The Linux System

Written by Dave Pawlowski, September 11, 2012

Introduction

By now you should have been introduced to Linux in a basic sense, learned how to perform basic operations within the operating system, login, create and remove files, and such. Before you learn more about the system, it is important to have an understanding of the file system, where and how to find things, and a little bit about what is going on in the background when you type a command.

The file system

Every Linux computer has a similar filesystem hierarchy in which there are standard directories where certain files are placed. It is important to note that

Everything in Linux is a file

Again, everything in Linux is a file. The reason your mouse or printer works, the reason the terminal spits out a bunch of information when you type ls -l, and the reason that you are transferred to another computer when you use ssh is because there is a file somewhere that does something when you perform these tasks. Being able to locate the files, or figure out where they should be is important if you want to be able to use a Linux system effectively. So, lets explore the filesystem a little.

The root directory

When you login to a Linux system, you are placed inside your home directory. This is by no means the highest directory in the hierarchy, nor the most important. If you type the command:

```
% pwd
```

and hit enter, you will see that you reside where you expect, i.e. for me, in the directory /Users/dpawlows. As you know already, typing

```
% cd ..
```

puts you up one level in to the /Users directory. But that's not the highest level in the hierarchy. There's one level more. Type

```
% cd ..
```

to get there. If you type

```
% pwd
```

now, you should see that you are in the directory "/". This is a special directory called "the root directory" or just "root". This is as high as you can go in the filesystem. Every file on this computer is contained somewhere in this directory or one of its subdirectories.

Every Linux machine has several subdirectories in root that are the same or at least very similar. Figure 1 shows an example of the top levels of a typical Linux file hierarchy. If you are on a Mac, you will see a few of the same directories in root (/bin, /usr,

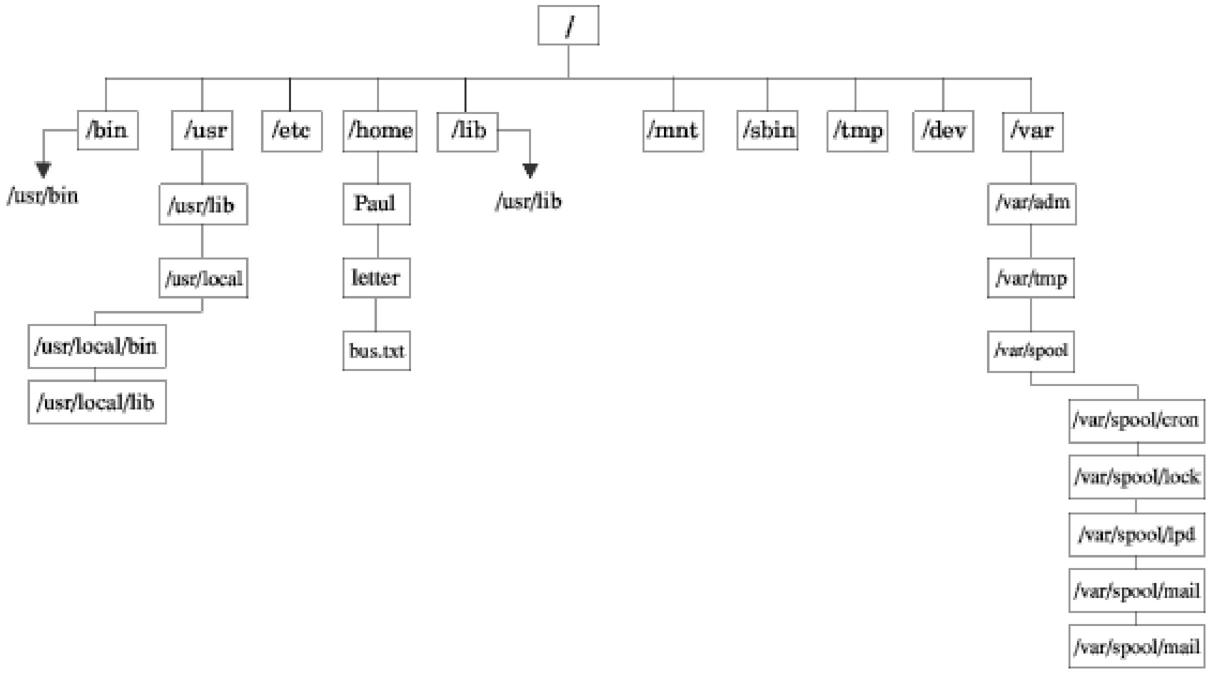

Figure 1: An example of the top levels of a typical Linux filesystem

/etc, /sbin, /dev, /var). Below are a few notes on some of these directories. For more information, the command

```
% man hier
```

will bring up a man page that describes the Linux hierarchy.

Directories

/bin

Contains fundamental executable files (commands or programs) that are used by users.

Only the most basic ones are here. Incidentally, the command

```
% which 'commandname'
```

will tell you where an executable actually resides.

/usr

Contains files and sub-directories which may be shared between machines. Important subdirectories include
- /usr/bin: Contains most of the executables on your system, e.g. emacs, man, sort.
- /usr/lib: Contains programming libraries- collections of program routines.
- /usr/local: Contains user installed programs and files.
- /usr/include: Contains include/header files. Primarily for the C language.

/etc

Most of the configuration files are installed here, such as those needed for the shell, ssh, web server, etc. These files are used every time you start/use one of these things.

/home

Contains home directories for all the users on the system. On a Mac this is called /Users.

/sbin

Executable files which are for system administration are stored here, as opposed to programs used by users in general. For example, the command shutdown resides here, which is a command that properly shuts the system down.

/dev

Contains files which give access to devices like the keyboard, mouse, screen, bluetooth, etc.

/var

Contains output and "temporary" files such as log files and email messages.

Others
In addition to these, Apple computers have other directories that are included in the root directory, most of which are needed for the graphical user interface (GUI) side of the OS or applications that are installed when using the GUI.

The pathname

There are two ways to enter a pathname to a file or directory. The absolute path of a file or directory is the path that you would get if you typed the pwd command. It starts from root and goes down the file hierarchy to your cwd.

For example, enter

```
% cd /usr/local/itt/idl/idl80
```

You just navigated to the main directory for the installed program, IDL, by entering the directory's *absolute path*.

Say you are already in /usr/local/. Then, you could have simply entered

```
% cd itt/idl/idl80
```

and you would have ended up in the same place. In this case though, you would have changed directories using the *relative path*. The relative path is based on the fact that the file or directory in the right most position of the path that you entered actually resides in the directory you are in. In other words, it is relative to your current position in the filesystem.

For example, had you entered

```
% cd itt/idl/idl80
```

from your home directory, you would have received an error, because the file ~/itt/idl/idl80 doesn't exist.

Entering a path either way is fine. It is just sometimes more convenient to enter a relative path, since it is almost always shorter. However, when using a pathname when creating a program or script, it is usually a good idea to use the absolute pathname, as you may want to run the program from any directory in the filesystem, which means that the relative path would usually not work.

Wildcards

One of the most useful features of Linux are called wildcards. Think of these as placeholders for omitted letters or numbers. For example, say you are looking for a file, but aren't sure if you named it *physicsprogram* or *physicsscript*, you can include a wildcard to stand for the part of the filename that you are uncertain of. In otherwords, you can list the files of a directory with

```
% ls phyics*
```

which would list all files that start with the letters *physics*. This could include, if they existed, *physics*, *physicsprogram*, *physicspants*, or *physics_help*.

You can use wildcards for just about any purpose in Linux, and many programming/scripting languages (such as IDL, python, and perl) understand them as well. Here are a few guidelines:

• Use ? as a placeholder for **one** character or number.
• Use * as a placeholder for **zero** or more characters or numbers.
• You can include a wildcard at any place in a name; the beginning, the middle, the end, it doesn't matter. Also, feel free to use them in multiple places! i.e. *physics*.
• The ˆ character can tell Linux to exclude a character

As an example enter

```
% cd /usr/bin
```

Try listing all files that begin with z. What about all files that have the word "mirror" in them?

What happens if you do this:

```
% ls z[ˆc]*
```

A last note on the filesystem

Because a user's home directory is used so frequently, it is treated specially by Linux. To see this, navigate away from your home directory, if you haven't done so already. Now, enter

```
% cd
```

Right, nothing after the cd command. After hitting enter and typing

```
% pwd
```

you will see that you are in your home directory. Linux assumes if you do not enter an argument after the cd command, you want to go to your home directory.

Now, type

```
% cd -
```

(that's a hyphen). This should take you to the last directory that you were in.

Now, lets say you wanted to go to your Documents directory. You could cd to your home directory, then cd again to Documents. Or you could enter the absolute path, beginning with /Users/...

But the easiest way is to enter

```
% cd ~/Documents
```

Linux assigns the character ~ to represent your home directory. In other words, typing ~ is exactly the same as typing /Users/dpawlows/ (in my case). This is simply to make life easier, as all the directories that you will create and use will reside beneath your home directory.

The Shell

One of the most important aspects of the Linux system is the shell. The shell is the interpreter that deals with all those commands that you enter on the command-line. When you open a terminal, you are gaining access to a shell session. The shell determines how you enter and re-enter commands. One of the great features about Linux, is there are different shells that you can choose from, and while having the same set of general features, all have their own unique features and capabilities that may be useful depending on your situation.

In order to find out which shell you are using, you can use the command

```
% echo $SHELL
```

Note that capitalization matters on most Linux systems, Darwin (Mac's Linux flavor) being the exception

The echo command tells Linux to display information about a variable, in this case the variable is SHELL. The $ character is Linux's way of differentiating between a word and a variable (something that stores information).

The system's response to the echo call above will be to display the full path to your shell; something like /bin/tcsh or /bin/bash. Those two are the most common, but there are others. bash stands for Bourne Again Shell, an extension of the original unix shell, sh (Bourne Shell), while tcsh is an extension of csh, which was designed to incorporate some C style programming-syntax features into the shell.

Other available shells are ksh and zsh, both of which have features that may or may not be desirable. Typically, the shell you start learning with becomes the one you stick with.

You can see which shells are available to you if you type

```
% cat /etc/shells
```

and if you would like, you can change your shell with the chsh command.

Common features

There are a few features that are extremely convenient and available in most shells.

Tab completion

cd into your home directory and type (but don't hit enter:

```
% cd Desk
```

Now, hit the **tab** key.

Most shells have a feature which will automatically complete the word you are typing when the **tab** key is pressed. In this case, the shell should have auto-completed the rest of Desktop/ . By default, this will only work if there is only one possible match. If there are more than one possible match, the shell will beep at you and wait for you to type in enough letters to distinguish the two (or more) files.

You can use tab completion to complete commands, directory names, filenames, and pretty much anything else you might enter into the command line that is sufficiently unambiguous.

A Note
It you enter some text into the shell and hit tab to autocomplete, but there are more than one possible entries that fit the text that you've entered, there is a way to have the shell return a list of possible completions instead of beep at you annoyingly. Ask my how to do this later (it involves modifying a configuration file. Don't worry, it's easy).

Viewing session history

Most shells will save a history of the most recent commands that you have entered. Use your shell for a little while, changing directories, listing files, redirecting output, etc. After you have entered several commands press the up arrow once. You should see the last command that you typed. You can continue to press up or down to cycle through your most recent commands, and when you find the a command you wish to re-enter, simply press enter.

As you cycle through the commands, you are able to modify any of the lines simply by using the left and right arrows.

Next, enter

```
% history
```

You will get a list of the most recent commands and what time you entered them. Now, you can re-enter any command by referencing the number in the history list

```
% !10
```

would re-enter the command that corresponds to the 10th entry in your history.

Summary

This document is meant to be a brief introduction to the Linux system for scientists. This information should help get you started working with Linux, but is by no mean exhaustive. There are a million books on Linux and UNIX operating systems providing any level of detail that you could ever want. But armed with this information, you are on your way to being able to use Linux to write useful scripts and execute useful programs.