**-Ampersan symbol : &**

**FOR NAVIGATION:**

ls /mnt

cd /mnt/c

cd users

cd Hp

cd Desktop

cd Project

**FOR GETTING INTO PREVIOUS DIRECTORY:**

cd –

**TO LIST ALL THE CONTENTS:**

ls

**-l,** to list the files and folders in the current directory with a detailed listing.

**-lh,** to use human-friendly file sizes.

**-lha,** to include hidden files.

**TO EXECUTE TWO OR MORE COMMANDS SIMULTANEOUSLY:**

Example: ls ; pwd ; ps -ubey207518

**TO GET INTO THE ROOT DIRECTORY:**

sudo -i

**TO GET OUT OF THE ROOT DIRECTORY:**

exit or ctrl+D

**FOR MAKING A NEW DIRECTORY:**

mkdir directory\_name

**FOR ADDING A FILE:**

touch project/new.(txt/docx/pptx……)

**FOR EDITING A TEXT FILE:**

nano new.txt

**FOR VIEWING A FILE**

cat new.txt

**TO MOVE A FILE FROM DOWNLOADS TO A DIRECTORY IN DESKTOP:**

mv file\_name /mnt/c/Users/Hp/Desktop/Directory\_name

**FOR REMOVING A PACKAGE:**

sudo apt purge package\_name

**FOR INSTALLING ZIP UNZIP UTILITY:**

sudo apt-get install zip unzip

**FOR UNZIPPING A .tgz FILE:**

tar zxvf fileNameHere.tgz

**FOR UNZIPPING A .zip FILE:**

unzip file\_name.zip

**FOR UNPACKING A FILE:**

bash file\_name.sh

**FOR ADDING A REPOSITORY:**

sudo add-apt-repository repository\_address

**TO DOWNLOAD A .DEB FILE:**

sudo dpkg -i package\_name.deb

**TO REMOVED A .DEB FILE:**

sudo apt purge package\_name.deb

**FOR ADDING THE CHANGE TO THE FOLDER**

git add new.txt

**FOR COMMITING THE CHANGE**

git commit -m “UPDATED”

**FOR PUSHNG THE CHANGES TO THE GITHUB ACCOUNT**

git push origin master

**FOR PULLING THE CHANGES OF THE GITHUB ACCOUNT INTO THE LOCAL FOLDER:**

git pull origin master

**TO REMOVE ORIGIN REPOSITORY**

git remote rm origin

**MY ANACONDA 3 LOCATION:**

/home/bey207518/anaconda3

**TO CREATE A NEW CONDA WORKSTATION IN SSH:**

conda create -n Nirmal

**CREATE A NEW CONDA WORKSTATION IN UNIX SHELL:**

conda create –name Nirmal python=version

**TO ACTIVATE CONDA WORKSTATION:**

conda activate Nirmal

**TO REMOVE A PACKAGE FROM CONDA WORKSTATION:**

conda remove -n Nirmal package\_name

**TO INSTALL MAMBA PACKAGE IN CONDA WORKSTATION:**

conda install -c conda-forge mamba

**TO REMOVE ANACONDA:**

conda install anaconda-clean

anaconda-clean

sudo rm -rf ~/anaconda3

**TO COMPILE AND EXECUTE C PROGRAM**

gcc file\_name.c -o output\_file\_name (compile the file)

./output\_file\_name (execute the file)

**TO RUN A R PROGRAM**

Write the program in the VS code with .r extension and run it in terminal using the following command:

rscript file\_name.r

**Nohup:** Nohup, short for no hang up is a command in Linux systems that keep processes running even after exiting the shell or terminal.

**Nohup command**: nohup command arguments. The output of this command will be saved in nohup.out.

**Less:** The less command allows you to view files without opening an editor. It’s faster to use, and there’s no chance of you inadvertently modifying the file.

Less nohup.out will display the result of the ongoing command.

**Alias command:** The alias command lets you give your own name to a command or sequence of commands.

alias cls=clear

Aliases defined on the command line will die with the terminal window. When we close it, they are gone.

**Curl:** The curl command is a tool to retrieve information and files from Uniform Resource Locators (URLs) or internet addresses.

Curl url ( the content of the file are scrolled rapidly in the terminal window but not saved to our computer)

Curl url -o file ( -o option is the output option which save the data retrieved from the url into the file)

Curl -s url -o file (if we don’t want to see the download progress information, we will use the -s (silent) option.)

**Df :** The df command shows the size, used space and available space on the mounted filesystems of our computer.

**-h option** displays the sizes in Mb or Gb instead of in bytes.

**-x option** allows us to tell df to discount filesystems we are not interested in.

**Diff:** The diff command compares two text files and shows the differences between them.

**-y option** shows the line differences side by side.

**-w option** lets us specify the maximum line width to use to avoid wraparound lines. Example -w 70.

**--suppress-common-lines** prevents diff from listing the matching lines, letting you focus on the lines which have differences.

Diff -y -w 70 alpha1.txt alpha2.txt –suppress-common-lines

**Du:**  It is short for disk usage, is used to estimate file space usage.

**-a option** write count of all files, not just directories.

**-apparent-size** print apparent sizes, rather than disk usage.

**-d, -max-depth=N** print total for directory only if it is N or fewer levels below command line argument.

**-h** print sizes in human readable format in MB or GB.

**-S, -separate-dirs:** for directories, do not include size of subdirectories.

**-s, -summarize** display only total for each directory.

**Echo:** The echo command prints a string of text to the terminal window.

Echo Hello World

**Finger:** The finger command gives us a short dump of information about a user, including the time of the user’s last login, the user’s home directory and the user account’s full name.

Finger bey207518

**Free:** The free command gives us a summary of the memory usage with our computer. It does this for both the main RAM and swap memory.

Free -h

**Grep:** The grep utility searches for lines which contain a search pattern. The grep command can also search the contents of the files.

grep train \*.txt (The output lists the name of the file and shows the lines that match. The matching text is highlighted.)

**gzip:** The gzip command compresses files. By default, it removes the original file and leaves us with the compressed version. To retain both the original and the compressed version, use the -k(keep) option.

gzip -k file.c

**head:** The head command gives us a listing of the first 10 lines of a file.

**-n option** can be used if we want to see fewer or more lines.

head -n 5 file.txt

**passwd:** The passwd command lets us change the password for a user. Just type passwd to change our own password.

sudo passwd bey207518

**ping:** The ping command lets us verify that we have notebook connectivity with another network device.

ping ip address. The ping command will run until we stop it with cntrl+c.

**ps:** The ps command lists running processes.

**-u option** to see all the processes related to a particular user. This is likely to be a long list, so for convenience pipe it through less.

ps -u user | less

**-e option** to see every process that is running.

ps -e | less

**shutdown:** the shutdown command lets us shut down or reboot our linux system. Using shutdown with no parameters will shut down our computer in one minute.

shutdown now; to shut down immediately.

shutdown 23:00 ; scheduling a shutdown.

**ssh:** Use the ssh command to make a connection to a remote Linux computer. To make a connection, we must provide the user name and the Ip address or domain name of the remote computer.

ssh [bey207518@baadal.iitd.ac.in](mailto:bey207518@baadal.iitd.ac.in)

**tail:** The tail command gives us a listing of the last 10 lines of a file.

**-n option** can be used if we want to see fewer or more lines.

tail -n 5 file.txt

**tar:** With the tar command, we can create an archive file that contain many other files.

**-c option** is the create option

**-v option** gives some visual feedback by listing the files to the terminal window as they added to the archive.

**-f option** is followed by the desired name of the archive.

tar -cvf file.tar directory\_name/

**-z (gzip) option** can be used if we want the archive file to be compressed.

tar -cvzf file.tar.gz directory\_name/

**-j (bzip2) option** can compress the archive file using a superior compression algorithm.

tar -cvjf file.tar.bz2 directory\_name/

The -j option is noticeably slower than the -z option.

**-x (extract) option** can be used to extract the files.

tar -xvf file.tar

tar -xvzf file.tar.bz

tar -xvjf file.tar.bz2

**.csv file extension:** comma separated values, **.tsv file extension:** tab separated values.

**Xargs command:** It is a unix command which can be used to build and execute commands from standard input.

xargs [OPTIONS] [COMMAND [initial arguments]]

**xargs options :**  
**-0 :** input items are terminated by null character instead of white spaces  
**-a file :** read items from file instead of standard input  
**–delimiter = delim :** input items are terminated by a special character  
**-E eof-str :** set the end of file string to eof-str  
**-I replace-str :** replace occurrences of replace-str in the initial arguments with names read from standard input  
**-L max-lines :** use at-most max-lines non-blank input lines per command line.  
**-p :** prompt the user about whether to run each command line and read a line from terminal.  
**-r :** If the standard input does not contain any nonblanks, do not run the command  
**-x :** exit if the size is exceeded.  
**–help :** print the summary of options to xargs and exit  
**–version :** print the version no. of xargs and exit

**-n :** This option specifies the number of arguments to be passed to the given command.

xargs -n 1 basename: The number of arguments that are read from the standard input is limited to 1.

**cut:** Cut is a command-line utility that allows us to cut parts of lines from specified files or piped data and print the results to standard output.

**-f option** select by specifying a field, a set of fields, or a range of fields.

**-b option** select by specifying a byte, a set of bytes, or a range of bytes.

**-c option** select by specifying a character, a set of characters, or a range of characters.

**-d option** specify a delimiter that will be used instead of the default “TAB” delimiter.

cut -f 1-2 -d”\_” : combine and cut the 1 and 2 portion of the file before \_.

**NOTE:** stdin, stdout, and stderr are three data streams created when we launch a Linux command.

In Linux, stdin is the standard input stream. This accepts text as its input. Text output from the command to the shell is delivered via the stdout (standard out) stream. Error messages from the command are sent through the stderr (standard error) stream.

File descriptor 2 represents standard error, other include 0 for standard input and 1 for standard output.

2> $raw/alignments/${nam}.sorted.stderr means to redirect standard error to $raw/alignments/${nam}.sorted.stderr

**/dev directory:** /dev is the location of special or device files. It is very interesting directory that highlights one important aspect of the linux filesystem - everything is a file or a directory.

/dev/stdin: standard input of the shell process.

/dev/stdout: standard output of the shell process.

/dev/stderr: standard error of the shell process.

**>** is used to overwrite (“[*clobber*](https://en.wikipedia.org/wiki/Clobbering)”) a file and **>>** is used to append (add) to a file.

**wc:** wc stands for word count. It is used to find out number of lines, word count, byte and characters count. By default, it displays four-columnar output.

**-l option** prints the number of lines present in a file. With this option wc command displays two-columnar output, 1st column shows number of lines present in a file and 2nd itself represent the file name.

**-w option** prints the number of words present in a file. 1st column shows number of words present in a file and 2nd is the file name.

**-c option** displays count of bytes present in a file. 1st column shows number of bytes present in a file and 2nd is the file name.

**-m option** displays count of characters from a file.

**-L option** prints out the length of longest (numbers of characters) line in a file.

**awk:** Awk is a general-purpose scripting language designed for advanced text processing. It is mostly used as a reporting and analysis tool. awk is data-driven, which means that you define a set of actions to be performed against the input text. It takes the input data, transforms it, and send the result to standard output.

An awk action is enclosed in {} and consists of statements. Each statement specifies the operation to be performed. An action can have more than one statement separated by newline or semi-colons.

**wc -l file.txt | awk '{print ((($1/4)/genome\_size)\*read\_length)}' (**This command firstly takes the total number of lines present in the file.txt and pipe it to the awk command. This command then performs a certain task on to the output of the wc command. The task is to divide the output of wc by 4, multiply it by read length and divide it by genome size and lastly print it.

**Zcat:** It is a command line utility for viewing the contents of a compressed file without literally uncompressing it. It expands a compressed file to standard output allowing us to have a look at its contents.

**Zgrep:** This command is used to search out expressions from a given a file even if it is compressed.

**-c option** is used to display the number of matching lines for each file.

­**-i option** is used to ignore case sensitivity.

**-n option** is used to display the line number of file if the given expression is present in the line.

**-v option** is used to display the lines which does not have the expression present in it.

**-e option** is used to specify the expression but can be used multiple times.

**-o option** is used to display only the matched section of the line from the given expression.

**-w option**  only displays lines only if the whole expression is found.

**-h option** is used to display the matched lines but doesn’t display the file names.

**Time command:** The time command is used to determine how long a given command takes to run. It is useful for testing the performance of your scripts and commands.

$ time command; example: $ time wget <https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.9.tar.xz>. We are going to measure the time taken to download the linux kernel using wget tool.

Output:

real 0m33.961s

user 0m0.340s

sys 0m0.940s

* **real** or **total** or **elapsed** (wall clock time) is the time from start to finish of the call. It is the time from the moment you hit the Enter key until the moment the wget command is completed.
* **user** - amount of CPU time spent in user mode.
* **system** or **sys** - amount of CPU time spent in kernel mode.

**Options: -f, --format** - Use this option to specify the format of output as you wish. Example: **-****f "\t%E real,\t%U user,\t%S sys" this will display the real, user and system time only.**

* **-p, --portability** - Use the portable output format.
* **-o file, --output=FILE** - Writes the output to **FILE** instead of displaying as standard output.
* **-a, --append** - Append the output to the FILE instead of overwriting it.
* **-v, --verbose** - This option displays the detailed description of the output of the 'time' utility.
* **--quiet** - This option prevents the time 'time' utility to report the status of the program.

Example: time -f "\t%E real,\t%U user,\t%S sys" -o /mnt/c/users/Hp/Desktop/file\_name.txt command

This will give the time utilized by the command written into the text file.

**Watch command:** This command will run the specified command in the argument repeatedly by showing its output and errors.

**Options:**

**-d, –differences:**This option highlights the differences between successive updates. The options will be going to read the optional argument which changes highlight to be permanent, allowing the user to see what has changed at least once since the first iteration.

**-n, –interval seconds:** This option will specify update interval. The command will not be going to allow quicker than the 0.1-second interval, in which the smaller values are getting converted.

**Top command:**  Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

* **PID:** Shows task’s unique process id.
* **PR:**Stands for priority of the task.
* **SHR:** Represents the amount of shared memory used by a task.
* **VIRT:** Total virtual memory used by the task.
* **USER:** Username of owner of task.
* **%CPU:**Represents the CPU usage.
* **TIME+:** CPU Time, the same as ‘TIME’, but reflecting more granularity through hundredths of a second.
* **SHR:** Represents the Shared Memory size (kb) used by a task.
* **NI:** Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
* **%MEM:**Shows the Memory usage of task.

**Batch Mode :**Send output from top to file or any other programs.

**top -b**

**top -n 1**: Top command will automatically exit after 2 number of repetitions.

**Important: watch -n 30 'top -b -n 1 | grep -i program >> memory\_usage.txt'** ; where the program is your assembler/aligner. (**To calculate memory used by the program**).

**ps command:** This command enables us to check the status of active processes on a system, as well as display technical information about the processes.

Depending on which options you use, the ps command reports the following information:

* Current status of the process
* Process ID
* Parent process ID
* User ID
* Scheduling class
* Priority
* Address of the process
* Memory used
* CPU time used

| **Field** | **Description** |
| --- | --- |
| UID | The effective user ID of the process's owner. |
| PID | The process ID. |
| PPID | The parent process ID. |
| C | The processor xutilization for scheduling. This field is not displayed when the **-c** option is used. |
| CLS | The scheduling class to which the process belongs such as real-time, system, or timesharing. This field is included only with the **-c** option. |
| PRI | The kernel thread's scheduling priority. Higher numbers indicate a higher priority. |
| NI | The process's nice number, which contributes to its scheduling priority. Making a process “nicer” means lowering its priority. |
| ADDR | The address of the proc structure. |
| SZ | The virtual address size of the process. |
| WCHAN | The address of an event or lock for which the process is sleeping. |
| STIME | The starting time of the process in hours, minutes, and seconds. |
| TTY | The terminal from which the process, or its parent, was started. A question mark indicates that there is no controlling terminal. |
| TIME | The total amount of CPU time used by the process since it began. |
| CMD | The command that generated the process. |

**Options:**

|  |  |
| --- | --- |
| a | Displays all processes on a terminal, with the exception of group leaders. |
| -c | Displays scheduler data. |
| -d | Displays all processes with the exception of session leaders. |
| -e | Displays all processes. |
| -f | Displays a full listing. |
| -g*list* | Displays data for the *list* of group leader IDs. |
| -j | Displays the process group ID and session ID. |
| -l | Displays a long listing |
| -p*list* | Displays data for the *list* of process IDs. |
| -s*list* | Displays data for the *list* of session leader IDs. |
| -t*list* | Displays data for the *list* of terminals. |
| -u*list* | Displays data for the *list* of usernames. |

**To kill a nohup command**: kill pid (of that command).

**To get the pid**: pidof command

free -th : **To know the ram**

**To calculate Elapsed Time of a process**: ps -p PID -o etime

**To make a directory readable and writeable by third party**: sudo chmode 777 directory\_name.

**Run scp in background**:

$ nohup scp file\_to\_copy user@server:/path/to/copy/the/file > nohup.out 2>&1

**Then press ctrl + z which will temporarily suspend the command, then enter the command:**

$ bg

**This will start executing the command in background.**