JESHURUN BINEY BIOL_792

Homework 1 - Linux Refresh

Part I: Questions

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

GUI lets a user interact with the device/system with the help of graphical elements, like windows, menus, icons, etc. The CLI, on the other hand, lets a user interact with their device/system with the help of various commands.

2. What does the shell do?

The shell is a program where users can type commands. With the shell, it's possible to invoke complicated programs like climate modeling software or simple commands that create an empty directory with only one line of code.

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?

```
<ls - - help> or <man ls>
```

7. What does the shell prompt look like?

```
$
```

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

```
ls -t
```

9. Name two ways to get to the 'home' or 'root' directory.

```
<cd> or <cd ~>
```

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

An absolute path specifies a location from the root of the file system.
A relative path specifies a location starting from the current location.

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

. shows current directory
.. moves up a directory

12. Why shouldn't you put spaces in filenames?

Spaces can present a problem when dealing on the command line. A space in a filename can cause errors when loading a file or when transferring files between computers. And the CLI might read the file name as a command instead.

13. Name a way to have multiple readable words in a filename without spaces?
Common replacements for spaces in a filenames are dashes (-) or underscores (_). Alternatively you can also use a technique called camelCase which uses letter case to separate name elements.

14. What is the program nano, what does it do? Do you use nano? If not what do you use?
nano is an easy to use command line text editor for Unix and Linux operating systems. It includes all the basic functionality you'd expect from a regular text editor, like syntax highlighting, multiple buffers, search and replace with regular expression support, spellchecking, UTF-8 encoding, and more. I use nano for .txt files.

15. ``*`` is a form of a wildcard. What does it mean?
* is a wildcard, which matches zero or more characters. When the shell sees a wildcard, it expands the wildcard to create a list of matching filenames before running the command that was asked for. [A wildcard in Linux (shell) means it might be a symbol or set of symbols representing other characters. It is generally used in substituting any string or a character.]

16. The following code would match what? ```ls *.txt```
It would list all the .txt files since * can be assigned to zero or more characters.

17. Why do we need to be careful with the ```mv``` command?
One must be careful when specifying the target file name, since mv will silently overwrite any existing file with the same name, which could lead to data loss.

18. Why do we need to be careful with the ```rm``` command?
rm should be used with great caution given that there is no way to retrieve files deleted using the shell. There is no trash bin to retrieve deleted files or folders.

19. What is the difference between ```>``` and ```>>```?
> redirects a command's output to a file (overwriting any existing content).
>> appends a command's output to a file.

20. What does ```head``` do?
The head command is a command-line utility for outputting the first part of files given to it via standard input. It writes results to standard output. By default head returns the first ten lines of each file that it is given.

21. What is the purpose of ```|``` (pipe)?
Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes

22. Interpret the following command:
```cat huge_file.fasta | uniq | head -n 5 >proteins.fasta```
The file “huge_file”’s contents are accessed using the 'cat' command. ‘fasta’ publishes them to standard output consecutively. The 'uniq' function is then used to eliminate duplicate lines from the file, and the 'head' command is used to display the first five (5) lines of the result. Finally, the output (the initial five lines) are directed into a file called proteins.fasta.

23. What is a loop and when would you use it?
A loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Loops allow you to repeat a process over and over without having to write the same (potentially long) instructions each time you want your program to perform a task.

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
```

Display the contents of every file in the current directory that ends in.txt, then add the contents of each file to a file called fruits.txt. In essence, the loop merges all of the txt files in the current directory into a single file called fruits.txt.

25. What is a shell script?
A shell script is a computer program designed to be run by a Unix shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages.

26. What are the benefits of writing and running a script over typing the code in?
It is faster to write scripts as they are usually written to automate a specific task within the main program/software. Coding in a Programming language is time taking as it involves designing a complete software. Scripts are written within a parent Program.

27. When looking at a script what does ```#``` mean? Why would you use one?
are using in writing comments and it asks the computer to ignore the preceding lines after the # sign.

28. What does ```grep``` stand for and what does it do?
Global regular expression print. The grep command searches through the file, looking for matches to the pattern specified. To use it type grep, then the pattern we're searching for and finally the name of the file (or files) we're searching in.

29. What does ```find``` do?
The find command will begin looking in the starting directory you specify and proceed to search through all accessible subdirectories.

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.

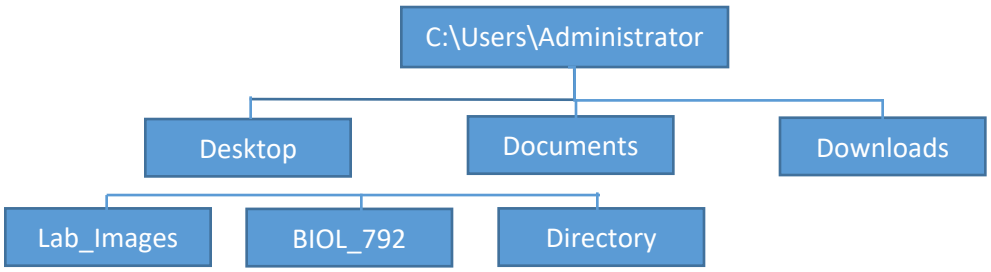
1. Open the shell and change directories to the desktop and list the contents, paste your code here.

```
Administrator@DML-MCIG9J654RU MINGW64 ~/Desktop/shell-lesson-data
$ cd

Administrator@DML-MCIG9J654RU MINGW64 ~
$ cd desktop/

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ ls
 BIOL_792/
```

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.



i. From main to desktop: $cd desktop
ii. From desktop to BIOL 792: $cd BIOL_792
iii. From BIOL_792 to documents: $cd ../ documents

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.

```
Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ pwd
/c/Users/Administrator/desktop

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ mkdir directory

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ cd directory

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ touch file_1.txt file_2.txt

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ ls *.txt
file_1.txt  file_2.txt
```

4. Download the file [Hutia_DNA.fasta](https://www.dropbox.com/s/07y610c8zt63nt5/Hutia_DNA.fasta?dl=0) 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?

```
Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ cp ~/Downloads/Hutia_DNA.fasta ~Desktop\directory

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ cp ../Downloads/Hutia_DNA.fasta ../Desktop/directory

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop
$ cd directory

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ ls
Hutia_DNA.fasta

Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ grep -c ">" Hutia_DNA.fasta
1191002
```

There are 1191002 sequences in this file

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.

```
Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ head -200 Hutia_DNA.fasta | sort | uniq | wc -l > MyOutputFile.fasta && wc -l < MyOutputFile.fasta
1
```

There is 1 unique line

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.

```
Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ for filename in *.txt do cp${filename} ~Desktop/directory done
> |
```

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.

runpy.sh

```
#!/bin/bash
fasta_files=(*.fasta)                  #Define the set of all the .fasta files in the current directory
for file in "${fasta_files[@]}"        #use a for loop through each file in the set of .fasta files
do
python find_taxa.py $file              #Run the 'find_taxa.py' program on each file
done                                   #end of the loop
echo "Program successful."             #Print message to confirm program completion
```

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?

```
Administrator@DML-MCIG9J654RU MINGW64 ~/desktop/directory
$ grep -c 'GAGA' Hutia_DNA.fasta
304806
```

304806 lines of the pattern ‘GAGA’