Programming for Bioinformatics | BIOL7200

Week 2 Exercise

Exercises

1. Using `who` and `finger`
   1. Use `who` to see yourself logged into the machine

   ANS: **who**
   2. Find ***all*** the logins on the machine

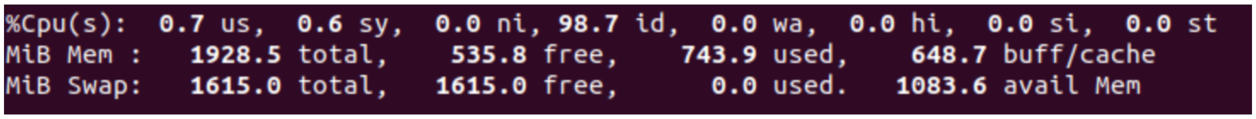
   ANS: **who -a**

   finger -s
   3. Use `finger` to find information about yourself on your Linux VM/whatever you are using

   ANS: **finger -l**

2. Monitoring system usage with `top` and `free`
   1. Use `top` to look at what's running on the computer

   ANS: **top**
   2. Find the CPU and memory usage of a program; how much memory is left?



```
%Cpu(s):  0.7 us,  0.6 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1928.5 total,    535.8 free,    743.9 used,    648.7 buff/cache
MiB Swap:   1615.0 total,   1615.0 free,      0.0 used.   1083.6 avail Mem
```

   ANS: **535.8 mebibytes**
   3. Order the processes by CPU usage, ascending and descending

   ANS: **top -o %CPU (gives descending)**

   top -o -%CPU (gives ascending)
   4. Do the same for memory

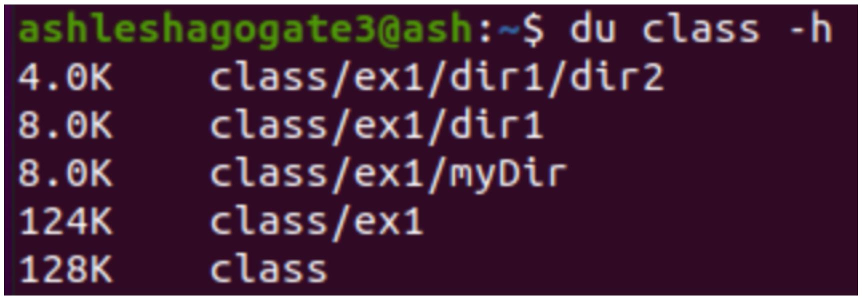
   ANS- **top -o %MEM (gives descending)**

   top -o -%MEM (gives ascending)

3. Looking at hard disk space
   1. Use `df` to find the free space on your machine's hard disk

   ANS: **df - -total -h**
   2. Use `du` to find the total size of some directory; make the output easy to read, e.g. '315M

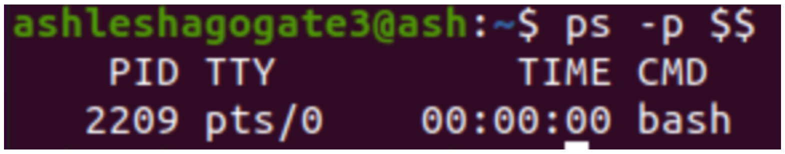
   ANS- **du class -h**



```
ashleshagogate3@ash:~$ du class -h
4.0K	class/ex1/dir1/dir2
8.0K	class/ex1/dir1
8.0K	class/ex1/myDir
124K	class/ex1
128K	class
```

4. Finding running processes
    1. Find all the processes that you have running using ps
    ANS: **ps**
    2. Find **all** processes that are running on the machine
    ANS: **ps -e**
    3. Find all of the processes being run by the user ‘root’ in user format
    ANS: **ps -U root -u root u**

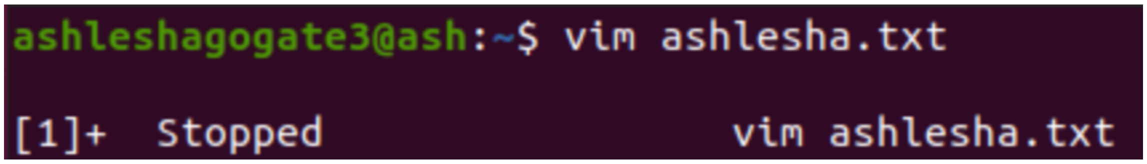
5. Murder most foul
    1. Find the PID of your login shell using ps
    ANS: **ps -p $$**
    The PID is 2209



```
ashleshagogate3@ash:~$ ps -p $$
    PID TTY          TIME CMD
   2209 pts/0    00:00:00 bash
```

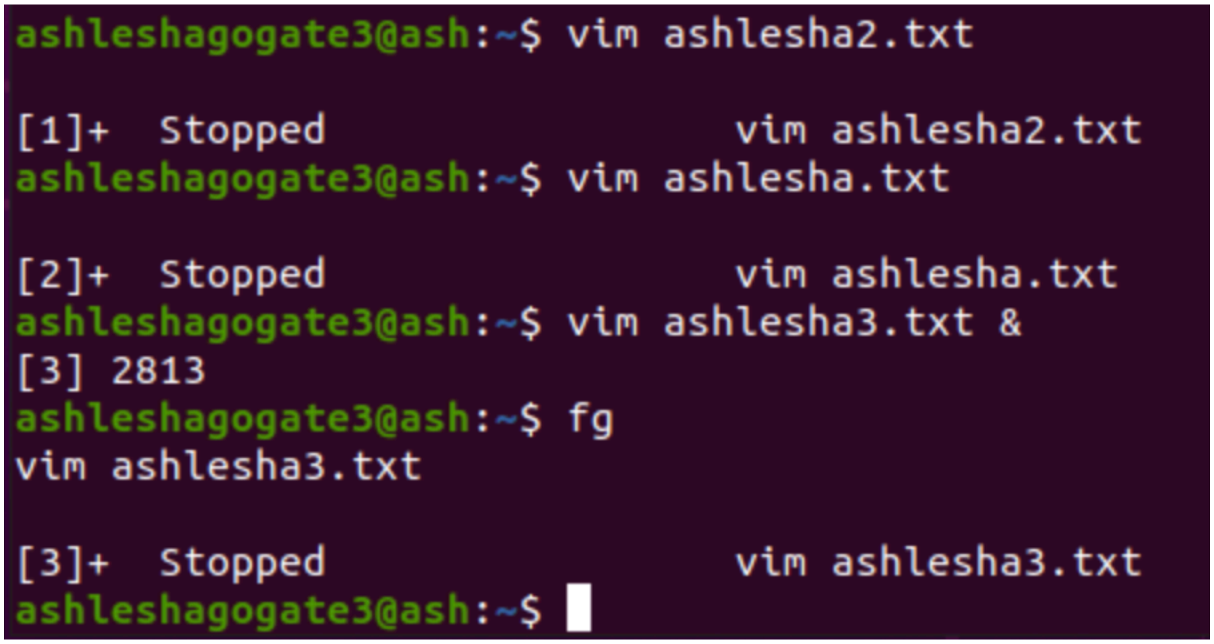
    2. Kill your login shell with a command
    ANS: **kill -9 2209**
    3. Open a new shell
    ANS: **done (control+option+T)**

6. Managing jobs
    1. Start editing a file with vim, then put it in the background using Ctrl + Z
    ANS: **vim ashlesha.txt**



```
ashleshagogate3@ash:~$ vim ashlesha.txt

[1]+  Stopped                 vim ashlesha.txt
```

    2. Find it using the jobs command
    ANS: **jobs**
    3. Restore it to the foreground using fg
    ANS: **fg**
    4. Start a new job in the background using & after the command
    ANS: **vim ashlesha3.txt &**
    5. Move the new job to the foreground using fg
    ANS: **fg**



```
ashleshagogate3@ash:~$ vim ashlesha2.txt

[1]+  Stopped                 vim ashlesha2.txt
ashleshagogate3@ash:~$ vim ashlesha.txt

[2]+  Stopped                 vim ashlesha.txt
ashleshagogate3@ash:~$ vim ashlesha3.txt &
[3] 2813
ashleshagogate3@ash:~$ fg
vim ashlesha3.txt

[3]+  Stopped                 vim ashlesha3.txt
ashleshagogate3@ash:~$
```

6. What is the difference between the commands Ctrl + c and Ctrl + z

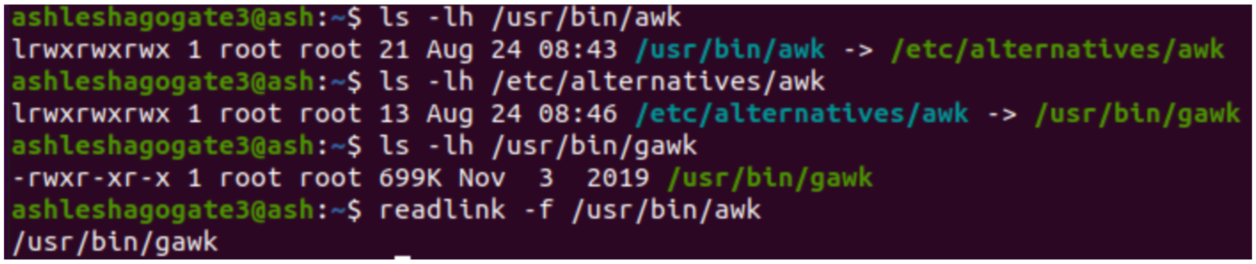
ANS: **ctrl+c terminates the process and ctrl+z pauses the process**

7. Finding the location of binaries, linking them and alias
    1. Find out where the awk command is located using which

    ANS: **which awk**

    2. Find out if it is a link or a real binary. Follow this through until you have found the real binary.

    ANS: **ls -lh /usr/bin/awk repeatedly OR**
    readlink -f /usr/bin/awk
    The real binary is /usr/bin/gawk



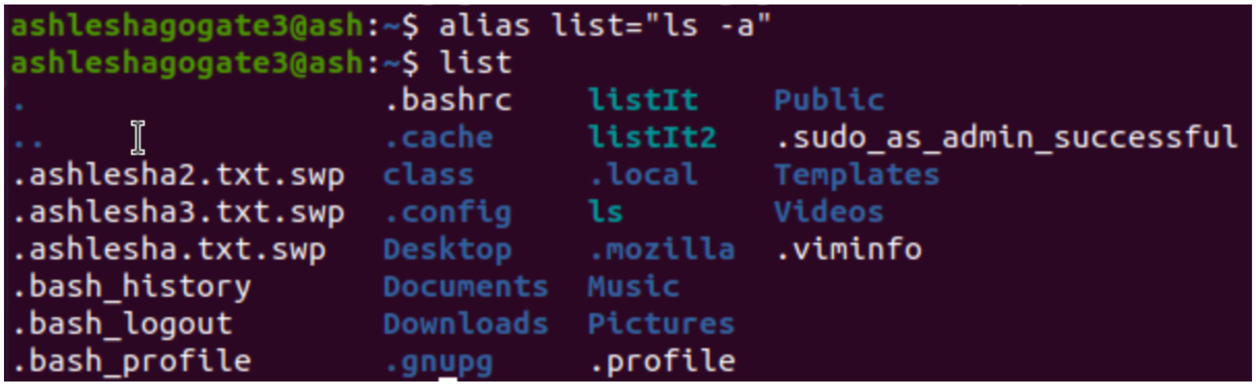
```
ashleshagogate3@ash:~$ ls -lh /usr/bin/awk
lrwxrwxrwx 1 root root 21 Aug 24 08:43 /usr/bin/awk -> /etc/alternatives/awk
ashleshagogate3@ash:~$ ls -lh /etc/alternatives/awk
lrwxrwxrwx 1 root root 13 Aug 24 08:46 /etc/alternatives/awk -> /usr/bin/gawk
ashleshagogate3@ash:~$ ls -lh /usr/bin/gawk
-rwxr-xr-x 1 root root 699K Nov  3  2019 /usr/bin/gawk
ashleshagogate3@ash:~$ readlink -f /usr/bin/awk
/usr/bin/gawk
```

    3. Create a symbolic link to the ls command in your home directory called listIt using the ln command

    ANS: **ln -s /usr/bin/ls listIt**

4. Create an alias for ls -a

ANS: **alias list="ls -a"**



```
ashleshagogate3@ash:~$ alias list="ls -a"
ashleshagogate3@ash:~$ list
.                    .bashrc    listIt    Public
..                   .cache     listIt2   .sudo_as_admin_successful
.ashlesha2.txt.swp   class      .local    Templates
.ashlesha3.txt.swp   .config    ls        Videos
.ashlesha.txt.swp    Desktop    .mozilla  .viminfo
.bash_history        Documents  Music
.bash_logout         Downloads  Pictures
.bash_profile        .gnupg     .profile
```

8. Hard vs Symbolic links

1. Create a file randomFile.txt and type the following in it:
   This is some text.
   This is some more text.
   And some more…

ANS: **vi randomFile.txt**

2. Now create a symbolic link for randomFile.txt; name the link symLink.txt

ANS: **ln -s randomFile.txt symLink.txt**

3. Now create a hard link for randomFile.txt; name the link hardLink.txt

ANS: **ln randomFile.txt hardLink.txt**

4. Create another file anotherFile.txt. Put this text in it:
   1234
   5678
   9101

ANS: **vi anotherFile.txt**

5. Now delete randomFile.txt; which of the two links still works?

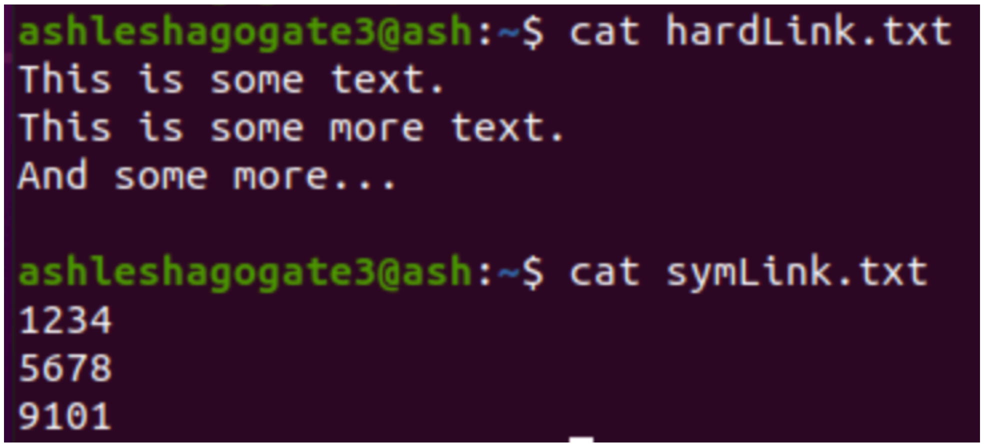
ANS: **rm randomFile.txt**

The hard link works when we display its content using cat hardline.txt
The symbolic link doesn't work. It says no such file or directory.

6. Rename anotherFile.txt to randomFile.txt. Which of the two still works? If both work, is there a difference between the two? If only one works, why?

ANS: **Both the links work.**

The hardlink still shows the original content of the randomFile.txt.
The symbolic link now shows the original content of anotherFile.txt



```
ashleshagogate3@ash:~$ cat hardLink.txt
This is some text.
This is some more text.
And some more...

ashleshagogate3@ash:~$ cat symLink.txt
1234
5678
9101
```

9. Downloading UCSC Genome Browser files with `wget` Toy exercise – the objective here is only to learn how to use `wget`. You don't need to wait for each file to finish downloading; you can quit the download if you have determined that you are using the correct command. Once you see a file getting downloaded, kill the download process if you want. Read wget manual for helpful options.

   1. What is the UCSC Genome Browser? What information does it store? What organisms does it focus on? What is the FTP site for accessing the databases?

   ANS: **UCSC genome browser is a downloadable genome browser hosted by the University of California Santa Cruz.**

We can use it to display a section of a genome of our choice and modify the view to any zoom scale. It also contains annotation tracks which provide a lot of information such as histone modifications, conservation of genes across different species and so on.

It mainly focuses on eukaryotic organisms.

The FTP site for accessing the database is ftp hgdownload.soe.ucsc.edu

   2. Navigate to the ENCODE repository on the UCSC FTP site.

      1. What is an ENCODE repository?

      ANS: **ENCODE stands for The Encyclopedia of DNA Elements. It is a whole genome database. It can be used to understand how the genome elements are linked to certain diseases. We can also get publicly available sequencing data for organisms.**

      2. Download a file (pick a small one of your choice) from the ENCODE repository on the UCSC Genome Browser with `wget` and briefly explain what data it contains.

      ANS: **wget https://hgdownload.soe.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeGisRnaSeq/wgEncodeGisRnaSeqK562CytosolPapRawData3Rep1.fastq.gz**

It is a fastq file which contains the RNA-seq raw data for the experiment. The file is gun-zipped and should be unzipped to view the contents. Upon unzipping, the file is a text like file which displays the nucleotide sequence.

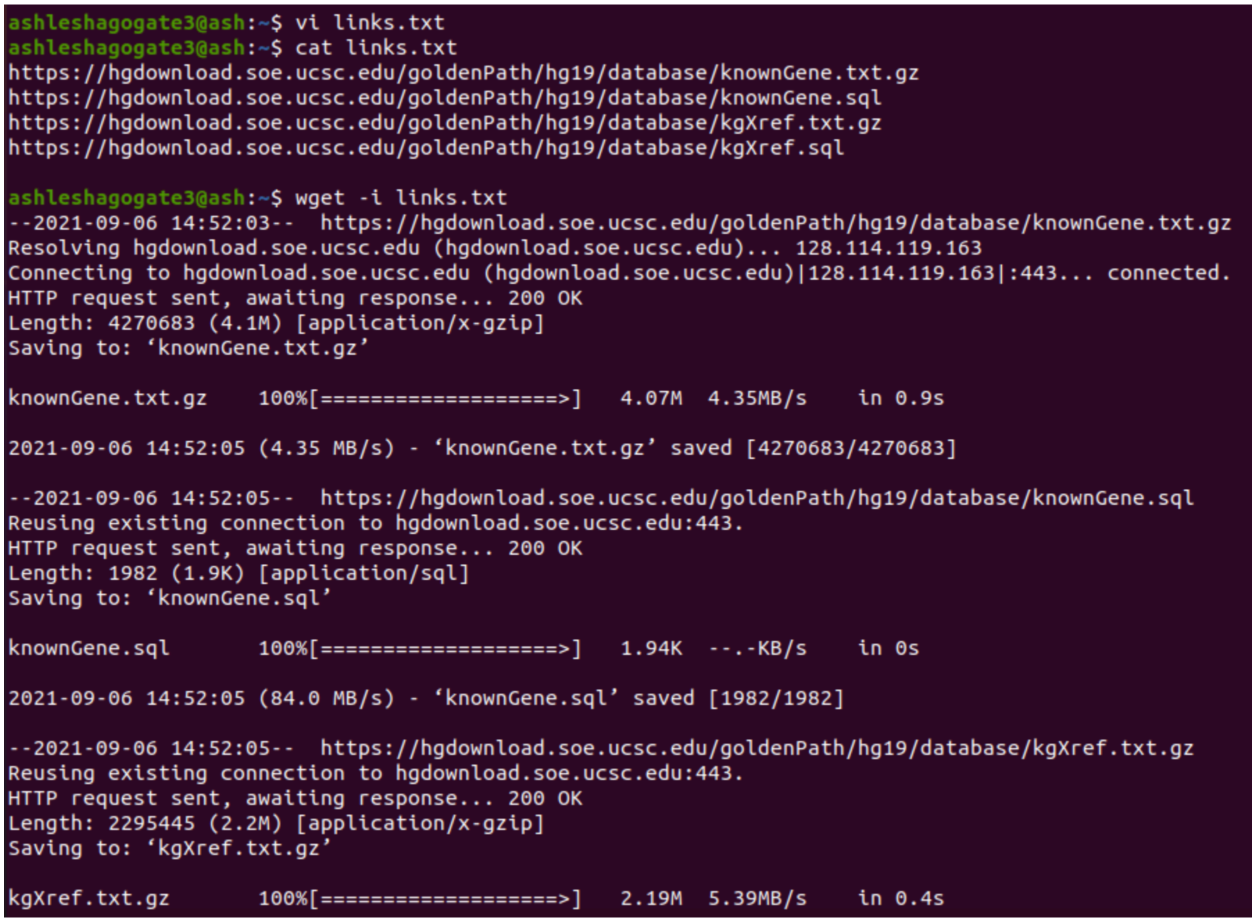
3. Download all of the Pol2b binding data available in one line with wget

ANS: **wget ftp://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/*Pol2b***
24 files were downloaded

4. The database folder located in hg19 in goldenPath is an important folder. We are going to fetch four files: knownGene.txt.gz, knownGene.sql, kgXref.txt.gz and kgXref.sql. Copy their link addresses and paste in a file. Now use wget to download the files by reading the URLs from the file you just created. In other words, this time the input URL comes from a file rather than from you giving it in the command, and there are multiple URLs. **Hold on to these files; we will use them later during the course.**

ANS: **vi links.txt**

Paste all the links in the file.
wget -i links.txt



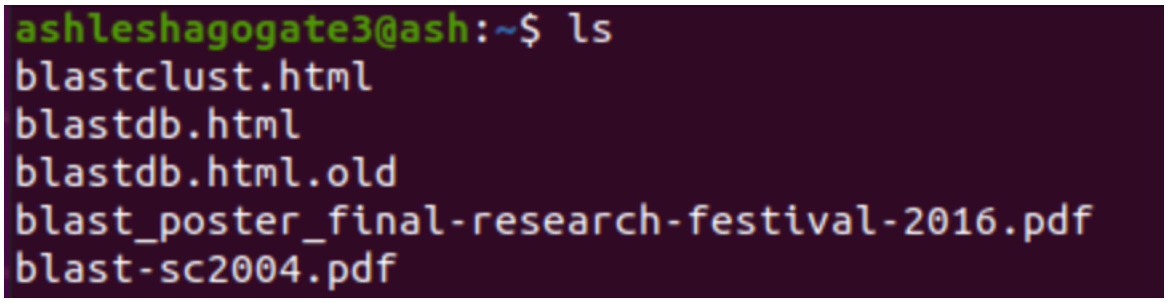
```
ashleshagogate3@ash:~$ vi links.txt
ashleshagogate3@ash:~$ cat links.txt
https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.txt.gz
https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.sql
https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/kgXref.txt.gz
https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/kgXref.sql

ashleshagogate3@ash:~$ wget -i links.txt
--2021-09-06 14:52:03--  https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.txt.gz
Resolving hgdownload.soe.ucsc.edu (hgdownload.soe.ucsc.edu)... 128.114.119.163
Connecting to hgdownload.soe.ucsc.edu (hgdownload.soe.ucsc.edu)|128.114.119.163|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4270683 (4.1M) [application/x-gzip]
Saving to: ‘knownGene.txt.gz’

knownGene.txt.gz    100%[===================>]   4.07M  4.35MB/s    in 0.9s

2021-09-06 14:52:05 (4.35 MB/s) - ‘knownGene.txt.gz’ saved [4270683/4270683]

--2021-09-06 14:52:05--  https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/knownGene.sql
Reusing existing connection to hgdownload.soe.ucsc.edu:443.
HTTP request sent, awaiting response... 200 OK
Length: 1982 (1.9K) [application/sql]
Saving to: ‘knownGene.sql’

knownGene.sql       100%[===================>]   1.94K  --.-KB/s    in 0s

2021-09-06 14:52:05 (84.0 MB/s) - ‘knownGene.sql’ saved [1982/1982]

--2021-09-06 14:52:05--  https://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/kgXref.txt.gz
Reusing existing connection to hgdownload.soe.ucsc.edu:443.
HTTP request sent, awaiting response... 200 OK
Length: 2295445 (2.2M) [application/x-gzip]
Saving to: ‘kgXref.txt.gz’

kgXref.txt.gz       100%[===================>]   2.19M  5.39MB/s    in 0.4s
```

5. Test if the URL http://www.ncbi.nlm.nih.gov/SNP/snp_ref.cgi?searchType=adhoc_search&type=rs&rs=rs12345 is correct using wget. [Hint: Have you seen error 404 while accessing a website? See if HTTP Status codes can help].

ANS: **The output gets redirected to a file named wget-log. Upon displaying the contents of the file using the cat command, the following status codes are displayed-**
301: The resource has moved permanently to a new location
302: The resource has moved temporarily to a new location
404: Page not found

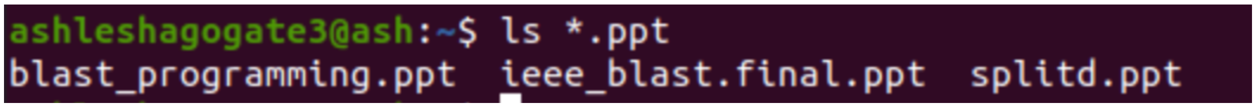
10. Write one-liners to perform the following actions with `wget` or `curl`
    1. Download the file `taxdb.tar.gz` from `ftp://ftp.ncbi.nlm.nih.gov/blast/db/`
    ANS: **wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/taxdb.tar.gz**

    2. Download all files that start with "blast" from `ftp://ftp.ncbi.nlm.nih.gov/blast/documents/`
    ANS: **wget ftp://ftp.ncbi.nlm.nih.gov/blast/documents/blast***



```
ashleshagogate3@ash:~$ ls
blastclust.html
blastdb.html
blastdb.html.old
blast_poster_final-research-festival-2016.pdf
blast-sc2004.pdf
```

    3. Download all "ppt" files from `ftp://ftp.ncbi.nlm.nih.gov/blast/demo/`
    ANS: **wget ftp://ftp.ncbi.nlm.nih.gov/blast/demo/*.ppt**



```
ashleshagogate3@ash:~$ ls *.ppt
blast_programming.ppt  ieee_blast.final.ppt  splitd.ppt
```

11. The `PATH` to enlightenment

The `PATH` environmental variable tells the shell where to look for executables. When you use the ls command, it works because the shell knows to look for it on the `PATH` .

    1. Print your `PATH` environmental variable to the screen. What are the directories in it? Some of them should look familiar at this point.
    ANS: **echo $PATH**
    2. Make a directory called bin in your home directory and put a symbolic link to some already existing executable in the newly created bin directory. Call the link something different from the actual executable or this is all for naught.

ANS: **mkdir bin**
cd bin
ln -s /usr/bin/mv MoveOrRename

3. Add you bin directory to your PATH environmental variable. Keep in mind what character separates the different directories in the PATH environmental variable. If you mess up your PATH , then just leave the shell and make a new one.

ANS: **export PATH=$PATH:/home/ashleshagogate3/bin/**

4. Try running the executable through the symbolic link. Did it work? Use which to see the alias.

ANS: **MoveOrRename Music music**
Yes it worked!
which MoveOrRename

5. Add your new bin directory to your PATH in your login file. Make sure you can still access all of the programs on your PATH.

ANS: **echo 'export PATH=/home/ashleshagogate3/bin:$PATH' >>~/.profile**

6. Congratulations! You now have a centralized folder that will be on your PATH every time you log in. You can install executables here if you don't have permission to install things.
7. If you're on a system with multiple users, what might be an advantage of install an executable in one of the directories on the default PATH ?

ANS: **It can be accessed globally even if you are installing an app outside of the default directory structure and need to run commands and get a way around permissions.**

12. The C compilers

There are various C compilers available today. They pretty much do the same thing: turn C code into a program, but they do it with widely varying efficiency.

1. Find out what your cc command points to. In other words, what is your default C compiler?

ANS: **sudo apt install gcc**
cc -v
The default C compiler is gcc version 9.3.0

2. Compile the haveSomeNumbers.c file from Canvas and run the result (you're going to need the -o flag. Read about it before using it). Add the results to your PATH and run it that way.

ANS: **cc -o haveSomeNumbers haveSomeNumbers.c**
./haveSomeNumbers
export PATH=$PATH:/home/ashleshagogate3/class/ex2/haveSomeNumbers
haveSomeNumbers (displayed the output)

3. C is beyond the scope of this course, but try to figure out what the haveSomeNumbers.c file does anyway. This program is very simple. Many, many bioinformatics programs are written in C.

ANS: **The program incorporates a for loop that starts with value of "i" variable as 1 and prints it. It increments the variable by 1 for every run and prints it in a new line. This continues until the variable reaches 100**

4. Look up comparisons of `gcc` and `icc` online. What are the advantages and disadvantages of each? What other C compilers exist?

ANS: **ICC stands for Intel C++ Compiler. It is a bunch of C and C++ compilers which can be used on different operating systems. The full form of GCC is GNU Compiler Collection. It is an open source software.**
Advantage of GCC: The overall compile time is lesser for gcc and higher for icc. It is very easy to download.
Disadvantage of GCC: certain string manipulation functions in Linux gcc compiler experience problems.
Advantage of ICC: Very good SIMD support, Supports both automatic parallelization (multi core optimzations), as well as manual (through OpenMP)
Disadvantage of ICC: only works on Intel CPU's
List of c compilers- Acorn C, Arm compiler, MCP, Oracle C compiler, XL C