002306 (MOV (SP)+,177776)

Exit the simulator by tying q at the sim prompt:
sim> q
Goodbye
```

5 Set up multiple TTY support

5.1 Create a multi-session capable normal boot SimH ini file

The following ini file is similar to the single session boot file, with the addition of lines enabling and configuring the DC11 device with 4 lines listening on the host system on port 2222. You can change the port to any unused port.

```
cat > mboot.ini << "EOF"
echo
echo After Disabling XQ is displayed type in boot
echo and at the : prompt type in hp(0,0)unix
echo
set cpu 11/70
set cpu 2M
set cpu idle
set rp0 rp06
att rp0 rp06-0.disk
set rp1 rp06
att rp1 rp06-1.disk
set dci en
set dci lines=4
att dci 2222
boot rp0
EOF
```

5.2 Boot from the single session ini file

```
pdp11 nboot.ini

PDP-11 simulator V4.0-0 Beta        git commit id: 247bd8d5

After Disabling XQ is displayed type in boot
and at the : prompt type in hp(0,0)unix

Disabling XQ
boot
Boot
: hp(0,0)unix
mem = 2020544
#
```

5.3 Log into multi-user mode as root

Rather than continuing on to single user mode, it is just a matter of pressing CTRL-D to obtain a properly configured multi-user mode console. root is the username and password.

```
# CTRL-D

# RESTRICTED RIGHTS: USE, DUPLICATION, OR DISCLOSURE
IS SUBJECT TO RESTRICTIONS STATED IN YOUR CONTRACT WITH
WESTERN ELECTRIC COMPANY, INC.
WED DEC 31 20:02:48 EST 1969

login: root
Password:
You have mail.
#
```

5.4 Reconfigure the system to support 4 DC11 lines

In order to connect to the system via the SimH dci device through telnet, unix needs to support the DC11 device and have tty's available and listening. This section is based on user aap's notes at `http://a.papnet.eu/UNIX/v7/Installation`.

```
# cd /usr/sys/conf
# rm l.o c.o
# cp hptmconf myconfnf
# echo 4dc >> myconf
# mkconf < myconf
# make
as - -o l.o l.s
cc  -c c.c
ld -o unix -X -i l.o mch.o c.o ../sys/LIB1 ../dev/LIB2
# sum unix
10314   106
# ls -l unix
-rwxrwxr-x 1 root    54122 Dec 31 19:09 unix
# mv unix /munix
# ed /etc/ttys
266
2,5s/./1/
w
266
q
# sed -n '1,6p' /etc/ttys
14console
10tty00
10tty01
10tty02
10tty03
00tty04
# /etc/mknod /dev/tty00 c 3 0
# /etc/mknod /dev/tty01 c 3 1
# /etc/mknod /dev/tty02 c 3 2
# /etc/mknod /dev/tty03 c 3 3
# chmod 640 /dev/tty0?
```

5.5 Shutdown

The system is now 'fully' operational for multiple sessions. To exit, write the superblock and halt Unix, then exit the simulation.

```
$ sync
$ sync
$ sync
$ CTRL-E
Simulation stopped, PC: 002306 (MOV (SP)+,177776)

Exit the simulator by tying q at the sim prompt:
sim> q
Goodbye
```

5.6 Boot from the multi session ini file

```
pdp11 mboot.ini

PDP-11 simulator V4.0-0 Beta        git commit id: 247bd8d5

After Disabling XQ is displayed type in boot
and at the : prompt type in hp(0,0)unix

Disabling XQ
Listening on port 2222
boot
Boot
: hp(0,0)munix
mem = 2019584
# RESTRICTED RIGHTS: USE, DUPLICATION, OR DISCLOSURE
IS SUBJECT TO RESTRICTIONS STATED IN YOUR CONTRACT WITH
WESTERN ELECTRIC COMPANY, INC.
WED DEC 31 19:12:15 EST 1969

login:
```

5.7 Telnet into the V7 instance

On the host system, open two additional terminal windows and telnet into the V7 instance on both.

```
telnet localhost 2222
Trying ::1...
Connected to localhost.
Escape character is '^]'.


Connected to the PDP-11 simulator DCI device, line 0


login: dmr
$
------------------
telnet  localhost 2222
Trying ::1...
Connected to localhost.
Escape character is '^]'.


Connected to the PDP-11 simulator DCI device, line 1


login: root
Password:
You have mail.
#
```

To end your sessions, type `^d to logout of the unix session and ^]` and `q` to exit the telnet session, then `^e` and `q` to end the SimH session.

5.8 Celebrate

Celebrate your successful installation and first use of a classic operating system of the 1970's and a job well done. Try out mail to communicate between users asynchronously and write to communicate synchronously.

Appendices

Appendix A - Build SimH from git repository hosted source code

1. Download the source code

```
git clone https://github.com/simh/simh
```

2. Compile the simulator

   You don't need SDL for this note, but it is a dependency for pdp11 that you may want to have. The build script will warn you about it, but it's up to you whether or not you want to go to the trouble of satisfying the dependency. I haven't needed, but YMMV.

```
cd simh
make clean
make pdp11
```

3. Install the simulator

```
cp BIN/pdp11 ~/bin/pdp11
```

Appendix B - mktape.pl source code

Use the cat command to create the mktape.pl script by copying and pasting[5] the following (up through the line that only contains the text *EOF* on to the command-line:

```
cat > mktape.pl <<"EOF"
#!/usr/bin/perl
use strict;
# Written by Will Senn. Create a bootable SimH tap file to install the system.
# Inspired by various Perl scripts and based on Hellwig Geisse's mktape.c
#
# modified 20171012 binmode required for windows use

my @files = ("f0", "f1", "f2", "f3", "f4", "f5", "f6");
my @blkszs = (512, 512, 512, 512, 512, 10240, 10240);

my $outfile = "v7.tap";

my $EOF = "\x00\x00\x00\x00";
my $EOT = "\xFF\xFF\xFF\xFF";

open(OUTFILE, ">$outfile") || die("Unable to open $outfile: $!\n");
binmode(OUTFILE);
for(my $i = 0; $i <= $#files; $i++) {
   my ($bytes, $blocksize, $buffer, $packedlen, $blockswritten, $file) = 0;

   $file = $files[$i];
   $blocksize = $blkszs[$i];
   $packedlen = pack("V", $blocksize);

   open(INFILE, $file) || die("Unable to open $file: $!\n");
   binmode(INFILE);
   while($bytes = read(INFILE, $buffer, $blocksize)) {
     $buffer .= $bytes < $blocksize ? "\x00" x ($blocksize - $bytes) : "";
     print OUTFILE $packedlen, $buffer, $packedlen;
     $blockswritten++;
   }
   close(INFILE);
   print OUTFILE $EOF;
   printf "%s: %d bytes = %d records (blocksize %d bytes)\n", $file,
$blockswritten * $blocksize, $blockswritten, $blocksize;
}
print OUTFILE $EOT
EOF
```

[5]PDFs are notoriously unreliable about copying and pasting. If it doesn't seem to work correctly, either type it in or paste it into an editor first and then copy and paste from there, after making any necessary fixups.

Appendix C - Various Notes

This section has helpful notes that don't fit anywhere else. They are generally system or situation specific so YMMV.

- Configure the Mac terminal so that the delete key provides backspace functionality.

  Open up terminal preferences and click on the default profile, or create a custom profile, click the Keyboard tab, and click + to add a key customization. Select Key: <-Delete, Modifier: None, Action: Send Text:, put cursor in the text box under Send Text: and type CTRL-h. It will be replaced with \010 (ASCII code for backspace). This change combined with the stty erase '^h' change we made for dmr and root will cause the terminal to behave as one might expect - type helli, hit delete, and the cursor will back up over the i. It won't delete the i from the screen, but when you type o, it will replace the i onscreen and in the input buffer with o.