Linux Commands
Optional extra tutorial

If you have never used the command line before, start by following the setup instructions on Camino

A: File handling: ls, cat

1) Launch the terminal and navigate to your folder for the class using cd. Refer to the setup instructions if needed.

2) List the files in your directory with the command:
`ls`

3) Using your text editor (Sublime or other), create a file named hello.txt, and add a few lines of text to the file. Save your work (In most text editors, you can use cmd-s or ctrl-s as a shortcut to save your file).

4) List the files in your directory again. What command will you use? You should now have a file named hello.txt.

5) List the information in long form.
`ls -l`

Look at the information listed; What do you think it means? Note that the -l is a parameter to ls that modifies its behavior. Many Linux commands have parameters like this.

6) Type `cat hel` and then hit TAB. This should autocomplete `hel` to `hello.txt` (assuming you have no other files in the same folder starting with `hel`). Tab-completion is a very useful feature in linux.

7) With an empty prompt, type the up-arrow key. This should allow you to cycle through your previous commands. This saves a TON of time.

B: More file handling: cp, rm, mv, and the man command

1) Make a copy of your hello.txt file called copy1.txt.
`cp animals.txt copy1.txt`

4) Make another copy of hello.txt called copy2.txt. What command will you use?
Verify that hello.txt, copy1.txt, and copy2.txt are all the same size. What command will you use?

5) Delete hello.txt
`rm animals.txt`

Remember, Linux has no undelete command.

9) Use the man command to learn more about the rm command.
`man rm`

Press g to top, G to end, q to stop, h for help

Note – you can use the man command to find out more about any other command. One of the things man will tell you is what parameters are available for that command.

C: Directories: pwd, cd, mkdir, rmdir

Directories (**AKA folders**) on your system are organized in a hierarchy, with folders nested in other folders. For instance, a typical login directory is a folder whose name is the same as the username – let's call it bronconame - that is inside of another folder called home. This would be represented at the command line as:

/home/bronconame

The slashes are used to represent that the thing to the right of the slash is inside the folder on the left. For example, if we are in /home/bronconame, that means we are in a folder called bronconame, which is inside a folder called home

1) Open the terminal and navigate to the folder you created at the beginning (Go back to the instructions in the setup document if necessary). Type the command:

`pwd`

This means "print working directory" — it will show you where you are in the directory tree.

2) You can move upward one level in the directory tree by typing:

`cd ..`

Check out which folder you're in. What command will you use?
You should now see:

/home (PC) or /Desktop (Mac)

3) If you ever get lost in the tree use the cd command to get back to the default directory.

`cd`

Verify which folder you're in again. What command will you use?
Now navigate back to the folder where you are storing your programs, using cd.

4) The mkdir command creates a subdirectory in your current working directory.

`mkdir foo`

Print out the contents of the current folder, with the additional information about each item. What command will you use?
This will show that you have created the subdirectory foo of your login directory. How can you tell that foo is a subdirectory and not just a file?

5) Enter the subdirectory by typing

`cd foo`

Verify you are in the foo folder. What command will you use? Show that there are no files in the subdirectory foo. What command will you use?

6) Using your text editor (Sublime or other) create a file in foo called animals.txt, with whatever contents you like.

7) Let's create another subdirectory of your login (or Desktop) directory.
First, navigate to the appropriate folder. What command will you use?
Then, create a new folder called bar. What command will you use?
Verify the folder has been created. What command will you use?

8) Use the rmdir command to *delete* the subdirectory bar by typing:

`rmdir bar`

Verify that bar is gone. What command will you use?

9) Can we delete the foo subdirectory the same way? Do you have any ideas why you get this behavior? Do whatever is required to delete the subdirectory foo.

10) Let's try a multi-step task. Create new directories foo and bar. Navigate to foo and create a file that contains your name, called name.txt

11) You can make more than one directory move at a time by specifying a path. For instance, to go directly from foo to bar, type:
`cd ../bar`

12) You can access name.txt from bar by typing: ../foo/name.txt Staying in bar, create a copy of name in bar called copy1.txt. What command will you use?

13) Navigate back to foo, and from foo, create a copy of name in bar called copy2.txt. What command will you use?

D: g++ with parameters

1) A useful parameter that can be used with the g++ command, which compiles your code, is the -o parameter. This allows you to give your executable file a name besides a.out or a.exe. Navigate to a folder that holds a C++ program. Type:
`g++ -o main.exe yourfile.cpp`
Then to run your program you will type ./main.exe. Note: to make this work on Mac you might need to give your executable a name ending in .out.

2) If you are using auto loops or other features that were added to the C++ standards in the 2011 update, you might get warnings about using these features. Your code will still compile, but the warnings can make it a little harder to understand your errors. If you'd like to get rid of those warnings, you can type
`g++ -std=c++11 yourfile.cpp`

3) Try to compile with both of these parameters `-o` and `-std=c++11` at the same time. What do you learn about the order of parameters to g++?