1

WHAT IS THE SHELL?

When we speak of the command line, we are really referring to the shell. The *shell* is a program that takes keyboard commands and passes them to the operating system to carry out. Almost all Linux distributions supply a shell program from the GNU Project called `bash`. The name *bash* is an acronym for *Bourne Again Shell*, a reference to the fact that `bash` is an enhanced replacement for `sh`, the original Unix shell program written by Steve Bourne.

Terminal Emulators

When using a graphical user interface, we need another program called a *terminal emulator* to interact with the shell. If we look through our desktop menus, we will probably find one. KDE uses `konsole` and GNOME uses `gnome-terminal`, though it's likely called simply "terminal" on our menu. A

number of other terminal emulators are available for Linux, but they all do basically the same thing: give us access to the shell. You will probably develop a preference for one or another based on the number of bells and whistles it has.

Your First Keystrokes

So let's get started. Launch the terminal emulator! Once it comes up, you should see something like this:

```
[me@linuxbox ~]$
```

This is called a *shell prompt,* and it appears whenever the shell is ready to accept input. While it may vary in appearance somewhat, depending on the distribution, it will usually include your *username@machinename*, followed by the current working directory (more about that in a little bit) and a dollar sign.

If the last character of the prompt is a hash mark (#) rather than a dollar sign, the terminal session has *superuser* privileges. This means that either we are logged in as the root user or we've selected a terminal emulator that provides superuser (administrative) privileges.

Assuming that things are good so far, let's try some typing. Enter some gibberish at the prompt like so:

```
[me@linuxbox ~]$ kaekfjaeifj
```

Since this command makes no sense, the shell tells us so and gives us another chance:

```
bash: kaekfjaeifj: command not found
[me@linuxbox ~]$
```

Command History

If we press the up-arrow key, we see that the previous command `kaekfjaeifj` reappears after the prompt. This is called *command history*. Most Linux distributions remember the last 500 commands by default. Press the down-arrow key, and the previous command disappears.

Cursor Movement

Recall the previous command with the up-arrow key again. Now try the left- and right-arrow keys. See how we can position the cursor anywhere on the command line? This makes editing commands easy.

A FEW WORDS ABOUT MICE AND FOCUS

While the shell is all about the keyboard, you can also use a mouse with your terminal emulator. A mechanism built into the X Window System (the underlying engine that makes the GUI go) supports a quick copy-and-paste technique. If you highlight some text by holding down the left mouse button and dragging the mouse over it (or double-clicking a word), it is copied into a buffer maintained by X. Pressing the middle mouse button will cause the text to be pasted at the cursor location. Try it.

Don't be tempted to use CTRL-C and CTRL-V to perform copy and paste inside a terminal window. They don't work. For the shell, these control codes have different meanings that were assigned many years before Microsoft Windows came on the scene.

Your graphical desktop environment (most likely KDE or GNOME), in an effort to behave like Windows, probably has its *focus policy* set to "click to focus." This means for a window to get focus (become active), you need to click it. This is contrary to the traditional X behavior of "focus follows mouse," which means that a window gets focus when the mouse just passes over it. The window will not come to the foreground until you click it, but it will be able to receive input. Setting the focus policy to "focus follows mouse" will make using terminal windows easier. Give it a try. I think if you give it a chance, you will prefer it. You will find this setting in the configuration program for your window manager.

Try Some Simple Commands

Now that we have learned to type, let's try a few simple commands. The first one is `date`. This command displays the current time and date:

```
[me@linuxbox ~]$ date
Thu Oct 25 13:51:54 EDT 2012
```

A related command is `cal`, which, by default, displays a calendar of the current month:

```
[me@linuxbox ~]$ cal
    October 2012
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

To see the current amount of free space on your disk drives, enter **df**:

```
[me@linuxbox ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2             15115452   5012392   9949716  34% /
/dev/sda5             59631908  26545424  30008432  47% /home
/dev/sda1               147764     17370    122765  13% /boot
tmpfs                   256856         0    256856   0% /dev/shm
```

Likewise, to display the amount of free memory, enter the **free** command:

```
[me@linuxbox ~]$ free
             total       used       free     shared    buffers     cached
Mem:        513712     503976       9736          0       5312     122916
-/+ buffers/cache:     375748     137964
Swap:      1052248     104712     947536
```

Ending a Terminal Session

We can end a terminal session by either closing the terminal emulator window or entering the `exit` command at the shell prompt:

```
[me@linuxbox ~]$ exit
```

THE CONSOLE BEHIND THE CURTAIN

Even if we have no terminal emulator running, several terminal sessions continue to run behind the graphical desktop. Called *virtual terminals* or *virtual consoles*, these sessions can be accessed on most Linux distributions by pressing CTRL-ALT-F1 through CTRL-ALT-F6 on most systems. When a session is accessed, it presents a login prompt into which we can enter our username and password. To switch from one virtual console to another, press ALT and F1–F6. To return to the graphical desktop, press ALT-F7.

2

NAVIGATION

The first thing we need to learn (besides just typing) is how to navigate the filesystem on our Linux system. In this chapter we will introduce the following commands:

- `pwd`—Print name of current working directory.
- `cd`—Change directory.
- `ls`—List directory contents.

Understanding the Filesystem Tree

Like Windows, a Unix-like operating system such as Linux organizes its files in what is called a *hierarchical directory structure*. This means that they are organized in a tree-like pattern of directories (sometimes called folders in other systems), which may contain files and other directories. The first directory in the filesystem is called the *root directory*. The root directory contains files and subdirectories, which contain more files and subdirectories, and so on.