**Homework 1 - Linux Refresh**

**Ellen Wilson 2/9/2021**

In this homework I will be asking you to review the purpose of the shell and how it works, common shell commands and demonstrate scripting skills.

You will need this file [Hutia\_DNA.fasta](https://www.dropbox.com/s/h52y375q8sxcpj0/Hutia_DNA.fasta?dl=0)

**General concepts to refresh and lessons to review if you need them:**

[Shell Commands](http://swcarpentry.github.io/shell-novice/01-intro/index.html)

[Working with Files and Directories](http://swcarpentry.github.io/shell-novice/03-create/index.html)

[Pipes and Filters](http://swcarpentry.github.io/shell-novice/04-pipefilter/index.html)

[Loops](http://swcarpentry.github.io/shell-novice/05-loop/index.html)

[Shell Scripts](http://swcarpentry.github.io/shell-novice/06-script/index.html)

[Documentation](https://blog.submain.com/code-documentation-the-complete-beginners-guide/) -specifically in-line-comments

**Remember these Common commands as you work through this exercise**

ls mkdir cd pwd

cp mv rm rmdir

cat less head tail

grep wc uniq top

man sort wc \*

| . .. /

**Part I: Questions**

1. What is a Graphical User Interface (GUI) and how does that differ from the Command Line Interface (CLI)? The GUI is the most common interface for computers today. It is ran with clicks and movement of the mouse to accomplish tasks. It is usually easier to learn However, it is not the most useful when trying to accomplish very large tasks with huge files. The CLI is less intuitive and harder to learn, but can assist with large, repetitive tasks very quickly.
2. What does the shell do? It is a program that runs other programs, and most computers today come equipped with it. Also called the Terminal.
3. What is the command used for listing things in a directory? ls
4. What is the command used for changing directories? cd
5. What command would you use to get your current working directory? pwd
6. How do you get the manual for these commands? man
7. What does the shell prompt look like? Typically, it is a symbol such as $ or %, prompting us to enter in the next information. This is my prompt: Ellens-MacBook-Pro:~ ellenwilson$
8. How would you list things in a directory in chronological order? ls -lt
9. Name two ways to get to the ‘home’ or ‘root’ directory. cd ~; cd ..

## cd .. would only work if you are just down from the root directory.

1. What is the difference between an absolute path and a relative path? The absolute path is the exact, complete list to a place in the directory. The always contain the “root” element. The relative path is just relative to where you are, and can change depending on where you are in the directory. Using “..” is an example.
2. What are the two relative path directories we talked about and what do they mean? The two relative paths are . and .. The singular . means my current directory, and the double .. means up a directory relative to the path.
3. Why shouldn’t you put spaces in filenames? This is because the shell will read it as two separate items. For example, if you put greentree.txt, that will read it as one file. But if you put green tree.txt, it will look for green and tree.txt and try to complete the command on both of those things.

#because programs read arguments fed to them which are separated by spaces. The program will read the string after the space as another argument.

1. Name a way to have multiple readable words in a filename without spaces? You can use ., \_, -, or capital letters to separate words.
2. What is the program nano, what does it do? Do you use nano? If not, what do you use? Nano is a text editor program that is easy to access in the shell. I do not use Nano, I use BBEdit because it is slightly more user friendly for beginners.
3. \* is a form of a wildcard. What does it mean? The star wildcard can be used to search easier in large directories. It can find zero characters, single characters or strings. It can also be preceded by commands such as ls or rm.

#wildcard that matches any character

1. The following code would match what? ls \*.txt List all of the files in the directory that have the .txt extension/ are files with .txt.
2. Why do we need to be careful with the mv command? Mv is used to move files from one directory to another, or to rename files. But mv can also overwrite source files if done incorrectly, and can completely delete the contents of the file.
3. Why do we need to be careful with the rm command? You can easily remove files or directories and important information from your shell. You cannot restore those deleted files- there is no recovery. However, you can set up a double check system that askes you if you are sure you want to remove something.
4. What is the difference between > and >>? > will overwrite a file with new information given, but >> will append the files and add on to existing content.
5. What does head do? According to man head, head displays the first lines or first portion of a file. Tail does the opposite and shows us the end of a file.
6. What is the purpose of | (pipe)? A pipe lets you work on a command, and the output of that command becomes the input for the next command in the pipeline. It can redirect the process of one command to another for further processing.
7. Interpret the following command:

cat huge\_file.fasta | uniq | head -n 5 >proteins.fasta

This would take the first 5 lines of a file and put them into another file.

# take the first 5 unique lines

1. What is a loop and when would you use it?

A loop is a tool used to conduct a series of commands on each individual file, line, etc, repeatedly and quickly. This saves time when needing to analyze large files that need the same command applied to them.

1. In a directory with the following files. What would the following loop do? apple.txt, banana.txt, orange.txt $ for filename in \*.txt; do > cat ${filename} >>fruits.txt > done

This command would read each file separately in a loop, and then put that output into a file named fruits.txt. It would condense the information from three files into one.

1. What is a shell script? A shell script is a command line interpreter. We can use a shell script to execute repetitive tasks instead of typing out the command individually, and this can save a lot of time. However, it may be easy to make mistakes, and this can hurt your program.
2. What are the benefits of writing and running a script over typing the code in? It could be that it is more user friendly to write in a script, especially if a mistake is made, then it can be easily fixed. It is also better for making notes with the # sign. The script can also be saved and reran again, so we only need to type it out once.
3. When looking at a script what does # mean? Why would you use one? # in the script means information that is not part of the code and will not be ran. It is often used to make notes about the script so that you can understand why you worked on that piece of code or how it will be ran. It is also very useful to go back and look at your old notes.
4. What does grep stand for and what does it do? Grep stands for “global regular expression and print”. It searches files or other input that match a regular expression that we have given it. It then prints the input to the program’s output, and can be used to complete other commands. According to man grep, it finds the input files and selects lines that match the patterns. There are other options to use with grep as well.
5. What does find do? According to man find, find is walk a file hierarchy. It can be used to find files, and specific things about those files.

**Part II - Writing Code**

As part of this exercise please open the shell and practice moving around into different files. Best practice is to try a little bit every day. Try to challenge yourself by not using the mouse. For each of the questions below type your code and the result from the shell prompt.

1. Open the shell and change directories to the desktop and list the contents, paste your code here.

cd Desktop/

ls

1. Draw the file structure from the root directory into one of the folders on the desktop. You can submit a pdf with a drawing. Select three locations and type the commands for changing directories to those locations use a combination of absolute and relative paths.

Location: planets. Absolute path: Ellens-MacBook-Pro/Desktop/planets/mars.txt

cd Desktop/planets

Location: Desktop. Absolute path: Ellens-MacBook-Pro/Desktop

cd ../ (from planets directory)

cd ../Desktop (from Documents directory)

cd Desktop (from home directory)

Location: data. Absolute path: Ellens-MacBook-Pro/BIOL792/data/

cd ../../BIOL792/data (from planets directory)

cd BIOL792/data (from home directory)

1. On your computer create a directory on your desktop. Create two files without opening them that have file extensions .txt. Then use wildcards to list the files in that directory. Paste you code here.

cd Desktop/

mkdir Homework1

cd Homework1/

touch hw1.txt

touch hw2.txt

ls \*.txt

1. Download the file Hutia\_DNA.fasta from the Github repo. Make a new directory on the desktop. Move the file into that directory. List the contents of the directory. How many lines are in this file? Fasta files are a type of data file that holds DNA sequences. They are all formatted the same. The first line starts with a > and a name followed by a line break, and then DNA sequence followed by a line break. The next line starts with an > and another name followed by a line break and the DNA sequence. With that in mind, how could you use linux commands to determine how many sequences are in this file? How many sequences are in this file?

cd Desktop/

mkdir Hutia

mv Hutia\_DNA.fasta Hutia/

cd Hutia

ls

wc -l Hutia\_DNA.fasta (There are 2382004 lines in this file)

grep -c “>” Hutia\_DNA.fasta (There are 1191002 sequences in this file)

#excellent!

1. Build a single line of code that would take the first 100 sequences of this file and put them in a new file called ‘MyOutputFile.fasta’. Use a combination of wc, sort and uniq in a single line of code to tell me how many unique lines are in this file. Paste your commands and the answer here.

touch MyOutputFile.fasta

head -100 Hutia\_DNA.fasta > MyOutputFile.fasta

sort MyOutputFile.fasta | uniq | wc -l (There are 100 unique lines)

#great!

1. Write code that would create a loop to copy all the files in one directory ending in .txt to another directory. Paste the code here.

(In directory Hutia)

mkdir Hutia2

for f in \*.fasta; do echo copying ${f}; cp ${f} Hutia2/; done;

#using the echo commands means it would just print your command to the screen nad not run it. This looks correct just without ‘echo’

1. Write a bash script with in-line documentation (hint #) to show how to run a made up python program (script) called 'find\_taxa.py' on set of files ending in ‘.fasta’. Show in the script how to run the program on each of those files. What is the name of your script? Type the script here.

#!/usr/bin/python3

#Find Taxa- script name.

#Find\_taxa.py will run on files ending in “.fasta”.

#Each file will be ran using a for loop. This will cut back on time.

#!/usr/local/bin/sh

#for file in \*.fasta;

#do

#python3 find\_taxa.py $file

#done

1. From the Hutia\_DNA.fasta file tell me how on how many lines do we find the pattern ‘GAGA’. What was the code used to find this?

304806 lines have the pattern “GAGA”.

grep -c “GAGA” Hutia\_DNA.fasta (from Hutia directory)