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 `bash` is an enhanced replacement for `sh`, the original Unix shell program written by Steve Bourne.

Terminal Emulators

When using a graphical user interface (GUI), 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 basically do the same thing; give us access to the shell. You will probably develop a preference for one or another terminal emulator based on the number of bells and whistles it has.

Making Your First Keystrokes

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

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

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

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

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

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

Because this command makes no sense, the shell tells us so and give us another chance.

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

Command History

If we press the up-arrow key, we will see that the previous command `kaekfjaeifj` reappears after the prompt. This is called *command history*. Most Linux distributions remember the last 1000 commands by default. Press the down-arrow key and the previous command disappears.

Cursor Movement

Recall the previous command by pressing the up-arrow key again. If we try the left and right-arrow keys, we'll 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 on 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.

Note: Don't be tempted to use `Ctrl-c` and `Ctrl-v` to perform copy and paste inside a terminal window. They don't work. These control codes have different meanings to the shell and were assigned many years before the release of Microsoft Windows.

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 on it. This is contrary to the traditional X behavior of "focus follows mouse" which means that a window gets focus just by passing the mouse over it. The window will not come to the foreground until you click on it but it will be able to receive input. Setting the focus policy to "focus follows mouse" will make the copy and paste technique even more useful. Give it a try if you can (some desktop environments such as Ubuntu's Unity no longer support it). 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 enter text in our terminal emulator, let's try a few simple commands. Let's begin with the `date` command, which displays the current time and date.

```
[me@linuxbox ~]$ date
Thu Mar  8 15:09:41 EST 2018
```

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

```
[me@linuxbox ~]$ cal
     March 2018
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 our 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, by entering the `exit` command at the shell prompt, or pressing `Ctrl-d`.

```
[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., We can access these sessions, called *virtual terminals* or *virtual consoles*, by pressing `Ctrl-Alt-F1` through `Ctrl-Alt-F6` on most Linux distributions. 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-F1` through `Alt-F6`. On most system we can return to the graphical desktop by pressing `Alt-F7`.

Summing Up

This chapter marks the beginning of our journey into the Linux command line with an introduction to the shell and a brief glimpse at the command line and a lesson on how to start and end a terminal session. We also saw how to issue some simple commands and perform a little light command line editing. That wasn't so scary was it?

In the next chapter, we'll learn a few more commands and wander around the Linux file system.

Further Reading

- To learn more about Steve Bourne, father of the Bourne Shell, see this Wikipedia article:
  http://en.wikipedia.org/wiki/Steve_Bourne
- This Wikipedia article is about Brian Fox, the original author of `bash`:
  https://en.wikipedia.org/wiki/Brian_Fox_(computer_programmer)
- Here is an article about the concept of shells in computing:
  http://en.wikipedia.org/wiki/Shell_(computing)