**BIOL 792 (Special Problems)**

**Dr Julie Allen**

**Homework 1 – Linux Refresh**

**Pankaj Bhatta**

**#Part I: Questions**

Q.1. What is a Graphical User Interface (GUI) and how does that differ from the Command Line Interface (CLI)?

Ans: GUI provides you with menus (windows, scrollbar, taskbar, buttons etc.) and is controlled via mouse or touchscreen, but CLI requires you to type commands to do your task.

Q.2. What does the shell do?

Ans: It is a computer program that provides a commands line interface which allows you to control your computer using commands entered with a keyboard.

# also to interact with other computers

Q.3. What is the command used for listing things in a directory?

Ans: ls #ls = list

Q.4. What is the command used for changing directories?

Ans: cd, cd.., cd~, cd/, cd- #cd = change directory

Q.5.What command would you use to get your current working directory?

Ans: pwd #pwd = print working directory

Q.6. How do you get the manual for these commands?

Ans: man #man = manual; views system's reference manuals

Q.7. What does the shell prompt look like?

Ans:

Text

Description automatically generated

Q.8. How would you list things in a directory in chronological order?

Ans: ls -t #t = Sort by time modified (most recently modified first) before sorting the operands by lexicographical order.

Q.9. Name two ways to get to the ‘home’ or ‘root’ directory.

Ans: cd or cd~ (for home directory); cd/ (for root directory)

Q.10. What is the difference between an absolute path and a relative path?

Ans: Absolute path starts from the directory root (/) and goes up to the actual object (file or directory) whereas relative path starts from the current directory (pwd) and goes up to the actual object.

# relative paths are relative to where you are this directory or one up

Q.11. What are the two relative path directories we talked about and what do they mean?

Ans: Single dot (.) and double dots (..)

The single dot refers to the directory itself and the double dots refers to its parent directory or the directory that contains it.

By default, these dots are hidden and do not show in the output of the command ls. To view these dots, we have to use the option a with the command ls i.e., ls-a.

# The single dot refers to the directory you are in, so the path of the single dot changes as you move into different directories

Q.12. Why shouldn’t you put spaces in filenames?

Ans: Because most programming languages use spaces to signify the end of a character string, and many software applications don't recognize file names that contain spaces.

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

Q.13. Name a way to have multiple readable words in a filename without spaces?

Ans: Underscore (\_), hypen (-), uppercases, numbers etc.

Q.14. What is the program nano, what does it do? Do you use nano? If not, what do you use?

Ans; nano is a text editor for Unix-like computing systems or operating environments using a command line interface.

No, I do not use nano, I use BBEdit text editor.

Q.15. \* is a form of a wildcard. What does it mean?

Ans: Wildcard (asterisk or \*) matches any character zero or more times. A wildcard in Linux is a symbol or a set of symbols that stands in for other characters.

It can be used to substitute for any other character or characters in a string.

For example, "prog\*"" matches anything beginning with "prog" which means "prog," "programs," and "programming" are all matched.

# fantastic!

Q.16. The following code would match what? ls \*.txt

Ans: It would match all the files having different or same characters ending with .txt extension.

Q.17. Why do we need to be careful with the mv command?

Ans: We are using mv to rename or move file so we should be careful of where are we moving it to or renaming it to.

#yes because mv can rename a file you want to be careful when moving it that you don’t also accidently rename it in the process.

Q.18. Why do we need to be careful with the rm command?

Ans: rm is used to remove files and folder so we need to be very careful because this command can wipe out your files and folders completely.

Q.19. What is the difference between > and >>?

Ans: ">" operator redirects the output to a file, if the file already exist and contains some data in it, the ">" would cause the data over written with new data.

">>" operator redirects the output to be appended to the end of the file instead of over writing it.

#> will always open a new file if that file exists it will overwrite it

Q.20. What does head do?

Ans: The head command is a command-line that displays the first part of a file given to it via standard input.

Q.21. What is the purpose of | (pipe)?

Ans: Pipes help you mash-up two or more commands at the same time and run them consecutively.

For example, when you use 'cat' command to view a file which spans multiple pages, the prompt quickly jumps to the last page of the file, and you do not see the content in the middle. You can use pipe the output of the 'cat' command to 'less' which will show you only one scroll length of content at a time i.e., cat filename | less.

Q.22. Interpret the following command:

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

Ans: Redirects and overwrites the first 5 lines from a unique file to already existed file named proteins.fasta. In case if the proteins.fasta file doesn't exist it creates one.

# in fact it will always create a file with the > command

Q.23. What is a loop and when would you use it?

Ans: A loop is used to repeat a specific block of code a known number of times. We would use it when we need to search for information or when we need to produce output.

For example, if we want to know the goal scores of all the major footballers in the world, we would loop from goal 1 and to the highest goal, and it would list out the names

of the footballers with their respective goals.

#nice!

Q.24. 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

Ans: It will append fruits.txt file to the folder named 'filename.'

#it will append the contents of each of the files banana.txt, orange.txt and apple.txt to a file called fruits.txt

Q.25. What is a shell script?

Ans: Shell script is simply a file containing a series of command executable by a Unix-shell, a command-line interpreter.

Q.26. What are the benefits of writing and running a script over typing the code in?

Ans: The biggest advantage of writing and running a shell script is that the commands and syntax are exactly the same as those directly entered at the command-line.

We don't have to switch to a totally different syntax, as we would if the script were written in a different language.

Thera are many advantages like easy program or file selection, quick start, interactive debugging etc.

#also so you can run a series of steps over and over and don’t have to remember the code

Q.27. When looking at a script what does # mean? Why would you use one?

Ans: Hashtag (#) is an excellent way of keeping track of your shell script. Using #, you can include notes or comments of what you did, the function of your commands, and the progress of your coding.

# hashtag is for twitter it is called the pound or hash. Nice job

Q. 28. What does grep stand for and what does it do?

Ans: grep stands for global regular expression print. The grep utility searches any given input files, selecting lines that match one or more patterns.

Q.29. What does find do?

Ans: It searches for files and directories in a directory hierarchy based on a user given expression and can perform user-specified action on each matched file.

The general syntax for the find command is- find [options] [path...] [expression]

**Part II - Writing Code**

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

**Ans:** Pankajs-MacBook-Pro:~ pankajbhatta$ cd desktop

Pankajs-MacBook-Pro:desktop pankajbhatta$ ls

Q.2. 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.

**Ans:** cd /users/pankajbhatta/desktop/BIOL792\_2 #absolute path to get to BIOL792\_2 to root directory

cd .. #to get to desktop desktop directory

cd ../../ # to get to users directory

cd pankajbhatta #relative path from /users to pankajbhatta

cd ~ #to home directory /users/pankajbhatta

cd ~/desktop #relative path from home directory to desktop

cd BIOL792\_2 #relative path from desktop to BIOL792\_2

Diagram

Description automatically generated

Q.3. 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.

**Ans:** Pankajs-MacBook-Pro:desktop pankajbhatta$ mkdir PB

Pankajs-MacBook-Pro:desktop pankajbhatta$ cd PB

Pankajs-MacBook-Pro:PB pankajbhatta$ touch PB1.txt PB2.txt

Pankajs-MacBook-Pro:PB pankajbhatta$ ls \*.txt

PB1.txt PB2.txt

Q.4. 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?

**Ans:** Pankajs-MacBook-Pro:desktop pankajbhatta$ mkdir new

Pankajs-MacBook-Pro:desktop pankajbhatta$ mv Hutia\_DNA.fasta new

Pankajs-MacBook-Pro:desktop pankajbhatta$ cd new

Pankajs-MacBook-Pro:new pankajbhatta$ ls

Hutia\_DNA.fasta

Pankajs-MacBook-Pro:new pankajbhatta$ wc -l Hutia\_DNA.fasta

2382004 Hutia\_DNA.fasta #2382004 number of lines in Hutia\_DNA.fasta file

Pankajs-MacBook-Pro:new pankajbhatta$ grep -c "^>" Hutia\_DNA.fasta

1191002 # number of sequences in Hutia\_DNA.fasta file

# excellent! You just need grep -c “>” Hutia\_DNA.fasta

Q.5. 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.

**Ans:** Pankajs-MacBook-Pro:new pankajbhatta$ head -n 100 Hutia\_DNA.fasta > MyOutputFile.fasta

Pankajs-MacBook-Pro:new pankajbhatta$ wc -l MyOutputFile.fasta | sort |uniq -c

1 100 MyOutputFile.fasta # there are 100 unique files without any duplicates

# here you made a file of 100 lines then asked how many lines were in that file. You want the output of uniq piped to wc

Q.6. 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.

Ans: Pankajs-MacBook-Pro:desktop pankajbhatta$ for file in desktop;

> do

> cp \*.txt new

> done

# all the file having .txt extensions are copied from desktop directory to new directory.

### almost

# for file in \*.txt;

# do

# cp $file new

# done

Q.7. 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.

**Ans:** #!/usr/local/bin/python3

find\_taxa = open ('\*.fasta', 'r')

for file in find\_taxa:

do

echo $file\_taxa

done

# Name of my script is 'find\_taxa.py'

# To run the program, simply write python 3 find\_taxa.py

### I was just looking for a bash script not a python script

#!/usr/local/bin/sh

#for file in \*.fasta;

#do

#python3 find\_taxa.py $file

#done

Q.8. 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?

**Ans:** grep -e 'GAGA' Hutia\_DNA.fasta

#grep -c ‘GAGA’ Hutia\_DNA.fasta

#how many lines were in the file?

## nice job Pankaj!

**Thank You! ☺**