Basic UNIX commands

Written by Dave Pawlowski, August 31, 2011

Introduction

In order to work with a UNIX based operating system, you have to become familiar with UNIX commands. There are many, and throughout this course you will become familiar with a lot of them. This document will get you started with some of the most basic ones. Next time, you'll get to know some more.

Every UNIX command has documentation called "man pages". You can access the man pages by typing:

man 'command name'

This will bring up a document in your terminal window that you can navigate through using the up and down arrows to scroll line by line, and the space bar to advance to the next page. Man pages tend to be a bit cryptic, in that they are meant to be brief and understood by people with a strong UNIX background. Understanding the details may be tricky, but you should be able to look at a man page and get an understanding of what a particular command is supposed to do, and also see some of the options for that command.

Pretty much every UNIX command has some options that you can specify when using the command. These options are referred to as flags and either are indicated by a single hyphen "-" followed by a letter (e.g. -r), or a double hyphen "--" followed by one or more words (e.g. --recursive). These flags may be similar or different for different commands. Often commands that do similar things have similar flags.

Commands

Files and Directories

Linux is all about manipulating files. The following commands will allow you to work with files (and directories); you will use these a lot.

But first, an introduction to this terminal thing we've talked about.

When you first login to a Linux system, the directory that you are accessing will be your home directory. The home directory has the same name as your user name, and it is where all of your personal files and subdirectories will reside. You will be able to create,

remove, or move files and directories in you home directory, but you will not be able to do so in any directories higher in the file hierarchy (more on this later). Lets start doing a few things in your home directory.

ls (list)

To find out what is in the current directory, you can use the ls command. This will list all the files and directories that are in your current working directory (which I will call cwd for short). If you are logged into chuck, you will see several files such as Desktop, Documents, Downloads, etc. These files are actually directories and may or may not contain other directories or files. These directories are specific to this computer, namely a Mac. If you were to login to a newly created account on a RedHat Linux machine, you would see very different things.

But that doesn't matter, what matters is that you be able to tell what is in a directory. The ls command as you used it only told you the contents, nothing about the contents. To see more, use:

```
% ls -l
```

The -l flag stands for "long". This tells you much more information about what is in a directory. We will talk about some of the information later, but a few notes on what you are looking at. (1) If there is a "d" in the first character of a line, that means the corresponding file is a directory. (2) You should see your username in the 3rd row. This tells you that you are the owner of the file, and therefore, you control the permissions (who can read, write, or execute) for the file. (3) The row before the date tells you the size of the file or directory in kilobytes. (4) The date is the timestamp that signifies the last time the file or directory was modified.

Another useful flag for ls is -a:

```
% ls -a
```

This will show any "hidden" files. A file may be hidden in Linux by having the first character of the filename start with ".". You will notice that there are 2 strange files '.' and '..' . The directory, '.' is Linux for "this directory", while '..' is Linux for the directory one level up in the hierarchy. So, for example you could execute:

```
% ls -a .
```

and you would see the contents of your cwd, exactly the same as ls -a. Or, you could do:

```
% ls -a ..
```

to see the contents of the directory one level up. Note that ls can take an argument, so I

could replace the ".." with another directory to see its contents. Also note that you can combine flags:

```
% ls -la Documents
```

would show you the content of the Documents directory, in long format, including hidden files.

There are many more flags that you can use with ls, and I'll leave it to you to explore them. These tend to be the most useful.

cd (change directory)

Now that you can see the contens of a directory, it is useful to be able to more around the file system. The command

```
% cd 'directory'
```

means change the cwd to "directory". For example, assuming you are in your home directory, you can change to Documents using

```
% cd Documents
```

Now you can do an ls to see the contents, if there are any. To get back to your home directory, which is one level up in the hierarchy, you do

```
% cd ..
```

You can confirm to yourself that you are back in your home directory by doing a ls. What happens if you run the command cd .?

One last note on cd, if you type the cd command with no arguments, you will change directories to your home directory.

mkdir (make directory)

You can make directories by simlpy using the command:

```
% mkdir unix
```

typing ls unix will show the contents of you newly created directory, which should of course be empty.

pwd (print working directory)

A Pathname tells you where you are in relation to the entire UNIX filesystem. Let's say you are in your home directory. Typing the command:

```
% pwd
```

will print the pathname of your current working directory. In my case, it is simply "/Users/dpawlows" (for me). We will talk more about the filesystem later, but you should know that each of your home directories are in the Users directory. You can change into this directory and see this for yourself:

```
% cd /Users
% ls
```

Doing this should show you a list of everyone's home directory. What happens if you try to make a directory here?

```
% mkdir temp
```

Since you are not the owner of this directory, and the permissions for the directory are set such that the owner is the only one that can write to it, you get an error. If you do a ls -la, you will see that the owner is someone named root. The root user, also know as the superuser, is the user that has permission to do anything they want on the system. root is an account on every UNIX system, and it is very dangerous. When you are logged on as root, you can delete, rename, modify, etc any file on the system, including those that are required for performing system tasks. For this reason, most people try to avoid logging on as root unless it's absolutely necessary. You will not have access to the root account on this machine for obvious reasons.

mv (move)

You can move a file from one directory to another or change it's filename using the mv command. This doesn't copy the file; i.e. you only end up with one file after the command has been executed. If you change directories to your home directory, and make a new directory called 'temp', you can change its name to MyFiles by typing:

```
% mv temp MyFiles
```

Alternatively, you could move the directory to another directory by typing:

```
% mv temp Documents/
```

or do both:

```
% mv temp Documents/MyFiles
```

cp (copy)

If you want to copy a file or directory, you use the cp command. For example, try this:

```
% cp /Users/dpawlows/Public/Phy379/assignment_1.txt .
```

Note the ".". cp requires at least 2 arguments, the source file and the target file. If you just specify a directory for the target file, as in this case, then the target file has the same name as the source file. This command should put a copy of the first homework assignment in your working directory.

rm (remove file)

To delete (remove) a file, use the rm command:

```
% cp assignment_1.txt temp
% ls
% rm temp
% ls
```

You can use rmdir to remove a directory (make sure its empty) or rm -r to recursively remove directories and all the files inside. Careful, this can be dangerous! It is common to include the -i flag when using rm, which stands for interactive; e.g. ask before actually deleting anything.

cat (concatenate)

For now, one last command: cat. Cat is actually a pretty useful and powerful command. Its primary use is to display the contents of a file to the screen. For example, in the directory that you copied the first homework assignment, you can type:

```
% cat assignment_1.txt
```

This will write out the contents of that file to the screen. Note that it doesn't know what to do with formating, or program specific characters., i.e. if you tried to use cat on a word document, you would get a big jumbled mess; cat is used to read normal, basic, text.

But cat can do more, you can also create a file using it:

```
% cat > newfile
```

When you use cat like this, the UNIX prompt doesn't come back!. UNIX is waiting for you to do something. Anything that you type will be placed into a file called newfile.

When you are done typing, press control-d to tell UNIX that you are done, and you will be back at the prompt. If you do an

```
% ls
```

you will see a file called newfile, and if you want to see the contents of the file, you can use

```
% cat newfile
```

The '>' symbol is what is known as a redirection symbol. UNIX reads that as: "take the output from this command and insert it into this other file". You can use '>' with any command that gives output. For example, try

```
% ps aux > tempfile
```

This will print a bunch of information on the processes that are currently running on your machine into the file tempfile. You can see this by typing

```
% cat tempfile
```

You'll notice that if you run the ps command above over and over again, you don't actually add anything to tempfile. When one '>' is used, UNIX creates a new file, and if one exists, it just overwrites it. However, if you use two of them '>>' then UNIX will append the file that you are trying to redirect to. For example:

```
% cat assignment_1.txt >> tempfile
```

will place the contents of assignment_1.txt at the end of the existing contents of tempfile. In this manner, cat can be used to do powerful things. Later on we will talk about writing scripts, which are small programs that are comprised of one or more lines of UNIX (or other) commands. Being able to manipulate files using cat can be extremely useful in this context.