2G03 Homework 1, 2020

Due: Friday, September 18

This homework is intended to help you learn more about unix, how it works and how to configure your account.

1 File Wildcard Expressions

Regular expressions are very important in unix. There is a small set built into the command shell that simplifies using commands that manipulate more than one file at once.
e.g. **ls *.out** Lists all file names that end in .out

Exercise: Work out the syntax for a single command to perform the following tasks.

1. List all files that start with b and end in 3.
   Note: You might use the long list of files somewhere like **/bin/** to test that wildcards work the way you think.
2. Copy all files that end in a . followed by any three characters (eg. 'mydoc.pdf') to a directory called backup.
3. Remove (delete) all files that start with a single uppercase letter.

2 Options for Unix Commands

Every unix command usually has many options that change its behaviour:
Command [**Options**] **...**
Options most commonly take the form of a hyphen followed by a letter:
e.g. **ls -l** List files in long format.

Many commands allow you to use a shorthand form for several options at once:
e.g. **ls -lt** List files in long format (-l) and ordered with most recent files first (-t).

This exercise will make you more familiar with some unix commands and their options. Use the manual pages e.g. **man** *command* to find out more about each command and its options.

When you run the manual command the manual page is viewed using the **less** text file viewing tool. Using less you can search the manual for *text* using */text* and hit **n** to go to the next match in the file. **Enter** moves forward one line, **SPACE** or **CTRL-F** goes forward one page and **CTRL-B** back one page. **q** quits.

Exercise: Find out what the options are to modify the following list of command to perform the requested function. Test the options in question on some files to see if they work.

1. **ls**: How do you make ls list files in a single column?

2. **grep**: How do you make grep ignore case (e.g. treat 'and' the same as 'AND')?

3. **grep**: How do you make grep search files in all subdirectories, not just the current directory?

4. **rm:** How do you make rm ask you to confirm before it deletes a file?

5. **tail:** How do you make tail show the last 30 lines of a file?

6. **sort:** How do you sort in reverse order?

3 Shells and Commands

Modern shell program such as **tcsh** include powerful features like command line editing and automatic completion of filenames.

With features such as command line completion it is good to know where the commands are coming from. Shell commands like **which** and **whereis** can be used to determine where the actual executable file for a command resides. For example, **which more** tells you that the **more** command is actually a program and the executable file is **/bin/more**. Not all commands are associated with an executable file. Some are built into the shell itself. For example, **which jobs** tells you that **jobs** is a shell built-in command.

Exercise: Answer the following about **tcsh**. You may need to experiment a bit to work out the behaviour.

1. **path**: Find the value of the **path** variable for your default shell on phys-ugrad.mcmaster.ca and list it here. You can google to find out how to do this and run the command oh phys-ugrad to get the answer.

2. **path**: What is the function of the **path** variable? (You may want to look at the man page for **tcsh** or google about path in unix)

3. If you type **xeyes <RETURN>** at the command prompt, which would run, the standard **xeyes** program or a program in your current directory also called **xeyes**? Most importantly: *Why?* (Note: It is a *really* bad idea to give a program the same name as an existing command.) Hint: The **.** character indicates the current directory – where you are now. There is a command to tell you which program would run without running it. You can make an runable xeyes in your directory by creating the file called **xeyes** (e.g. **touch xeyes**) and then making it executable (runable) with **chmod +x xeyes**

4. **TAB completion**: The TAB key attempts to complete the current word you are typing at the command line. Which files does the shell examine for completions to the first word on a command line?

5. **TAB completion**: Which files does the shell examine for completions to the second word on a command line?

4 Shell Start-up scripts

Many programs have a start-up script that the user can modify to affect the default behaviour of the program. This avoids cumbersome starting options. The most important script is the .cshrc file. This is run every time you start up a new shell in a terminal window. Common practice is to generate aliases for commands you like to use and to customize the behaviour of the shell.

Edit the start-up script for your account. Start an editor in your home directory to edit the .cshrc file. e.g. **gedit .cshrc &** or **xemacs .cshrc &** or whatever editor you prefer. You can test if your .cshrc file is correct by typing **source .cshrc**. Any aliases you have put in the file will become in effect at this point and may have to be removed with **unalias**.

Exercise: Regardless of what you chose to put in your own .cshrc file, list here the appropriate line you would need to put in the .cshrc file to have to the following work.
Hint: **man tcsh** or **man set** and locate information on the shell commands listed in **bold** below. You can also google of course.

1. **alias**: How would you arrange it so that when you type **rm** it will always ask for comfirmation before deleting each file?

2. **alias**: How would you arrange it so that when you type **xterm** you get an xterm window with a red cursor?

3. **set**: How do you prevent accidental overwriting of existing files with a redirection (i.e. **ls** > **junk** will not overwrite any existing file called junk) ?

4. **set prompt:** In a terminal window by default it gives you a prompt something like this:
   `[wadsley@phys-ugrad ~]$`
   How would you make a prompt that showed the time? e.g.
   `{10:40am}`

5. **set prompt:** Come up with your own prompt. State how to set the prompt to look that way and describe what it looks like.

5 Pipes and Redirection

Unix is a multiple user, multiple program environment. Pipes provide a mechanism to take output from one program and give it to another without intermediate steps such as a user file.

e.g.

man ls > junk Gets the man page on ls and puts it in a file called junk

grep ' -' < junk Looks in the file junk for all lines containing a space followed by a hyphen (i.e. options)

OR:

man ls | grep ' -' Does it all in one go with no file!

Exercise: You can pipe together as many commands as you like. The **ps aux** command lists all the processes on the machine. Work out a single line using 3 commands that will count the number of lines produced by ps that contain the word 'tcsh'. Try running the commands independently to make sure they do what you want before stringing them all together. You will need to discover a command that can count the lines in a file (look in the unix reference handout sheet).