Introduction to Unix and the X Window System

Scott D. Anderson*
Wellesley College
Scott.Anderson@acm.org

1 Overview

Unix™ is an old and venerable operating system that has also managed to grow with the times. It currently runs on almost any kind of computer, from PC hardware to big mainframes and supercomputers. Well worth learning, for any computer scientist.

The X Window System is a graphical user interface (GUI) that runs on most Unix machines. It has some nice advantages over a Microsoft Windows or Apple Macintosh GUI, primarily in being able to run over networks. Again, this is well worth learning. Indeed, a computer scientist with any breadth of knowledge would be expected to understand Unix and the X Window System. (The latter is often abbreviated to "X Windows" or even just "X.")

There is too much about both of these topics for me to do more than scratch the surface. My goal is just to give you enough to get started and be able to run a few simple commands, write simple programs, and learn more.

2 Unix and Linux

Linux is an operating system that was intended to (1) work just like Unix, and (2) run on inexpensive IBM PC hardware (the Intel 80x86 family of chips and the PC clones). It technically isn't Unix because "Unix" is a trademarked name,[1] but you won't find any practical differences. In practice, the word "Unix" is an umbrella term that covers many variations on the Unix idea, including Linux.

Because Linux runs on PC hardware, it becomes a viable alternative to Windows NT, 95, 98 and 2000. There are also version of Linux that run on Macintosh hardware, so it becomes an alternative to the Macintosh operating system, MacOS. However, the impact of Linux has been much greater in the server market than the desktop market, primarily because of its speed and reliability: Linux servers run 24/7 for months without needing to be rebooted.

Linux is also part of the Open Source movement, which means different vendors and volunteer developers share source code and build on each other's work. Thus, you can buy Linux from many different companies, or even download it for free. In the Linux lab, Wellesley has about 13 PCs running Linux. What I describe here runs on those computers, although almost everything runs on any version of Unix.

3 Logging In

When you come to a Linux machine, say "Teddy," there will be a login prompt, like this:

```
Red Hat Linux release 7.3 (Valhalla)
Teddy login:
```

You type your login name (typically the same as your First Class username) and your password. People may call this your "Puma" password, but in fact the same password works for all the machines. Your password doesn't have to be the same as your First Class password, and probably shouldn't be.

Once you give your password, you'll get a simple text interface (like DOS, if you know what that looks like). This is quick and effective if you just want to do a few operations, such as printing a file or sending a short email, but if you are doing much work, you'll want to start up the X Window System. Issue the following command:

```
startx
```

The X Window System will start up, and you'll have a desktop that looks a little like an MS Windows desktop. Unlike Windows, however, different window managers have been developed for Linux. We use one called KDE (the K Desktop Environment). It is recognizable because there will be a big K in the lower left corner. This is an icon similar to the "Start" button in MS Windows. You'll also see some icons along the bottom (which is called the "panel"). Using the K menu, select "Utilities" and "terminal." Equivalently, you can click on the panel icon that looks like a monitor partly covered by a seashell. Both of these are ways of starting up a "shell."

*Thanks to LeeAnn Tzeng for helpful comments and to Agnes Ampadu for resurrecting this document that I feared was lost forever.

[1] Unix is a trademark of AT&T Bell Laboratories.

4 Shell Commands

Back in the olden days, when you logged into a computer, there were no windows, and you typed commands to a "command line" and results were typed back to you (or put into files). You still get that when you log in before you start X Windows and when you login remotely using Telnet. Unix still bears that legacy, and the place where you type commands is called "the shell." (It's called the shell in contrast to "the kernel," which is the core of the operating system; the shell is a command-line interface to the kernel.) Nowadays, each shell runs in a window and you can have as many shell windows as you like.

The shell *prompts* you for input and responds to your commands. You can customize your prompt. Here at Wellesley, we've defined it to be your username and host (machine name) and the name of the directory you are in. If I'm logged into Teddy and I'm in my `public_html` directory, my prompt would look like:

```
[anderson@teddy public_html]
```

In this document, however, I will pretend that the prompt is a percent sign.

Here are a few of the shell commands that people use every day:

ls This prints a listing of all the files in a directory (called "folders" in other operating systems). By default, it lists the files in the *current* directory, but with a command-line argument, it lists the files in the directories named on the command line..

pwd Prints the complete name of the current working directory; in other words, what the current directory is.

mkdir This creates a new directory, contained in the current one. That is, the new directory is a sub-directory.

cd This changes your current directory. In other words, it moves you from one place to another, like changing your location.

cp This copies a file from one place to another.

mv This moves a file from one place to another (or one name to another). You can rename a file by using `mv`.

rm This removes (deletes) a file. *Warning:* you can't get it back again!

rmdir This removes (deletes) a directory, but only if it's empty.

Now, let's see these in action. See if you understand what is happening at each step here. Afterward, I'll go over them and interpret. Here's a session by Wendy Wellesley:

```
% ls
HelloWorld.class  public_html
HelloWorld.java   read-only
% pwd
/students/wwellesl
% cd public_html
% pwd
/students/wwellesl/public_html
% ls
% mkdir newdir
% ls
newdir
% cd newdir
% pwd
/students/wwellesl/public_html/newdir
% cd ..
% rmdir newdir
% cd
% pwd
/students/wwellesl
% ls -1a
.
..
.bash_profile
.bashrc
.emacs
HelloWorld.class
HelloWorld.java
.kde
.login
.profile
public_html
read-only
% cp .profile dot-profile
% ls
dot-profile       public_html
HelloWorld.class  read-only
HelloWorld.java
% mv dot-profile renamed-file
% ls
HelloWorld.class  read-only
HelloWorld.java   renamed-file
public_html
% rm renamed-file
```

The first `ls` shows that we have two things in our current directory. The `pwd` shows us that the current directory is `/students/wwellesl`. It happens that `public_html` is a directory within our current directory, so we can `cd` into it. The `pwd` shows that the `cd` command worked. The second `ls` shows that the `private` directory is empty. We make a sub-directory and use `ls` to check that it exists. We can `cd` to it, as the `pwd` confirms. The Unix file structure is a tree, just like all computers, with the directories in a path separated by slashes (Windows uses backslashes and MacOS uses colons).

The `cd..` changes to the directory above the current one. (The ".." is a special name for the parent directory.) This brings us back to our `public_html` first directory. The `lsnewdir` shows that `ls` can be followed by a directory name, to list the contents of that other directory. Since `newdir` is empty (we just created it, after all), we can use `rmdir` to remove it.

The bare `cd` command changes to the original directory, also called the "home" directory. We confirm this with `pwd`. We then do an `ls-al` and discover that there are additional, invisible files in our home directory. The `-a` option to `ls` means to show *all* files. Usually, `ls` hides any files whose name begins with a dot. The `-1` option means to give a listing in one column, which I did just for convenience in this document; in real life, `-1` is rarely used. However, `-l` (the letter "l," for a "long" listing) is often used and gives a lot more information about each file.

We can use `cp` to copy one of these files, use `mv` to rename (or move) the copy to a new name, and finally use `rm` to delete the copy.

5 Creating/Editing Files

Having seen how to rename and remove files, how do we create them in the first place? This is done with an "editor," which is any program that allows you to change the contents of a file. One of the two standard editors in Unix is Emacs. (The other is `vi`.) I've written a separate document introducing Emacs, so I will describe it even more briefly here.

There are quite a few ways to start Emacs, depending on what effects you want. I'll just tell you two.

- If you're physically at the machine, called using the *console*, go to the K menu, go to the Applications menu (the top item) and select Emacs. That starts up a new window running Emacs. You can use the mouse to switch back and forth between any shell windows and your Emacs window.
- If you're logged in remotely via ssh, and you're on a Unix machine running X Windows, you can start up a new window running Emacs by typing its name as a command, followed by an ampersand.

  ```
  % emacs &
  ```

- If you're logged in remotely via telnet (ssh is better than telnet because it's secure, but telnet is more widely available), so you only have a single shell and no desktop, you can run Emacs by typing its name as a command:

  ```
  % emacs
  ```

You can move around using the arrow keys, and you can type stuff into your file. You can save the file with a particular name using the "C-x C-w" command (hold down the "control" key while typing "x" then "w"). This is what other programs call "Save As." To just save a file you're working on, without specifying a name, type "C-x C-s." You can also access these commands via the "Files" menu.

I highly recommend running the Emacs tutorial. To start the tutorial, type "C-h t," which means to hold down the control key while typing "h," then release the control key and type a "t."

Note: a single Emacs can edit any number of files, so unless you have a good reason to, it's not necessary to start up multiple Emacs applications, and it will just slow down your machine unnecessarily. Instead, switch among different Emacs buffers.

6 Compiling C Programs

There are two ways to compile a C program. There's the old-fashioned way, and the spiffy new way (actually, both are older than you are). I'll explain both. For concreteness, let's assume you're editing a file called `foo.c`.

The old-fashioned way is to switch to the shell window, make sure you're *in* the directory that contains `foo.c` (use `cd` if you're not, and `ls` to check), and run the C compiler, which is called `cc`. Either of the following will do it:

```
% cc foo.c -o foo
% make foo
```

The first command compiles foo.c and puts the binary (executable) file in `foo`. Note that `foo` is *not* the same thing as `foo.c`! One is your source code and the other is your executable code. If you make a mistake and accidentally do:

```
% cc -o foo.c foo
```

the compiler will destroy your source code. So, be careful!

The second example, `makefoo`, is just like the first, but easier to remember and type, and won't accidentally overwrite your source code. The program called `make` attempts to figure out how to run `cc` in order to make `foo`, which is what you wanted. Highly recommended.

The new-fangled way to compile a C program is to ask Emacs to run it for you. Type the following in Emacs: "M-x compile RET." If you've run the tutorial, you know that "M-x" means to hold down the "meta" key (usually "alt") and type an "x," then to type "compile" in the mini-buffer, and finally hit the "return" or "enter" key. If the "alt" key doesn't work, use the "ESC" key.

That command will then prompt you for the compilation command, which defaults to some kind of "make." You can fill in the rest of the "make" command, or replace it with the "cc" command.

The reason this is cool is that if there are any compilation errors, Emacs will bring you to the next erroneous line when you type `C-x‘`: That's "C-x" followed by an open quote (not an apostrophe).

7 Running Your Programs

Once you've compiled your program and you have a binary file called `foo`, you can run it by switching to your shell window and typing:

```
% ./foo
```

The "./" is necessary for security reasons that are hard to explain.

8 Using Removable Disks

Using a diskette (floppy disk) or a Zip disk on a Macintosh or Windows machine is very easy. It's a bit more complicated under Unix, but not enough to keep you from doing so. You should keep a second copy of your programs and such on a removable disk, just in case you accidentally delete a file. Every computer scientist will tell you about the time he or she inadvertently deleted a valuable file or lost hours of editing due to an untimely power outage. Don't let it happen to you! Learn from our sad experience! In short, making duplicate copies of your work will probably, at some unknown point in the future, save you inordinate grief.

You don't have to use removeable disks in order to work on your program using different computers. Thanks to the wonders of NFS (the Network File System), all of your files live on Puma, our file server, and you can login to any of the Linux machines and your files will appear there. Of course, if you want to take your work to your home Linux computer (remember, you can install Linux on your home machine), a removeable disk is helpful.

Oh, I'm sorry to say that I don't know how to mount a Mac-formatted Zip disk onto a Linux machine. So, this section assumes PC-formatted disks.

The first issue is "mounting" the disk. You see, with Windows, the diskette is forced to be the A drive and the Zip disk is forced to be, say, the D drive. On a Macintosh, the disks appear on the desktop. In Unix, they can be added anywhere in the file system (they could even be a sub-directory of your home account), but they usually go in a directory called `/mnt`, which is short for "mount."[2] So, you stick your disk into the slot and issue one of the following commands:

```
% mount /mnt/zip
% mount /mnt/floppy
```

When your disk is mounted, the contents of those two directories are equated with your disk. So, you can `ls` those directories, `cd` to them, `cp` and `mv` files to and from them. You can even `mkdir` and `rmdir` directories. In short, you can do anything to the disks that you can do with your own files and directories, using the same commands.

When you're done, you can unmount the disks and pop them out:

```
% umount /mnt/zip
% umount /mnt/floppy
```

9 Logging Out

When you're done with a Linux machine, don't just walk away. You must log out so that others can use the machine. If you're at the K desktop, there is a menu item in the "K" menu for logging out, or you can use the icon on the panel. The icon isn't easy to describe, but if you put your mouse over any icon, a balloon will pop up to give a text description, and you can find it easily enough. If you're logged in via telnet, you should use the `logout` command:

```
% logout
```

If you're at the console, logging out only brings you to where you typed `startx`, so remember to give the `logout` command there.

10 Learning More

To learn more about a particular Unix command, the best source is the online manual. For example, to learn more about the `ls` command, type:

```
% man ls
```

Warning: The "man" pages are reference material, not tutorials or easy reading, which means they can get a little *dense* sometimes, so skim them at first reading, and go back for more when you've swallowed that. Reading man pages is another skill that is well worth acquiring, because software changes, and even experienced computer scientists check the man pages regularly.

To learn more about Emacs, use the built-in help system. Try "C-h i" to start up the "info" pages, and then learn about that. Emacs is enormous, so don't try to learn too much at once. Start with the basics, then gradually add new things.

A web search will turn up tutorials for Unix and Emacs. Also, there are copies of *The Linux User's Guide*, by Larry Greenfield, scattered about the lab. It's well worth dipping into as needed, though it's a bit thick to sit and read all at once. I recommend chapters 1, 4, 6, and 8 to start.

If you think of any important information or advice to add to this primer, please let me know.

[2] Some Linux machines are configured so that they go in `/`, so they are mounted as `/zip` or `/floppy`. My directions are for the Wellesley machines.