Experiment No.: 3

Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

1. pwd : print the working directory

$pwd

```
student@t2:~$ pwd
/home/student
student@t2:~$
```

a. ls -R : prints subdirectory contents

$ls -R

```
./jensample:
jenny

./jensample/jenny:
connect  months  names.txt
```

b. ls -l : long listing

$ls -l

```
-rw-rw-r-- 1 student student    18 Mar  6 12:26  week2.txt
-rw-rw-r-- 1 student student    33 Mar  6 12:27  weeks.txt
-rw-rw-r-- 1 student student    15 Mar  6 12:26  week.txt
```

c. ls -a : to list all hidden files

$ls -a

```
bubblesort.c              .gnupg                  .profile
bubblesort.o              'insertion sort.c'      Public
.cache                    .java                   PycharmProjects
```

d. ls -al : list the files and directory with detailed information

$ls -al

```
student@t2:~$ ls -al
total 524
drwxr-xr-x 23 student student  4096 Mar  7 15:19  .
drwxr-xr-x  6 root    root     4096 Jun 17  2022  ..
```

e. ls -t : list the file sorted in the order of the last modified file.

$ls -t

```
student@t2:~$ ls -t
 jensample                      linklist.c
 jenny                          linklis.c
 new.txt                        merge.c
 weeks.txt                     'insertion sort.c'
 week2.txt                      bubblesort
 week.txt                       bubblesort.o
 file2.txt                      bubblesort.c
 file1.txt                      merge
 Downloads                      merge.o
```

f. ls -r : to reverse the natural sorting order

$ls -r

```
student@t2:~$ ls -r
 week.txt          mergsort.c             'doubley link list creation.c'
 weeks.txt         mergesort.c             Documents
 week2.txt         merge.o                 Desktop
```

3. history : to review the commands that have been previously executed for certain period of time.

$history

```
student@t2:~$ history
    1  ./studio.sh
    2  ./studio.sh
    3  su mca
    4  ifconfg
    5  ifconfig
    6  su admin
    7  su mca
    8  sudo apt install :/code_1.75.11675893397_amd64.deb
    9  pwd
   10  ls
   11  ls-R
   12  ls-r
   13  ls -R
   14  ls -m
   15  ls -l
   16  ls -a
   17  ls -al
```

4. man : we can learn and understand different commands right from the shell.

$man ls

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..
```

5. mkdir : creates new directory

$mkdir jenny

```
student@t2:~$ man ls
student@t2:~$ mkdir Jenny
student@t2:~$ cd Jenny
student@t2:~/Jenny$ ls
student@t2:~/Jenny$
```

6. rmdir : to remove directory

$rmdir Jenny

```
student@t2:~$ rmdir Jenny
student@t2:~$ cd Jenny
bash: cd: Jenny: No such file or directory
```

7. touch : to create new file

$touch file.txt

```
student@t2:~$ touch file.txt
```

8. cat > [filename] : create a new file and open it to add content.

$cat > colors

```
student@t2:~$ cat > colors
red
blue
yellow
black
^Z
[3]+  Stopped                 cat > colors
```

a. cat >> [filename] : to append new contents to existing file contents

$cat >> colors

```
student@t2:~$ cat >> colors
green
white
```

b. cat [filename] : to display file contents.

$cat colors

```
student@t2:~$ cat colors
red
blue
yellow
black
green
white
```

c. cat -n [filename] : to display content with line numbers

$cat -n colors

```
student@t2:~$ cat -n colors
     1  red
     2  blue
     3  yellow
     4  black
     5  green
     6  white
```

d. cat -b [filename] : No line numbering for blank spacing.

$cat -b colors

```
student@t2:~$ cat -b colors
     1  red
     2  blue
     3  yellow
     4  black
     5  green
     6  white


     7  magenta
```

e. cat -e [filename] : to display $ character at the end of each line.

$cat -e colors

```
student@t2:~$ cat -e colors
red$
blue$
yellow$
black$
green$
white$
$
$
$
magenta$
```

f. cat [filename] << EOF : used as page end marker.

$cat colors << EOF

g. cat [filename] | tr a-z A-Z > [filename] : to convert lower case letters to upper case

h. cat -b : cut by byte position

i. cat -c: cut by character

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment No.: 4

Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

1. cut : used to cut file contents

a. cut -d - -f1 file2 : cut command to just print the first field of the file using the delimiter "-"

```
jenny@jenny-VirtualBox:~$ cut -d - -f1 file2
jenny
riya
tinu
```

2.paste:used to join files horizontally(Each file consisting of different lines)

a)paste file1 file2-To paste file1 contents in file2

```
jenny@jenny-VirtualBox:~$ paste file1 file2
jenny   jenny-3
riya    riya-29
tinu    tinu-45
alfiya
```

b)paste file1 file2 > file3-To paste file1 and file2 contents in a new file

```
jenny@jenny-VirtualBox:~$ paste file1 file2 > file3
jenny@jenny-VirtualBox:~$ cat file3
jenny   jenny-3
riya    riya-29
tinu    tinu-45
alfiya
```

c)paste –d '%' file1 file2- By specifying the delimiter, we can also split the lines into columns with specified delimiter.

```
jenny@jenny-VirtualBox:~$ paste -d '%d' file1 file2
jenny%jenny-3
riya%riya-29
tinu%tinu-45
alfiya%
```

3) cp - To copy the content to a new file

a)cp file1 file2-To copy file1 contents in file2

```
jenny@jenny-VirtualBox:~$ cp file1 file2
jenny@jenny-VirtualBox:~$ cat file2
jenny
riya
tinu
alfiya
```

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment No.: 5

Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

1. $read Screenshot from 2023-03-21 15-30-36Screenshot from 2023-03-21 15-30-36

To read the content of the line. Reads the line into a variable

$REPLY : to print the line

```
student@t2:~$ read
jenny
student@t2:~$ echo $REPLY
jenny
```

a. $read x y z

Declare variables to store data

To print : echo “[$x][$y][$z]”

```
student@t2:~$ read x y z
jenny johnson mca
student@t2:~$ echo "[$x][$y][$z]"
[jenny][johnson][mca]
```

b. To read contents through multiple lines we use”\” at the end of each line.

```
student@t2:~$ read
j\
> e\
> n\
> n\
> y
student@t2:~$ echo $REPLY
jenny
```

c. $read -p [prompt message]

Prompt user to enter data

```
student@t2:~$ read -p"Enter your name"
Enter your name Jenny
student@t2:~$ echo "My name is $REPLY"
My name is  Jenny
```

d. $read -n [imit]

Specifies the limit.

```
student@t2:~$ read -n 6 -p "Enter 6 characters: "
Enter 6 characters: jennyjstudent@t2:~$
```

e. $read -s :

it gives the security(hides the data)

```
student@t2:~$ read -s -p "Enter password: "
Enter password: student@t2:~$ echo "Password is $REPLY"
Password is 123456
```

2. $wc [filename]

Prints the no.of lines,words,bytes etc.

```
student@t2:~$ cat > Details
Jenny Johnson
RMCA
AmalJyothi College of Engineering
^Z
[2]+  Stopped                 cat > Details
student@t2:~$ wc Details
 3  7 53 Details
```

a. $wc -l [filename]

Prints no.of lines

```
student@t2:~$ wc -l Details
3 Details
```

b. $wc -m [filename]

Prints no.of characters

```
student@t2:~$ wc -m Details
53 Details
```

c. $wc -w [filename]

Prints no.of words

```
student@t2:~$ wc -w Details
7 Details
```

d. $wc -L [filename]

Prints the length of the longest line.

```
student@t2:~$ wc -L Details
33 Details
```

3. $more

The more command is similar to cat command to display content. The only difference is that, incase of large files cat command will scroll off your screen while more command display output one screen full at a time.

```
student@t2:~$ more corona.txt
```

```
The IBV-like novel cold viruses were soon shown to be also morphologically rela
ted to the mouse hepatitis virus.[19] This new group of viruses were named coro
naviruses after their distinctive morphological appearance.[7] Human coronaviru
s 229E and human coronavirus OC43 continued to be studied in subsequent decades
--More--(99%)
```

a. $more +20 [filename]

```
student@t2:~$ more +20 corona.txt


Transmission electron micrograph of organ cultured coronavirus OC43
Scottish virologist June Almeida at St Thomas' Hospital in London, collaboratin
g with Tyrrell, compared the structures of IBV, B814 and 229E in 1967.[32][33]
```

b. more +/[pattern] [filename]

Used to search the string inside your text document you can view all the instances navigating through the result.

```
student@t2:~$ more +/Human corona.txt

...skipping
as not realized at the time that these three different viruses were related.[20
][12]

Human coronaviruses were discovered in the 1960s[21][22] using two different me
thods in the United Kingdom and the United States.[23] E.C. Kendall, Malcolm By
```

c. $more -d[filename]

It helps the user to navigate according to instructions.

```
student@t2:~$ more -d corona.txt
```

```
d successfully cultivated rhinoviruses, adenoviruses and other known common col
d viruses. In 1965, Tyrrell and Bynoe successfully cultivated the novel virus b
--More--(4%)[Press space to continue, 'q' to quit.]
```

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment No.: 6

Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

$grep -i [word][filename]

Used to filter the content which makes our search easier(Case insensitive)

$ grep -i Hindi mark

```
student@t2:~/Jenny$ cat > mark
English 99
Maths 56
Hindi 78
Malayalam 89
```

```
student@t2:~/Jenny$ grep -i Hindi mark
Hindi 78
```

$grep -v[word][filename]

To view all the contents expect the searched one

$grep -v Hindi mark

```
student@t2:~/Jenny$ grep -v Hindi mark
English 99
Maths 56
Malayalam 89
```

$grep -A1[word][filename]

To view the content along with one line before that

$grep -A1 Hindi mark

```
student@t2:~/Jenny$ grep -A1 Hindi mark
Hindi 78
Malayalam 89
```

$grep -B1[word][filename]

To view the content along with one line before that.

$grep -B1 Hindi mark

```
student@t2:~/Jenny$ grep -B1 Hindi mark
Maths 56
Hindi 78
```

$grep -C1[word][filename]

To view the content along with one line after and before.

$grep -C1 Hindi mark

```
student@t2:~/Jenny$ grep -C1 Hindi mark
Maths 56
Hindi 78
Malayalam 89
```

$ head [filename]

To display the top lines of the file. By default it will display top 10 lines

$ head txt

$ head -5[filename]

Display top 5 lines

$head -5 txt

```
student@t2:~/Jenny$ head txt
1
2
3
4
5
6
7
8
9
11
student@t2:~/Jenny$ head -5 txt
1
2
3
4
5
```

$tail [filename]

To display the last contents of the file, by default it will display the last 10 lines.

$tail txt

$tail -5[filename]

To display the last five contents of the file.

$tail -5 txt

```
student@t2:~/Jenny$ tail txt
9
11
22
33
44
55
66
77
88
99
student@t2:~/Jenny$ tail -5 txt
55
66
77
88
99
```

$mv [filename][filename]

To move one file content to another file(contents will be overwritten).

$mv -b mark Bio : Backups file

```
student@t2:~/Jenny$ cat >Bio
Jenny John
RMCA
^Z
[4]+  Stopped                 cat > Bio
student@t2:~/Jenny$ mv -b mark Bio
student@t2:~/Jenny$ ls
Bio  Bio~  marks
student@t2:~/Jenny$ cat Bio~
Jenny John
RMCA
```

$mv -i [filename][filename]

prompt user for confirmation

$mv -i Bio Bio1

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment No.: 7

Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

1. $expr [expressions]

Calculate the expressions and return the output.

```
student@t2:~/jenny$ expr 45 + 9
54
student@t2:~/jenny$ expr 45 - 9
36
student@t2:~/jenny$ expr 45 \* 9
405
student@t2:~/jenny$ expr 45 / 9
5
```

2. $df

used to find report on disc utilisation

```
student@t2:~/jenny$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             3953504        0   3953504   0% /dev
tmpfs             797752     1736    796016   1% /run
/dev/sda6      143074460 28142036 107591832  21% /
tmpfs            3988756    32404   3956352   1% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            3988756        0   3988756   0% /sys/fs/cgroup
```

3. $du[filename]

Used to check how much space a file or directory takes in the current directory.

$du names.txt

```
student@t2:~/jenny$ ls
connect  months  names.txt
student@t2:~/jenny$ du names.txt
4       names.txt
student@t2:~/jenny$
```

4. sudo :- superuser do

a) $sudo useradd[username]: Add new user

```
mca@t2:~$ sudo useradd jenny
[sudo] password for mca:
mca@t2:~$ sudo useradd jenny
useradd: user 'jenny' already exists
```

b) sudo passwd user :- Update password of the user

$sudo passwd jenny

```
mca@t2:~$ sudo passwd jenny
New password:
Retype new password:
passwd: password updated successfully
```

c) sudo groupadd -g [identifier name]:- To create new group

$sudo groupadd -g 77 mcastudent

d) sudo usermod -G [name][user] :- Add users to group

$sudo usermod -G mcastudent jenny

e) id [user] :- Details on group name and numeric id of particular user.

$id jenny

```
mca@t2:~$ sudo groupadd -g 77 mcastudent
mca@t2:~$ sudo usermod -G mcastudent jenny
mca@t2:~$ id jenny
uid=1014(jenny) gid=1020(jenny) groups=1020(jenny),77(mcastudent)
```

5. $compgen -g: Display all the groups created

```
mca@t2:~$ compgen -g
root
daemon
bin
sys
adm
tty
disk
lp
mail
news
```

6. $chmod

Used to change the access permission of files and directories. It stands for change mode.

a) chmod -wx [filename] :- deny permission to write and execute for file

$chmod -wx names

```
mca@t2:~$ chmod -wx names
mca@t2:~$ cat >> names
bash: names: Permission denied
mca@t2:~$ chmod +rws names
mca@t2:~$ cat >> names
jomol
^Z
[2]+  Stopped                 cat >> names
mca@t2:~$
```

7. $chown

It is used to change a file ownership or directory ownership for a user or a group. It stands for change owner.

$sudo chown [directory][filename]

$chown jenny names

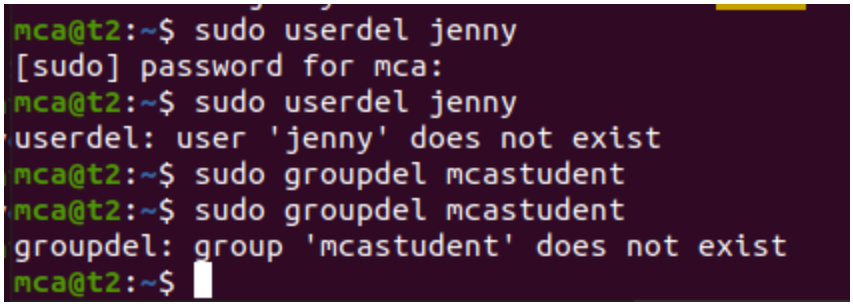
```
mca@t2:~$ sudo chown jenny names
[sudo] password for mca:
mca@t2:~$ chmod +rwx names
chmod: changing permissions of 'names': Operation not permitted
mca@t2:~$ ls -l names
-rw-rwSr-- 1 jenny mca 25 Mar 20 11:58 names
mca@t2:~$
```

8. $ sudo userdel user :- Delete user

$ sudo userdel jenny

9. $ sudo groupdel: deletes group

$ sudo group mcastudent



Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment No.: 8

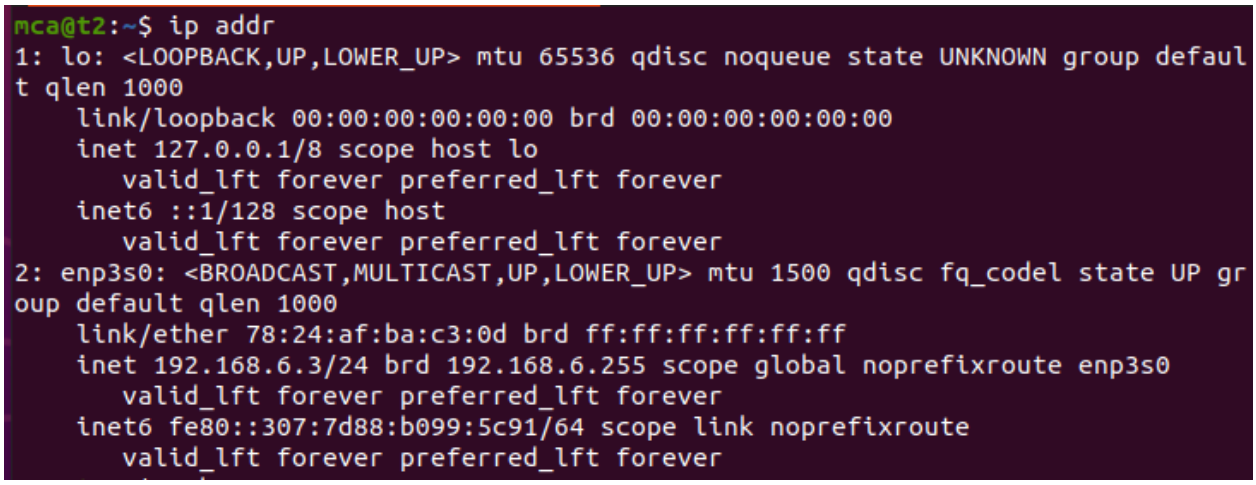
Aim: Familiarisation of linux commands.

CO2: Perform system administration task.

Procedure:

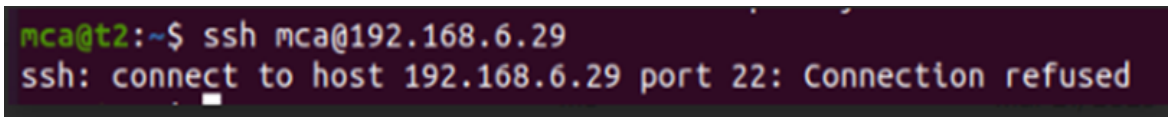
1)ip addr - Get ip address of the system

$ip addr



2)ssh user@ip address- Stands for Secure Shell Protocol used to securely connect to a remote server or system. ssh is secure in the sense that it transfers data in encrypted form between host and client.

$ssh mca@192.168.6.29



a. sudo apt-get install openssh -server :- Update port

```
mca@t2:~$ sudo apt-get update
[sudo] password for mca:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:3 https://dl.google.com/linux/chrome/deb stable InRelease [1,811 B]
Get:4 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal InRe
lease [17.6 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:6 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,079 B
]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2,04
6 kB]
Get:8 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal/main
 amd64 Packages [2,052 B]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:10 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal/mai
n Translation-en [324 B]
```

```
mca@t2:~$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-client openssh-sftp-server ssh-import-id
Suggested packages:
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
The following packages will be upgraded:
  openssh-client
1 upgraded, 4 newly installed, 0 to remove and 682 not upgraded.
Need to get 1,359 kB of archives.
After this operation, 6,010 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-clien
```

b. sudo ufw allow 22

```
mca@t2:~$ sudo ufw allow 22
Rules updated
Rules updated (v6)
```

c. $ssh mca@192.168.6.29

```
mca@t2:~$ ssh mca@192.168.6.29
The authenticity of host '192.168.6.29 (192.168.6.29)' can't be established.
ECDSA key fingerprint is SHA256:kKk7sOMYRkq6/HO6Go97XKajNqNDSThuvCv+GSJz40U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.6.29' (ECDSA) to the list of known hosts.
mca@192.168.6.29's password:
Permission denied, please try again.
mca@192.168.6.29's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


698 updates can be installed immediately.
459 of these updates are security updates.
To see these additional updates run: apt list --upgradable
```

d. ssh-keygen :- Generating a key for secure shell

$ssh-keygen

```
mca@t2:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mca/.ssh/id_rsa): jen.txt
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in jen.txt
Your public key has been saved in jen.txt.pub
The key fingerprint is:
SHA256:VY++5295UY4kWP5ceeeQ2JqhgKjQxsSiN/0q8CCK1vo mca@t2
The key's randomart image is:
+---[RSA 3072]----+
| .           .   |
|. o        ..o   |
|.= . . .  .+.o...|
|o * o . ....= =.=|
| + o .  S. ..O *+|
|+ .   .   . o.+.o|
|++.  .      . . o|
|o.o..        o .o|
|..oE           .oo|
+----[SHA256]-----+
```

```
mca@t2:~$ cat jen.txt
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA0qc3TMLgtbSNzLdKJmZ3qzDooL+cMiivZx0mCb5ndpLpUb40lwLX
zZ8GE4W3IuvCWOCby6WWkXljCfSNaN7Woy7hBIaVDjUUhAec3r8uequCWsFDp8oAI2QnY2
YMN/GIevXZHJ2MFOD5wRyRLxA2iGn9ruLZM9LzCzs6VXn4UR5iP0ubQzcORl/18Ekx9dRd
/G4dRUCeGFUC7POU7atXrtx2pt5fEF5aEAAADBAN/BzqEr4t8Zn489EDhEwIKcOEVSsBEE
wgq7WifXaIQnUyjMtOquRY5bGD76OLBCQXRFM+9gSjtvro8eh0MIjq+dAUIKJ24PyZ0sni
YY1O4z38s+m0tB8u7xpujy6WPPlyhL/jp0M1ZBqck5iToYi/rKN05hWLbZ5gvWOZxmFBJD
niZWuA9CZ+Tj86NAtD1zkFHQA0cOwTEC0iFkKkpDc45LDJIbOPzUFoGXbVG8Aq7SrrH1/p
C0Rzw6CsVOQNJ6jwAAAAZtY2FAdDIBAgM=
-----END OPENSSH PRIVATE KEY-----
mca@t2:~$
```

3) ps - Stands for Process. Currently running programs and running instances.

$ps

a)ps –u [user] :- Display all running processes of a particular user

$ps -u mca

```
mca@t2:~$ ps
    PID TTY          TIME CMD
   7220 pts/1    00:00:00 bash
   9000 pts/1    00:00:00 ps
mca@t2:~$ ps -u mca
    PID TTY          TIME CMD
   1364 ?        00:00:00 systemd
   1365 ?        00:00:00 (sd-pam)
   1371 ?        00:00:00 pulseaudio
   1373 ?        00:00:00 tracker-miner-f
   1376 ?        00:00:00 dbus-daemon
   1381 ?        00:00:00 gnome-keyring-d
   1384 ?        00:00:00 gvfsd
```

b)ps -C :- Specific process

$ps -C firefox

```
mca@t2:~$ ps -C firefox
    PID TTY          TIME CMD
   9047 ?        00:00:01 firefox
mca@t2:~$
```

c)ps -f –p PID :- List the process by id

$ps -f -p 9047

```
mca@t2:~$ ps -f -p 9047
UID          PID    PPID  C STIME TTY          TIME CMD
mca         9047    1364 10 15:58 ?        00:00:10 /usr/lib/firefox/firefox -ne
mca@t2:~$
```

Result

The program was executed and the result was successfully obtained. Thus CO2 was obtained.

Experiment: 9 28-03-2023

Aim : Shell scripting

CO4: Write shell scripts required for system administration

Procedure

1. Shell script to display date:

```
#!/bin/bash
date
:wq!
~
~
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg1_date.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg1_date.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg1_date.sh
Monday 03 April 2023 07:16:52 AM IST
jenny@jenny-VirtualBox:~/Jenny$
```

2. Shell script to display your name

```
#!/bin/bash/
echo "What is your name?"
read name
echo"My name is $name"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg2_name.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg2_name.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg2_name.sh
 What is your name?
jenny
my name is jenny
```

3. Multiple Commands (ls, pwd, date) in Shell Script:

```
#!/bin/bash
date
ls
pwd
ls
```

```
jenny@jenny-VirtualBox:~$ vi muitcmd.sh
jenny@jenny-VirtualBox:~$ chmod +x muitcmd.sh
jenny@jenny-VirtualBox:~$ ./muitcmd.sh
Monday 17 April 2023 02:20:32 AM IST
Desktop    file1  first.sh  muitcmd.sh  para.sh   second.sh  third.sh
Documents  file2  jenny     Music       Pictures  snap       Videos
Downloads  file3  Jenny     parag.sh    Public    Templates
/home/jenny
Desktop    file1  first.sh  muitcmd.sh  para.sh   second.sh  third.sh
Documents  file2  jenny     Music       Pictures  snap       Videos
Downloads  file3  Jenny     parag.sh    Public    Templates
```

4. Shell script to demonstrate variables

```
#!/bin/bash
echo "File name:$0"
echo "First parameter:$1"
echo "Total parameter:$#"
echo "Second parameter: $2"
echo "Quoted values:$@"
echo "Quoted values:$*"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg3_var.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg3_var.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg3_var.sh Jenny Johnson
File name:./prg3_var.sh
First parameter:Jenny
Total parameter:2
Second parameter: Johnson
Quoted values:Jenny Johnson
Quoted values:Jenny Johnson
```

5. Shell script to count lines and words in a file

```
#!/bin/bash
file_path=/home/jenny/Jenny/prg3_var.sh
countlines=`wc --lines < $file_path`
countwords=`wc --word < $file_path`
echo "Number of lines: $countlines"
echo "Number of words: $countwords"
~
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg4_countl.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg4_countl.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg4_countl.sh
Number of lines: 8
Number of words: 20
```

6. Shell script to display array index

```
#!/bin/bash
Name[0]="Jenny"
Name[1]="Riya"
Name[2]="Sara"
Name[3]="Tinu"
Name[4]="Alfiya"
echo "First index: ${Name[0]}"
echo "Last index: ${Name[4]}"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg5_array.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg5_array.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg5_array.sh
First index: Jenny
Last index: Alfiya
```

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained.

Experiment: 10 **03-04-2023**

Aim : Shell scripting

CO4: Write shell scripts required for system administration

Procedure

1. Shell script to add two number:

```
#!/bin/bash
num=`expr 3 + 7`
echo "Total= $num"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg6_add.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg6_add.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg6_add.sh
Total= 10
```

2. Write a shell script to initialize two numeric variables. Then perform addition operation on both values and store the result in the third variable.

```
#!/bin/bash
n1=11
n2=22
n=`expr $n1 + $n2`
echo "Total :$n"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg6_add2.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg6_add2.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg6_add2.sh
Total :33
```

3. Shell script to read two numbers as command line parameters and perform the addition operation

```
#!/bin/bash
sum=$(( $1 + $2))
echo "Sum= $sum"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg6_add3.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg6_add3.sh
Sum= 3
```

4. Shell script which takes input from the user at run time and then calculate the sum of given number and store to a variable and show the result

```
#!/bin/bash
echo "Enter first number:"
read a
echo "Enter second number:"
read b
c=`expr $a + $b`
echo "Sum= $c"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg6_add4.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg6_add4.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg6_add4.sh
Enter first number:
3
Enter second number:
9
Sum= 12
```

5. Shell script to demonstrate Arithmetic operators (addition, subtraction, multiplication, division, modulus, increment, decrement) by taking user input and store to another variable

```
#!/bin/bash
read -p "Enter first number: " a
read -p "Enter second number: " b
add=$(( a + b))
echo "Sum of $a and $b is $add"
sub=$(( a - b))
echo "$a minus $b is $sub"
mul=$(( a * b ))
echo "Product of $a and $b is $mul"
div=$((a / b))
echo "$a divided by $b is $div"
mod=$((a % b))
echo "$a modulo $b is $mod"
if [ $a == $b ]
then
      echo "$a and $b are equal"
fi
if [ $a != $b ]
then
echo "$a and $b are not equal"
fi
(( ++a ))
echo "After incrementing the value of first number: $a"
(( --b ))
echo "After decrementing the value of second number: $b"
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg8_arith.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg8_arith.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg8_arith.sh
Enter first number: 4
Enter second number: 2
Sum of 4 and 2 is 6
4 minus 2 is 2
Product of 4 and 2 is 8
4 divided by 2 is 2
4 modulo 2 is 0
4 and 2 are not equal
After incrementing the value of first number: 5
After decrementing the value of second number: 1
```

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained.

Experiment: 11 04-04-2023

Aim : Shell scripting

CO4: Write shell scripts required for system administration

Procedure

1. Shell script to demonstrate Relational operators (equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to) by taking user input

```
#!/bin/bash

read -p "Enter first number: " a
read -p "Enter second number: " b
if(( $a == $b ))
then
echo "$a and $b are equal"
else
echo "$a and $b are not equal"
fi
if(( $a > $b ))
then
echo "$a is greater than $b"
else
echo "$a is not greater than $b"
fi
if(( $a < $b ))
then
echo "$a is less  than $b"
else
echo "$a is not less than $b"
fi
if(( $a >= $b ))
then
echo "$a is greater than or equal to $b"

else
echo "$a is not greater than or equal to $b"
fi
if(( $a <= $b ))
then
echo "$a is less than or  equal to $b"

else
echo "$a is not lesS than or equal to $b"
fi
if(( $a != $b ))
then
echo "$a and $b are not equal"
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg9_operators.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg9_operators.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg9_operators.sh
Enter first number: 3
Enter second number: 2
3 and 2 are not equal
3 is greater than 2
3 is not less than 2
3 is greater than or equal to 2
3 is not lesS than or equal to 2
3 and 2 are not equal
```

2. Shell script to demonstrate Relational operators (equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to)

```
#!/bin/bash

read -p "Enter first number: " a
read -p "Enter second number: " b
if [ $a -eq  $b ]
then
echo "$a and $b are equal"
else
echo "$a and $b are not equal"
fi
if [ $a -gt $b ]
then
echo "$a is greater than $b"
else
echo "$a is not greater than $b"
fi
if [ $a -lt $b ]
then
echo "$a is less  than $b"
else
echo "$a is not less than $b"
fi
if [ $a -ge $b ]
then
echo "$a is greater than or equal to $b"
else
echo "$a is not greater than or equal to $b"
fi
if [ $a -le $b ]
then
echo "$a is less than or  equal to $b"
else
echo "$a is not lesS than or equal to $b"
fi
if [ $a -ne  $b ]
then
echo "$a and $b are not equal"
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi prg10_operators.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x prg10_operators.sh
jenny@jenny-VirtualBox:~/Jenny$ ./prg10_operators.sh
Enter first number: 2
Enter second number: 3
2 and 3 are not equal
2 is not greater than 3
2 is less  than 3
2 is not greater than or equal to 3
2 is less than or  equal to 3
2 and 3 are not equal
```

3. Shell script to demonstrate Logical operators (AND, OR, NOT) by taking user input

```
#!/bin/bash
read -p "Enter first number: " a
read -p "Enter second number: " b
if(($a == "true" & $b == "true"))
then
echo Both are true
else
echo Both are false
fi
if(($a == "true" || $b == "true"))
then
echo Atleast one of them is true
else
echo None of them is true
fi
if(( ! $a == "true" ))
then
echo "a" was initially false
else
echo "a" was initially true
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ ./prg11_logical.sh
Enter first number: 2
Enter second number: 3
Both are false
None of them is true
a was initially false
```

4. Write a shell script to check if a number is even or odd.

```
#!/bin/bash
read -p "Enter the number:" n
if(($n % 2 == 0 ))
then
echo Number is even
else
echo Number is odd
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi even_odd.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x even_odd.sh
jenny@jenny-VirtualBox:~/Jenny$ ./even_odd.sh
Enter the number:7
Number is odd
jenny@jenny-VirtualBox:~/Jenny$ ./even_odd.sh
Enter the number:2
Number is even
```

5. Write a shell script to check whether a number is positive or negative

```
#!/bin/bash
read -p "Enter the number: " n
if(( $n < 0))
then
echo Number is negative
elif(($n > 0))
then
echo Number is positive
elif(($n == 0))
then
echo Number is zero
else
echo Invalid
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi num_check.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x num_check.sh
jenny@jenny-VirtualBox:~/Jenny$ ./num_check.sh
Enter the number: 7
Number is positive
jenny@jenny-VirtualBox:~/Jenny$ ./num_check.sh
Enter the number: -1
Number is negative
jenny@jenny-VirtualBox:~/Jenny$ ./num_check.sh
Enter the number: 0
Number is zero
```

6. Write a shell script to find the greatest of two numbers

```
#!/bin/bash
read -p "Enter first number: " a
read -p "Enter second number: " b
if(($a < $b))
then
echo $b is greatest
else
echo $a is greatest
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ ./greatest_2.sh
Enter first number: 7
Enter second number: 3
7 is greatest
```

7. Write a shell script to find the greatest of two numbers

```
student@t2:~/Jenny$ ./largest3.sh
Enter the first number
2
Enter second number
3
Enter third number
1
3 is largest
```

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained.

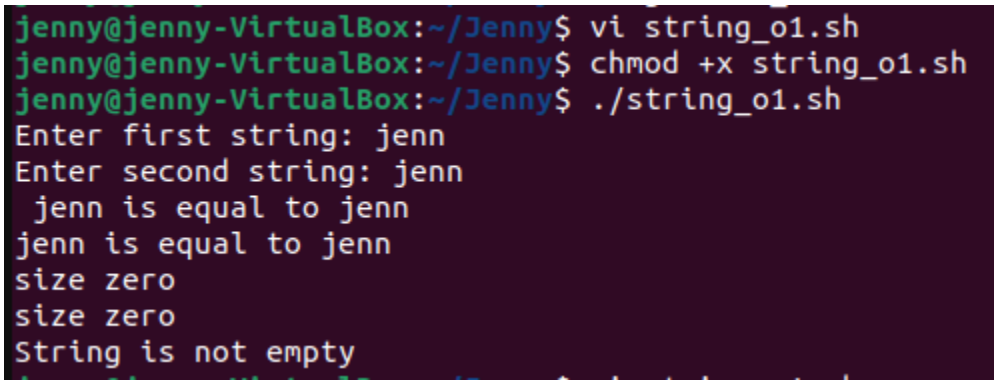
Experiment: 12 **11-04-2023**

Aim : Shell scripting

CO4: Write shell scripts required for system administration

Procedure

1. Shell script to demonstrate String operators (Equal, Not Equals, Size is zero, Size is non-zero, Empty string) by taking user input



```
#!/bin/bash
read -p "Enter first string: " s1
read -p "Enter second string: " s2

if [ $s1 = $s2 ]
then
    echo " $s1 is equal to $s2"
else
    echo "$s1 is not equal to $s2"
fi

if [ $s1 != $s2 ]
then
    echo "$s1 is not equal to $s2"
else
    echo "$s1 is equal to $s2"
fi

if [ -z $s1 ]
then
    echo "Size not zero"
else
    echo "size zero"
fi

if [ -n $s2 ]
then
    echo "size zero"
else
    echo "size not zero"
fi

if [ $s1 ]
then
   echo "String is not empty"
else
   echo "String is empty"
fi
```

2. Shell script to demonstrate Bitwise operators (AND, OR, XOR, Complement, Right Shift, Left Shift) by taking user input

```
jenny@jenny-VirtualBox:~/Jenny$ vi bitwise.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x bitwise.sh
jenny@jenny-VirtualBox:~/Jenny$ ./bitwise.sh
Enter value: 10
Enter value: 8
Bitwise AND of 10 and 8 is 8
Bitwise OR of 10 and 8 is 10
Bitwise XOR of 10 and 8 is 2
compliment of 10 is -11
Left shift of 10 is 20
 Right shift of 10 is 5
Left shift of 8 is 16
 Right shift of 8 is 4
```

```
#!/bin/bash
read -p "Enter value: " n
read -p "Enter value: " m

band=$(( n&m ))
   echo "Bitwise AND of $n and $m is $band"
bor=$(( n|m ))
   echo "Bitwise OR of $n and $m is $bor"
bxor=$(( n^m ))
   echo "Bitwise XOR of $n and $m is $bxor"
bcom=$(( ~n ))
  echo "compliment of $n is $bcom"
ls=$(( n<<1 ))
  echo "Left shift of $n is $ls"
rs=$(( n>>1 ))
  echo " Right shift of $n is $rs"
ls=$(( m<<1 ))
  echo "Left shift of $m is $ls"
rs=$(( m>>1 ))
  echo " Right shift of $m is $rs"
```

3. Shell script to demonstrate File Test operators (Exist(e), Size(s), Read Permission(r), Execute Permission(x), Write Permission(w)) by taking user input

```
student@t2:~/Jenny$ ./file_test.sh
Enter filename: string_2.sh
 file exist
the given file is not empty
the file has read access
The file has write access
the file has execute access
student@t2:~/Jenny$ ./file_test.sh
Enter filename: txt
file does not exsit
the fle s empty
the file doesnot have read access
the file doesnot have write access
the file does not has execute access
```

```
#!/bin/bash
read -p "Enter filename: " f
if [ -e $f ]
then
    echo " file exist"
else
    echo "file does not exsit"
fi

if [ -s $f ]
then
    echo "the given file is not empty"
else
    echo "the fle s empty"
fi

if [ -r $f ]
then
    echo "the file has read access"
else
    echo "the file doesnot have read access"
fi

if [ -w $f ]
then
    echo "The file has write access"
else
    echo "the file doesnot have write access"
fi

if [ -x $f ]
then
   echo "the file has execute access"
else
   echo "the file does not has execute access"
fi
```

4. Shell Script to check if two numbers are equal using if statement

```
jenny@jenny-VirtualBox:~/Jenny$ vi if_else.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x if_else.sh
jenny@jenny-VirtualBox:~/Jenny$ ./if_else.sh
Enter the first number: 3
Enter the second number: 3
Both numbers are equal
```

```
#!/bin/bash
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
if(( $num1==$num2 ))
then
                echo "Both numbers are equal"
fi
if(( $num1!=$num2 ))
then
                echo "Both numbers are not equal"
fi
```

5. Shell Script to check the range of a number if numbers using else if ladder

```
jenny@jenny-VirtualBox:~/Jenny$ vi range.sh
jenny@jenny-VirtualBox:~/Jenny$ chmod +x range.sh
jenny@jenny-VirtualBox:~/Jenny$ ./range.sh
Enter the number: 13
Range of 13 is between 11 and 20
```

```
#!/bin/bash
read -p "Enter the number: " n
if(( $n>=0 && $n<=10))
then
   echo Range of $n is between 0 and 10
elif(( $n>=11 && $n<=20))
then
   echo Range of $n is between 11 and 20
elif(( $n>21 && $n<=30))
then
   echo Range of $n is between 21 and 30
else
   echo Range is above 31
fi
```

6. Shell Script to display the grade of a student by accepting his mark.

```
#!/bin/bash
read -p "Enter the mark: " n
if(( $n>=0 && $n<=40))
then
   echo Grade is E
elif(( $n>=41 && $n<=55))
then
   echo Grade is C
elif(( $n>=56 && $n<=80))
then
   echo Grade is B
else
   echo  Grade is A
fi
```

```
jenny@jenny-VirtualBox:~/Jenny$ vi grade.sh
jenny@jenny-VirtualBox:~/Jenny$ ./grade.sh
Enter the mark: 90
Grade is A
```

Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained.