Using V7/x86: A Brief Introduction

Robert Nordier
Nordier & Associates
www.nordier.com

October 3, 2007

1 Preliminary

V7/x86 is a relatively recent port of UNIX[1] Version 7 to the x86 (IA-32) based PC. The aim in what follows is not so much to provide comprehensive information as to give the reader a taste of using (and perhaps even a taste *for* using V7/x86), and show the kind of things that need to be done and the kind of approach that needs to be followed, in order to keep the system usable and operating in good health.

From purely a user standpoint, there is really not a great fundamental difference between V7 and a modern UNIX system. Most of the same basic commands are there, even though some of them are in different places. The system is, of course, simple and straightforward, rather than — in all its ramifications — vastly elaborate and complex, to take the contrasting example of (say) Solaris[2] which nevertheless developed from ultimately the same code base. For the user, the difference is really in the details. So, for instance, the V7 `cp` command has no options at all, rather than the half-dozen one might have grown used to. (The V7 `mv` command has no options either, and `cat` has just one.)

From a system administration perspective, the gulf between early and modern UNIX systems is much wider: few canned scripts, no menu-based tools, and of course no "guard rails" to protect the unwary from the consequences of their own assumptions. For this reason, the present document tends to focus on (in the broader sense) "admin"-related tasks: starting and stopping the system, making use of its features, and configuring various aspects of its behaviour.

[1] UNIX is a registered trademark of The Open Group in the US and other countries.
[2] Solaris is a trademark of Sun Microsystems, Inc.

2 Starting Up

2.1 At the Boot Prompt

The first thing you should see is the boot prompt

```
BOOT [hd(0,0)unix]: _
```

The second stage bootstrap (`/boot`) allows you to specify which kernel should be booted, and from which device and filesystem it should be loaded. Defaults appear between square brackets.

The syntax understood by the second stage bootstrap is

```
device(unit,offset)file
```

where "device" is a device name, "unit" is the device number, "offset" is the starting block number of the filesystem, and "file" is the file to load.

The bootstrap currently recognises two devices: `hd` (hard drive) and `fd` (floppy drive). Devices are numbered in the relative order they are recognised by the PC BIOS. The offset is from the start of the first V7/x86 partition, if the disk has "fdisk partitions", otherwise from the start of the disk.

So the default is to load the file "`unix`" from the filesystem at block 0 of the V7/x86 partition of the first physical hard drive.

The bootstrap tries to be flexible, and does not insist on an offset being specified if this is zero, so

```
hd(0)
```

is equivalent to

```
hd(0,0)
```

Along the same lines, if no filename is given, the bootstrap will assume the root directory is meant.

If the filename refers to a directory rather than a regular file, the bootstrap will obligingly print out a list of the contents, rather than attempt to load it.

So, entering

```
fd(0)
```

at the boot prompt is equivalent to entering

```
fd(0,0)/
```

and will select the first floppy drive and (assuming there is a diskette present with a valid V7/x86 filesystem) print out the contents of the root directory, for example:

```
. .. boot unix
```

There are no timeouts followed by automatic booting in the V7 world. So, until you respond to the boot prompt, the machine will just sit there, waiting.

2.2 Booting

Let us proceed to boot V7/x86 by entering

```
hd(0)unix
```

or

```
hd(1)unix
```

as appropriate, or by simply accepting the boot defaults, if valid.

When loading an executable file, the second stage bootstrap prints out section sizes, for example

```
text=43136  data=4576  bss=175892
```

and then a row of dots, corresponding to loaded disk sectors. It then hands over control to the loaded file.

Compared to subsequent incarnations of UNIX, V7 has a startup process that is extremely quick and simple. So, if all goes well, all you should see is a single line giving memory size information and then a prompt indicating that the system is in single-user mode, for example:

```
mem = 14016512
# _
```

2.3 In Single-User Mode

V7 does not automatically boot into multi-user mode, but requires an explicit "go ahead" from the user. It also make no provision for automatically checking filesystems, following a crash.

In single-user mode, only the root filesystem will have been mounted. However, if you were to request a list of mounted filesystems, the system might well claim that various filesystems *are* mounted, for example

```
# /etc/mount
hd0e on /usr
hd0g on /home
# _
```

This is because the file `/etc/mtab` may still be present, left over from when the system last ran. It is one of first tasks of `/etc/rc` to remove this file, but `/etc/rc` has not run yet. This highlights the point that, in V7 single-user mode, the user needs to know the underlying system fairly well, so as to know when to accept things at their face value and when not to.

It would probably be instructive to take a brief look at the `/etc/rc` file, at this stage. (It is preeminently one of the virtues of early UNIX that one can browse through the various files and make some kind of sense of many of them almost immediately.)

The following is a sample `/etc/rc` file, stripped of a few irrelevancies:

```
PATH=/bin:/usr/bin
rm /etc/mtab
cat /dev/null >/etc/utmp
/etc/mount /dev/hd0e /usr
/etc/mount /dev/hd0g /home
rm -f /usr/spool/lpd/lock
rm -f /usr/tmp/*
rm -f /tmp/*
/etc/update
date >/dev/console
/etc/cron
```

We can see the `rc` script removing the `/etc/mtab` file, the print spooler lock file, and clearing out the `tmp` directories; also truncating the file `/etc/utmp` to zero length. And that is already half the file. The remaining lines are devoted to mounting additional filesystems, and to starting the programs `update` and `cron`. Otherwise, the script prints out the date on the system console.

Since we are now at home with `/etc/rc`, we are also in a position to modify it when we need to. Suppose we needed to disable the mounting of the `/dev/hd0g` filesystem on `/home`. A quick approach would be to use the line editor `ed`, for example:

```
# ed /etc/rc
207
5s/^/: /
w
209
q
# _
```

where we tell `ed` to insert a colon plus space (": ") at the start of line 5 of the file, effectively "commenting out" (more or less) that particular command. The change, in context, looks like this:

```
/etc/mount /dev/hd0e /usr
: /etc/mount /dev/hd0g /home
rm -f /usr/spool/lpd/lock
```

Of course, we are not absolutely compelled to use `ed`, since V7/x86 comes with the full-screen editor `vi` (as well as the related line editor `ex`), but getting access to `vi` in this context would involve mounting the `/usr` filesystem and changing various environment variables and settings to allow editing in full-screen mode. Some basic `ed` skills are particularly useful in the V7 world, and most of all when one is involved on system administration tasks.

Before leaving single-user mode, let us take a quick look at checking filesystems. As already mentioned, the system does not, by default, do this itself, and regular filesystem checking – with particular care exercised following a system crash – is something the user needs to take responsibility for.

The UNIX filesystem checking program `fsck` was added late in the day to V7, and it is not completely supported by the filesystem structures. The result is that, although `fsck` does a good job of checking and repairing V7 filesystems, it also complains about incorrect information: information that is not in fact maintained by the V7 kernel at all.

A sample `fsck` session is shown below:

```
# /etc/fsck /dev/hd0g

/dev/hd0g
File System:  Volume:

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
FREE INODE COUNT WRONG IN SUPERBLK
FIX? n
** Phase 5 - Check Free List
FREE BLK COUNT WRONG IN SUPERBLK
FIX? n
190 files 5261 blocks 133736 free
# _
```

Notice the two questions and responses. There is actually no harm in allowing `fsck` to fix these counts, but since they are not maintained by the kernel, doing so will not stop further complaints by `fsck`.

2.4 Entering Multi-User Mode

To leave single-user mode, press `^D` (control D) at the single-user prompt. The `/etc/rc` script will then be run: and filesystems will be mounted, and whatever else done that happens to be contained in the `rc` script. Eventually the prompt

```
login: _
```

should appear on the console and on active terminals, and the system should be ready for normal use.

3 Shutting Down

3.1 Leaving Multi-User Mode

The system can be returned to single-user mode by sending a hangup signal to the `init` process. Since `init` always has a process identifier (pid) of 1, the following command as `root` will accomplish this:

```
# kill -1 1
# _
```

As can be seen, the system displays no particular indication that we are back in single-user mode.

At this point, filesystems can be unmounted ("by hand") and checked, if desired. A filesystem is unmounted by specifying the associated block device, for example

```
# /etc/umount /dev/hd0g
```

3.2 Power Off or Reboot

From single-user mode, it is necessary only to ensure that the system is "synced" before turning off or rebooting. Whether this should be done by

```
# sync
```

or

```
# sync;sync
```

or possibly

```
# sync;sync;sync
```

is one of the great imponderables of UNIX system administration and cannot be settled here. Though do ensure that the disk drive light has actually gone out before pressing the power button or reset switch.

3.3 Reentering Multi-User Mode

If you should be in single-user mode and decide to return to multi-user mode, rather than switch off or reboot, it is probably best to unmount the filesystems before doing so. Otherwise, the `rc` script (as already shown) will clear out the mount table, and mount attempts will fail (since the filesystems are already mounted). This will leave the `/etc/mtab` file empty, and no longer reflecting the state of the system.

4 Using the System

4.1 Logging In

There should be an account `guest`, with simple `.profile`, `.cshrc` and `.login` files, and the user could do worse than log in as `guest`, initially. If the Berkeley C shell is preferred, the command

```
exec /bin/csh
```

will make the change for the rest of the session. The account has no password, and from a present day perspective it may come as a surprise to discover that one can actually log in to most V7 accounts by default: for instance, to `sys` and `bin`.

4.2 Living With the Terminal Interface

The V7 terminal interface is an aspect of the system that probably will take some getting used to, for someone more familiar with modern UNIX systems. Most users will be conversant with the idea of pressing `^C` (control C) to interrupt a program; possibly using the Backspace key to erase the character to the left of the cursor; pressing `^U` (control U) to kill (discard) the current input line; and so on.

V7 dates from just before the time when use of screen-equipped terminals became standard, and the terminal interface reflects its origins in the days of typewriter-like teletype machines. On V7, one presses the Delete key to interrupt a program, the default erase character is '`#`', and the default kill character is '`@`'.

Terminal settings can be changed with the `stty` command. To use the Backspace key as your erase key, the command would be

```
stty erase '^H'
```

4.3 Working at the Console

The V7/x86 console driver `sc` provides a virtual terminal capability that allows the user to run full-screen programs making use of `/etc/termcap` and linked to the BSD `termlib` and `curses` libraries. Ensure that the environment variable `TERM` is set to the identifying string "`cons`", for instance by means of the Bourne shell commands

```
TERM=cons; export TERM
```

or the C shell equivalent

```
setenv TERM cons
```

The BSD full-screen editor `vi` and the pager `more` are in `/usr/ucb`.

4.4 Adding a Terminal

Whether it is an actual, dedicated terminal, or simply another PC running a communications program, it is a relatively easy matter to hook up a terminal to V7/x86.

For example, the following steps were taken to connect to V7/x86 from another PC running the popular Kermit package from Columbia University. The machines were joined using a minimal three-strand "null modem" cable through serial port 0.

On the V7/x86 side, the file `/etc/ttys` was edited to change the `tty00` entry

```
00tty00
```

to

```
12tty00
```

thus enabling `getty` on the port, and the system was taken down to single-user mode and back.

On the Kermit side, the following commands were entered

```
set line /dev/ttyd0
set carrier-watch off
set parity even
set stop 2
set speed 9600
connect
```

and which point the

```
login:
```

prompt appeared.

5 Availability

5.1 Distribution

The V7/x86 distribution is available from the V7/x86 home page at

`www.nordier.com/v7x86`

For typical configurations, setup is by means of a V7/x86-hosted, menu-driven install program run by booting from a CDROM or floppy disk.

5.2 Documentation

The two volumes of the *UNIX Programmer's Manual* are part of the distribution in source form (see `/usr/man` and `/usr/doc`). The `man` pages have in general been updated to reflect changes due to the V7/x86 port, but the additional (Volume 2) documents are unchanged. Various binary representations of the originals are readily accessible on the Internet.

A few V7/x86-specific documents are presently available from the V7/x86 home page.