### Exercises

1. Working with ls in the home directory
   1. List all the files in /usr/bin

cd /usr/bin

ls

* 1. Use the man command with ls

man ls

* 1. List the size and permissions of **all** the files in /usr/bin

ls -s -la

* 1. List the files that start with ‘a’ in /usr/bin

ls a\*

* 1. What are the . and .. files/directories?

. means current directory which is /usr/bin

.. means the parent directory which is /usr

* 1. List all the file with extension .py in /usr/bin

ls \*.py

1. Creating and editing with vi
   1. Use vi to open a file called file1.txt for editing

vi file1.txt

* 1. Write This is some text in the file

i This is some text

* 1. Save and close the file with Esc :wq Enter/Return

esc :wq

1. Copying and removing files
   1. Use cp to copy file1.txt to file1\_copy.txt

cp file1.txt file1\_copy.txt

* 1. Use rm to remove file1.txt

rm file1.txt

* 1. Use ls to see the contents of the current directory

ls (only file1\_copy.txt remains)

1. Using cat and redirecting output
   1. Use cat to display the contents of file1\_copy.txt

cat file1\_copy.txt

This is some text

* 1. Redirect the output of the cat command to another file called file1\_contents.txt using >

cat file1\_copy.txt > file1\_contents.txt

* 1. Use cat to display the contents of file1\_contents.txt

cat file1\_contents.txt

This is some text

* 1. Redirect the contents of file1\_copy.txt once again into file1\_contents.txt but use >> instead

cat file1\_copy.txt >> file1\_contents.txt

* 1. Use cat to display the contents of file1\_contents.txt

cat file1\_contents.txt

This is some text

This is some text

* 1. Repeat parts *2.* and *3.* How is the output different?

cat file1\_copy.txt > file1\_contents.txt

cat file1\_contents.txt

This is some text (only one line, the other two got replaced)

1. Looking at what’s in a file
   1. Navigate to ~/class/ex1

cd class/ex1

* 1. Use head and tail to see the first and last 10 lines of the file, respectively

head ex1.bed

tail ex1.bed

* 1. Use the head command to see the top 50 lines of the file

head -50 ex1.bed

* 1. Use the tail command to see the last 25 lines of the file

tail -25 ex1.bed

* 1. Use more on ex1.bed

more ex1.bed

display one screen of data, can only explore forward

* 1. Use less on ex1.bed

less ex1.bed

can explore both forward and backward

1. Making and removing directories
   1. Create a directory called myDir using mkdir

mkdir myDir

* 1. Remove the directory using rmdir

rmdir myDir

* 1. Repeat step *1.*

Mkdir myDir

* 1. Copy file1\_contents.txt in myDir

cp file1\_contents.txt myDir/file1\_contents.txt

* 1. Try to remove the directory using rmdir

rmdir myDir

rmdir: failed to remove ‘myDir’: Directory not empty

* 1. Remove the directory using rm

rm myDir

rm: failed to remove ‘myDir’: Is a directory

* 1. Create the following directory structure using a single mkdir command. Do not use semicolon ; , ampersand & or other command connectors for this purpose  
     dir1  
     |---- dir2

*hint: dir2 is inside dir1*mkdir -p dir1/dir2

1. Making the prompt prettier
   1. Use echo to see the current PS1 environmental variable; Look up what the PS1 variable does

echo $PS1

* 1. Use ls to display all the hidden files in your home directory

cd

ls -a

* 1. Use emacs/vi to open your .bash\_profile

vi .bash\_profile

* 1. Add the following line to it:  
     export PS1="\[\033[38;5;10m\]\u\[$(tput sgr0)\]@\[$(tput sgr0)\]\[\033[38;5;13m\]\h\[$(tput sgr0)\]:\[$(tput sgr0)\]\[\033[38;5;14m\]\w\[$(tput sgr0)\]\\$ \[$(tput sgr0)\]"
  2. Make sure you spaces & new lines don’t get interchanged. There is no new line in the above command.
  3. Save the file and exit. Start a new terminal. Colors!

Becomes green purple and blue

1. Counting characters
   1. Navigate to ~/class/ex1

cd class/ex1

* 1. Count the number of characters in ex1.bed using wc

wc -m ex1.bed

78290

* 1. Count the number of lines in ex1.bed using wc

wc -l ex1.bed

3414

1. Redirecting different streams
   1. Run this command:  
      perl -e 'foreach(1..100){print $\_."\n"; print STDERR ($\_ / 2)."\n"}'
   2. Run the command and redirect only the standard out to myOut.txt

perl -e 'foreach(1..100){print $\_."\n"; print STDERR ($\_ / 2)."\n"}' > myOut.txt

* 1. Run the command and redirect the standard error to myErr.txt

perl -e 'foreach(1..100){print $\_."\n"; print STDERR ($\_ / 2)."\n"}' 2> myErr.txt

* 1. Run the command and redirect the standard output to myOut.txt and standard error to myErr.txt

perl -e 'foreach(1..100){print $\_."\n"; print STDERR ($\_ / 2)."\n"}' > myOut.txt 2> myErr.txt

* 1. Run the command and redirect both the standard output and standard error to mySeq.txt in a single command

perl -e 'foreach(1..100){print $\_."\n"; print STDERR ($\_ / 2)."\n"}' >mySeq.txt 2>&1

1. Piping data
   1. Run this command:  
      seq 0 .5 100 > longSeq.txt
   2. Using the pipe with head and tail, get the 50th line of longSeq.txt

head -n 50 longSeq.txt | tail -n 1

24.5

* 1. Print everything but the top 13 lines

tail -n+14 longSeq.txt

Starts from 6.5

* 1. Print everything but the last 13 lines

head -n -13 longSeq.txt

* 1. Count the number of characters in lines 45-50 (inclusive)

head -n 45 longSeq.txt | tail -n 6 > newSeq.txt

wc -m newSeq.txt

30