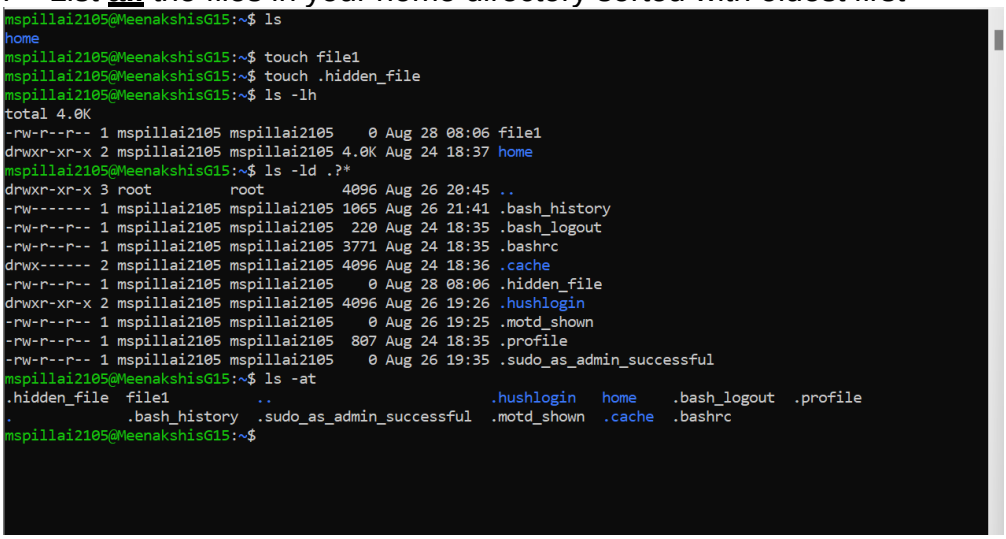
Programming for Bioinformatics | BIOL 7200

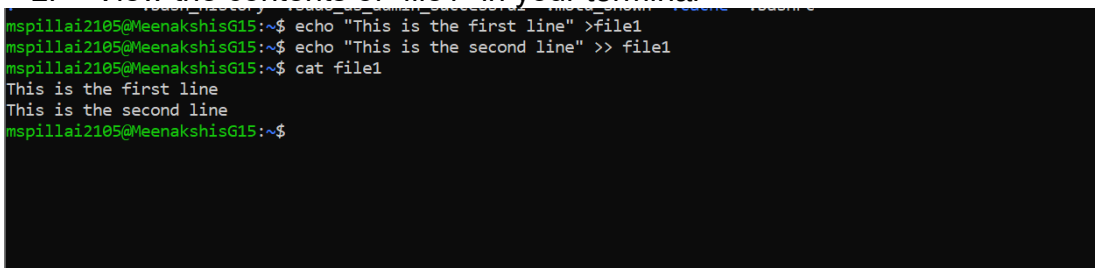
Submitted by: Meenakshi Somadasan (gtID:mpillai32)

Week 1 Exercise

1. Using documentation to explore functionality of ls
    1. List the files in your home directory
    2. Create two empty files in your home directory. One named “file1” and one named “.hidden_file” (note the dot in the second name)
    3. What is the size of file1? Show your working
    4. What is the size of the “.hidden_file”?
    5. List **all** the files in your home directory sorted with oldest first



2. Creating and viewing file contents **<u>using the terminal</u>**
    1. Add two lines of text to “file1”
    2. View the contents of “file1” in your terminal



3. Copying and removing files
    1. Use cp to copy “file1” to “file1_copy.txt”

2. Has the addition of ".txt" to the file name changed how the file contents look? Are file extensions significant in Unix systems?

   Ans. No the contents do not change as the file extensions valid on Windows are rarely valid on Unix systems
3. Use rm to remove "file1"
4. Create an empty file named "file2"
5. Run the command cp -n file1_copy.txt file2. Does "file2" now contain the same contents as "file1_copy.txt"? Explain

   Ans. The file "file2" is empty as the -n option has to be followed by an integer to make the command work

```
mspillai2105@MeenakshisG15:~$ cp file1 file1_copy.txt
mspillai2105@MeenakshisG15:~$ cat file1_copy.txt
This is the first line
This is the second line
mspillai2105@MeenakshisG15:~$ rm file2
rm: cannot remove 'file2': No such file or directory
mspillai2105@MeenakshisG15:~$ touch file2
mspillai2105@MeenakshisG15:~$ cp -n file1_copy.txt file2
mspillai2105@MeenakshisG15:~$ cat file2
mspillai2105@MeenakshisG15:~$
```

4. Using documentation to explore useful commands. State the command and options you could use to perform the following tasks:
   1. Create a directory structure "./a/b/c" in a single command (i.e., create a directory and any missing parent directories)

      Ans. mkdir
   2. Check if a file has Windows or Unix line endings

      Ans. cat -e <filename>
   3. Copy files but only replace existing files if they are older than the source file

      Ans. cp -u
   4. Check if whitespace characters in a file are tabs or spaces

      Ans. grep $'\t' <filename>
   5. View the last 5 commands you issued

      Ans. history | tail -n 5

5. Provide a glob or extended glob pattern that would match and not match the sets of filenames in the table below. Give 1 pattern for each row of the table

#	**Match these strings**	**Don't match these strings**
1	README.txt, data.tsv, figure.tiff	Homework.pdf, data_to_analyze/, doc.rtf
2	SRR124515, ERR123252, SRR3161371316	PRR161356 LRR124636 error.txt
3	File.txt, another.pdf	temp.csv, data.csv

4	sample_reads_1.fastq, sample_reads_2.fastq, SRR1352235_1.fq, SRR1352235_2.fq	sample_assembly.fasta, SRR1352235_assembly.fasta, sample_feats.bed, SRR1352235_feats.bed, longreads.fastq
5	Samples/a/assembly.fasta, Samples/b/assembly.fasta	assembly.fasta, Samples/assembly.fasta

Ans.

1. Match: `*.{txt,tsv,tiff}`
   Don't Match: `Homework.pdf`, `data_to_analyze/`, `doc.rtf`

2. Match: `+(SRR|ERR)+([0-9])`
   Don't Match: `PRR161356`, `LRR124636`, `error.txt`

3. Match: `*.*(txt|pdf)`
   Don't Match: `temp.csv`, `data.csv`

4. Match: `sample_reads_[1-2].fastq`, `SRR1352235_[1-2].fq`
   Don't Match: `sample_assembly.fasta`, `SRR1352235_assembly.fasta`, `sample_feats.bed`, `SRR1352235_feats.bed`, `longreads.fastq`

5. Match: `Samples/*/assembly.fasta`
   Don't Match: `assembly.fasta`, `Samples/assembly.fasta`

6. Redirecting outputs
   1. Pick a command that produces stdout, run it, and direct its stdout to a file

```
mspillai2105@MeenakshisG15:~$ echo "Direct this sentence to a file" > file3
mspillai2105@MeenakshisG15:~$ ls
file1  file1_copy.txt  file2  file3  home
mspillai2105@MeenakshisG15:~$ cat file3
Direct this sentence to a file
```

   2. ls a path that does not exist in your current directory. Which output stream does the message you see come from?

```
mspillai2105@MeenakshisG15:~$ ls /new_dir
ls: cannot access '/new_dir': No such file or directory
```

      Ans. The message comes from the std error stream

   3. Rerun the command from step 2, but now direct the output to a file

```
mspillai2105@MeenakshisG15:~$ ls /new_dir >output_and_error.txt 2>&1
mspillai2105@MeenakshisG15:~$ ls
file1  file1_copy.txt  file2  file3  home  output_and_error.txt
mspillai2105@MeenakshisG15:~$ cat output_and_error.txt
ls: cannot access '/new_dir': No such file or directory
mspillai2105@MeenakshisG15:~$
```

   4. ls both a non-existent path and "./" (i.e., provide two positional inputs). Direct the stdout to one file and the stderr to another file.

```
mspillai2105@MeenakshisG15:~$ ls new_dir ./>stdout.txt 2>stderr.txt
mspillai2105@MeenakshisG15:~$ cat stdout.txt
./:
file1
file1_copy.txt
file2
file3
home
output_and_error.txt
stderr.txt
stdout.txt
mspillai2105@MeenakshisG15:~$ cat stderr.txt
ls: cannot access 'new_dir': No such file or directory
```

5. Use grep to find the help message entry for the -l option of ls (hint: “-“ is a special character interpreted by bash so you need to get around that somehow)

```
mspillai2105@MeenakshisG15:~$ ls --help | grep -- '-l'
      --author               with -l, print the author of each file
      --block-size=SIZE      with -l, scale sizes by SIZE when printing them;
  -c                         with -lt: sort by, and show, ctime (time of last
                               with -l: show ctime and sort by name;
  -f                         do not sort, enable -aU, disable -ls --color
      --format=WORD          across -x, commas -m, horizontal -x, long -l,
                               single-column -1, verbose -l, vertical -C
      --full-time            like -l --time-style=full-iso
  -g                         like -l, but do not list owner
  -h, --human-readable       with -l and -s, print sizes like 1K 234M 2G etc.
  -H, --dereference-command-line
      --dereference-command-line-symlink-to-dir
  -l                         use a long listing format
  -n, --numeric-uid-gid      like -l, but list numeric user and group IDs
  -N, --literal              print entry names without quoting
  -o                         like -l, but do not list group information
                             with -l, WORD determines which time to show;
      --time-style=TIME_STYLE  time/date format with -l; see TIME_STYLE below
  -u                         with -lt: sort by, and show, access time;
                               with -l: show access time and sort by name;
 2  if serious trouble (e.g., cannot access command-line argument).
```

6. How many commands are there in your “/bin” dir?

7. Data cleaning. Bioinformaticians often have to work with data generated by others. Perform the following operations to tidy the data in file “ex1.bed” provided on Canvas

    1. Check if the file uses windows line endings instead of unix line endings

```
mspillai2105@MeenakshisG15:~$ file ex1.bed | grep CRLF
ex1.bed: ASCII text, with CRLF, CR line terminators
```

    Ans. The CRLF message indicates it has windows line endings

    2. Remove the windows line endings and output the new version to a new file, preserving the original file (always good practice)

```
mspillai2105@MeenakshisG15:~$ file ex1.bed | grep CRLF
ex1.bed: ASCII text, with CRLF, CR line terminators
mspillai2105@MeenakshisG15:~$ touch new_file.bed
mspillai2105@MeenakshisG15:~$ dos2unix ex1.bed new_file.bed
dos2unix: converting file ex1.bed to Unix format...
dos2unix: converting file new_file.bed to Unix format...
mspillai2105@MeenakshisG15:~$ file new_file.bed
new_file.bed: empty
mspillai2105@MeenakshisG15:~$ file new_file.bed |grep CRLF
```

    3. Remove the header lines starting with “#” and output the new version to a new file

```
mspillai2105@MeenakshisG15:~$ grep -v '^#' ex1.bed > new_file_no_headers.bed
mspillai2105@MeenakshisG15:~$ cat ex1.bed
# File generated by Person S. Unknown April 8th 2020
# Using script make_my_bed.py
chr1    2090    2475
chr1    2584    3083
chr1    4692    4832
chr1    4692    5658
chr1    4901    5658
chr1    5805    6469
chr1    5810    6469
chr1    6628    6716
```

```
chr1    21094581        21098776
chr1    21099023        21103962
chr1    21104051        21140570
chr1    21104051        21149082
chr1    21140475        21140570
chr1    21141410        21149082
chr1    21149188        21168553
chr1    21149191        21168553
mspillai2105@MeenakshisG15:~$ cat new_file_no_header.bed
cat: new_file_no_header.bed: No such file or directory
mspillai2105@MeenakshisG15:~$ cat new_file_no_headers.bed
chr1    2090    2475
chr1    2584    3083
chr1    4692    4832
chr1    4692    5658
chr1    4901    5658
chr1    5805    6469
chr1    5810    6469
chr1    6628    6716
chr1    6628    6720
```

8. Summarizing real data using bash commands. The following questions relate to the cleaned version of the “ex1.bed” file you generated above. BED format is a commonly used format for storing the location of features in an assembly. The provided file includes the three mandatory columns of a bed file: Sequence ID, start, and stop positions. Using bash commands answer the following commands about these data

1. How many sequence IDs are present in the file?
2. How many different start positions are there?
3. What is the highest number of features starting at the same start position?
4. How many features start in the first 10Kb of the sequence?
5. How many features start and end in the first 10Kb of the sequence?

EXTRA CREDIT (5 points)

6. Which feature is the largest? Show your work

```
mspillai2105@MeenakshisG15:~$ cut -f 1 new_file_no_headers.bed | sort -u | wc -l
1
mspillai2105@MeenakshisG15:~$ cut -f 2 new_file_no_headers.bed | sort -u | wc -l
3078
mspillai2105@MeenakshisG15:~$ cut -f 2 new_file_no_headers.bed | sort | uniq -c | sort -nr | head -n 1
      5 3765268
mspillai2105@MeenakshisG15:~$ awk '$2 <= 10000' new_file_no_headers.bed | wc -l
29
mspillai2105@MeenakshisG15:~$ awk '$2 >= 0 && <= 10000' new_file_no_headers.bed | wc -l
awk: cmd. line:1: $2 >= 0 && <= 10000
awk: cmd. line:1:            ^ syntax error
0
mspillai2105@MeenakshisG15:~$ awk '$2 >= 0 && $3 <= 10000' new_file_no_headers.bed | wc -l
22
mspillai2105@MeenakshisG15:~$ awk '{print $3 - $2}' new_file_no_headers.bed | sort -n | tail -n 1
400497
mspillai2105@MeenakshisG15:~$
```