ATMS 305: Computing and Analysis
Module 1: Introduction to Unix & Shell Scripting
Due: September 5, 2017

Basic Unix Commands

To complete the Module 1 lab exercise, you will need to open a terminal window in cocalc.com using by selecting the `+Create` button or the `+New` tab, followed by the `>_Terminal` button. You may name your terminal by the date or by whatever convention you choose. [50 pts]

1. **Directory Navigation**
   For each of the following directives, make note of the terminal response or output. You will hand in your notes as the first assignment. [20 pts]

   (a) Determine your current working directory. What is it?

```
     ~$ pwd     \
end{verbatim}
\item List the contents of the current directory in long format. What additional informat
     \begin{verbatim}
     ~$ ls -l
```

   (b) List the contents of the current directory in long format with hidden files revealed. What additional files/directories are present in your home directory?

```
  ~$ ls -al
```

   (c) Create a direct

```
     ~$ mkdir test
```

   (d) Change the current working directory to the new *test* directory.

```
     ~$ cd test && ls -l
```

   What does the symbol && do?

   (e) Change the working directory back to your home directory.

```
     ~$ cd ..
```

   What do the two periods accomplish? To discover for yourself, you may wish to create a directory beneath test and move to this new directory. Use cd with the periods, and use cd alone. What is the difference?

   (f) Write a new function called cd to combine the operations cd and ls as follows. Be careful with your syntax.

```
function cd() {
    new_directory="$*";
    if [ $# -eq 0 ]; then
        new_directory=${HOME};
    fi;
    builtin cd "${new_directory}" && ls
}
```

Describe what happens each time the builtin command cd is used? If you are annoyed by this feature, you may enter the function again as it appears above but omitting the `&& ls` portion of the final line.

(g) Within the directory *test*, make a new file with the command:

```
~$ echo this\ is\ my\ first\ file > test.txt
```

Why do we use the backslash (also called an escape)? How can you verify your file has been created?

(h) Read your file with the following command.

```
~$ more test.txt
```

What happens?

(i) We can add a second line as follows:

```
~$ echo this\ is\ the\ second\ line >> test.txt
```

How does this command differ from your previous version?

(j) When using `more filename`, you can search for a particular string by selecting the forward slash key, typing the string, and pressing enter. However, if you want to look for a string in multiple files, then the command grep is a better option. Type the following:

```
~$ grep second test.txt
```

Notice the syntax: `$ grep string filename`. What is the output?

(k) Move to your top directory and remove your *test* directory with the following command.

```
~$ rmdir test
```

Has your file been removed? How can you verify that your directory has been removed?

(l) If your file has not been removed, try typing

```
~$ man rmdir
```

This is the builtin documentation. Why does this command not work?

(m) Try removing the *test* directory by using the following:

```
~$ rm -rf test
```

What is the primary difference between rm and rmdir? What do the options -rf mean?

(a) Display the current file permissions on the file monthly_in_situ_co2_mlo.csv. What are they? (**N.B.** You can type the beginning of a filename to the point of it being a unique filename and then use tab to autocomplete the filename.)

```
~$ ls -l monthly_in_situ_co2_mlo.csv
```

(b) Restrict access to the file questions by removing read permissions for group and other.

```
~$ chmod go-r monthly_in_situ_co2_mlo.csv
```

(c) Display the

```
~$ ls -l questions
```

```
\end{enumerate}

\item \textbf { Viewing Files}\\

For each of the following directives make note of the response or output of your terminal

\begin{enumerate}
\item List your files.
\begin{verbatim}
~$ ls
```

(d) View the file history using either less or more.

```
~$ more monthly_
```

Using more, you will see a prompt at the bottom of your screen which looks something like :
–More–(15

(e) Type the Return key a few of times to see one new line of text displayed on the screen each time. Note how the percentage in the prompt changes. Press space onType b to go back one full page in the file.
Type q to quit the pager and return to the system prompt.

(f) Now that

```
~$ history
```

2. **Manipulating Files**
Upon completin

(a) List the

```
~$ ls
```

Note that the file old is one of the files in your home directory.

(b) Copy ol
cpoldnew ls

(c) Rename the file old to ancient. List your files to confirm that ancient and new exist.

```
~$ mv old anc
~$ ls
```

(d) Remove the file ancient and then list your files to ensure it has been removed.

```
~$ rm ancient
~$ ls
```

(e) Create a directory called myfolder. List your files to ensure that myfolder was created.

```
~$ mkdir myfolder
~$ ls
```

(f) Set your current working directory to myfolder. Do not, however, type in the entire name of the directory. Instead type the first two letters and use Tab completion to fill in the rest of the file name.

```
~$ cd my<Tab> (Press the Tab key)
```

(g) Copy the file new from your home directory to myfolder.

```
~$ cp ~/new new
```

(h) Using a wildcard, list all of the files that begin with the letter n.

```
~$  ls n*
```

(i) Return to your home directory.

```
~$ cd
```

(j) Remove the directory called myfolder and all of its contents. List your files to ensure that it was removed.

```
~$ rm -r myfolder
~$ ls
```