Experiment No.: 3

Aim

Familiarisation of Linux Commands

CO2

Perform System administration tasks

Procedure

1)pwd-The path of the current working directory

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

2)ls- To view the contents of the directory

```
mca@t2:~$ ls
c.txt  Desktop  Documents  Downloads  file1  Music  output.txt  Pictures  profile  Public  Templates  tinu
```

a)ls –R – To view list of all files in subdirectory

```
mca@t2:~$ ls -R
.:
c.txt  Desktop  Documents  Downloads  file1  Music  output.txt

./Desktop:
tinu
```

b)ls –l – Long listing of the contents

```
mca@t2:~$ ls -l
total 56
-rw-rw-r-- 1 mca mca   16 Mar  7 16:42 c.txt
drwxr-xr-x 3 mca mca 4096 Mar  7 16:56 Desktop
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Documents
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Downloads
-rw-rw-r-- 1 mca mca   41 Mar  7 16:44 file1
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Music
-rw-rw-r-- 1 mca mca   41 Mar  7 16:46 output.txt
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Pictures
-rw-rw-r-- 1 mca mca   83 Mar 13 11:33 profile
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Public
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Templates
-rw-rw-r-- 1 mca mca   43 Mar  7 16:32 tinu
-rw-rw-r-- 1 mca mca   15 Mar  7 16:37 tinusample
drwxr-xr-x 2 mca mca 4096 Jun 17  2022 Videos
```

c)ls –a – To list all the hidden files

```
mca@t2:~$ ls -a
.                .bash_logout   .config   Documents
..               .bashrc        c.txt     Downloads
.bash_history    .cache         Desktop   file1
```

d)ls –al- List the files and directories with detailed information such as owner, file size, permission etc.

```
mca@t2:~$ ls -al
total 104
drwxr-xr-x 16 mca  mca  4096 Mar 13 11:32 .
drwxr-xr-x  6 root root 4096 Jun 17  2022 ..
-rw-------  1 mca  mca   716 Mar 13 11:05 .bash_history
-rw-r--r--  1 mca  mca   220 Jun 17  2022 .bash_logout
-rw-r--r--  1 mca  mca  3771 Jun 17  2022 .bashrc
drwxr-xr-x 14 mca  mca  4096 Mar  7 16:48 .cache
drwxr-xr-x 13 mca  mca  4096 Mar 13 11:16 .config
-rw-rw-r--  1 mca  mca    16 Mar  7 16:42 c.txt
drwxr-xr-x  3 mca  mca  4096 Mar  7 16:56 Desktop
drwxr-xr-x  2 mca  mca  4096 Jun 17  2022 Documents
drwxr-xr-x  2 mca  mca  4096 Jun 17  2022 Downloads
-rw-rw-r--  1 mca  mca    41 Mar  7 16:44 file1
drwx------  3 mca  mca  4096 Mar 13 10:55 .gnupg
drwxr-xr-x  3 mca  mca  4096 Jun 17  2022 .local
drwxr-xr-x  2 mca  mca  4096 Jun 17  2022 Music
-rw-rw-r--  1 mca  mca    41 Mar  7 16:46 output.txt
drwxr-xr-x  2 mca  mca  4096 Jun 17  2022 Pictures
drwx------  3 mca  mca  4096 Mar  7 16:05 .pki
```

e)ls –t – List the files in the order of last modified

```
mca@t2:~$ ls -t
Desktop    file1  tinusample  Documents  Music    Public    Videos
output.txt c.txt  tinu        Downloads  Pictures Templates
```

f)ls –r – To reverse in natural sorting order

```
mca@t2:~$ ls -r
Videos     tinu      Public    output.txt  file1     Documents  c.txt
tinusample Templates Pictures  Music       Downloads Desktop
```

3) history – To review the commands that have been previously executed for certain period of time

```
mca@t2:~$ history
    1  ./studio.sh
    2  sudo ./studio.sh
    3  ./ studio
    4  ./ studio.sh
    5  ./studio.sh
    6  sudo ./studio.sh
    7  javac
    8  sudo apt-get install openjdk-11-jdk
    9  sudo ./studio.sh
   10  su student
   11  ls
```

4)man – You can learn and understand about different commands, write from the shell using man command

```
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 specified.

       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 – To create a new directory

```
mca@t2:~$ mkdir new
mca@t2:~$ cd new
mca@t2:~/new$ ls
mca@t2:~/new$ cd ..
mca@t2:~$ ls
Desktop  Documents  Downloads  Music  new  Pictures  Public  Templates  Videos
```

6)cd –Used to change the directory to previous directory

```
mca@t2:~$ cd
mca@t2:~$ cd new
mca@t2:~/new$ cd ..
```

7)rmdir - To remove the empty directory

```
mca@t2:~$ rmdir new
mca@t2:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

8)touch – To create a new empty file

```
mca@t2:~$ touch tinu
mca@t2:~$ man touch
mca@t2:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  tinu  Videos
```

9)cat – To view, create, concatenate files

a)cat > file1.txt – To add contents

```
mca@t2:~$ cat > tinu
TINU CLARA EMMANUEL
AJCE
MCA
^Z
[1]+  Stopped                 cat > tinu
```

b)cat file1.txt – To view

```
mca@t2:~$ cat tinu
TINU CLARA EMMANUEL
AJCE
MCA
```

c)cat >> file1.txt – To append the contents

```
mca@t2:~$ cat >> tinu
kanjirappally
^Z
[2]+  Stopped                 cat >> tinu
mca@t2:~$ cat tinu
TINU CLARA EMMANUEL
AJCE
MCA
kanjirappally
```

d)cat file1.txt file2.txt > file3.txt – To store the contents of the two files to another file

```
mca@t2:~$ cat tinu
TINU CLARA EMMANUEL
AJCE
MCA
kanjirappally
```

```
mca@t2:~$ cat > tinusample
minu
anu
sebin
^Z
[3]+  Stopped                 cat > tinusample
mca@t2:~$ cat tinu tinusample > c.txt
mca@t2:~$ cat c.txt
TINU CLARA EMMANUEL
AJCE
MCA
kanjirappally
minu
anu
sebin
```

e)cat -n file3.txt – To display the contents with line numbers

```
mca@t2:~$ cat -n c.txt
     1  TINU CLARA EMMANUEL
     2  AJCE
     3  MCA
     4  kanjirappally
     5  minu
     6  anu
     7  sebin
```

f) cat -b file4.txt – To remove the empty line number

```
mca@t2:~$ cat file1
anu
minu
tinu
sebin
happy family



time
mca@t2:~$ cat -b file1
     1  anu
     2  minu
     3  tinu
     4  sebin
     5  happy family



     6  time
```

g)cat file1.txt | tr a-z A-Z > output.txt – To change the contents to capital letters

```
mca@t2:~$ cat file1|tr a-z A-Z > output.txt
mca@t2:~$ cat output.txt
ANU
MINU
TINU
SEBIN
HAPPY FAMILY
```

<u>Result</u>

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 tasks

Procedure

1)cut-For cutting out sections from each line of files and writing the result to standard output

a)$cut -b1 filename-cut by byte position

```
ubuntu@Tinu:~/newfile$ cat > fil1
colours-green,red,black,blue
numbers-3,24,29,45
```

```
ubuntu@Tinu:~/newfile$ cut -b1 fil1
c
n
```

b)$cut -c3 filename-cut by character position

```
ubuntu@Tinu:~/newfile$ cut -c3 fil1
l
m
```

c)cut -d - -f1 filename: cut command to just print the first field of the file using the delimiter "-"

```
ubuntu@Tinu:~/newfile$ cut -d - -f1 fil1
colours
numbers
```

```
ubuntu@Tinu:~/newfile$ cut -d - -f2 fil1
green,red,black,blue
3,24,29,45
```

d)cut –c 1,4,6 filename – cut command to cut and print the specified character position

```
ubuntu@Tinu:~/newfile$ cut -c 1,4,6 fil1
cor
nbr
```

e)cut –d ‘ ‘–f filename - cut command to just print the first field of the file using the empty delimiter “ ”

```
ubuntu@Tinu:~/newfile$ cut -d ' ' -f1 fil1
colours-green,red,black,blue
numbers-3,24,29,45
```

2)paste- Paste command is used to join files horizontally(Each file consisting of different lines)

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

```
ubuntu@Tinu:~/newfile$ cat > fil1
colours-green,red,black,blue
numbers-3,24,29,45
^Z
[1]+  Stopped                 cat > fil1
```

```
ubuntu@Tinu:~/newfile$ cat > fil2
year-2021,2022,2023,2024
months-jan,feb,march,april
^Z
[2]+  Stopped                 cat > fil2
ubuntu@Tinu:~/newfile$ paste fil1 fil2
colours-green,red,black,blue    year-2021,2022,2023,2024
numbers-3,24,29,45      months-jan,feb,march,april
```

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

```
ubuntu@Tinu:~/newfile$ paste fil1 fil2 > fil3
ubuntu@Tinu:~/newfile$ cat fil3
colours-green,red,black,blue    year-2021,2022,2023,2024
numbers-3,24,29,45      months-jan,feb,march,april
```

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

```
ubuntu@Tinu:~/newfile$ paste -d "%" fil1 fil2
colours-green,red,black,blue%year-2021,2022,2023,2024
numbers-3,24,29,45%months-jan,feb,march,april
```

d)paste –s file1- Helps to display the contents in the file in a horizontal format

```
ubuntu@Tinu:~/newfile$ cat > fil1
colours-green,red,black,blue
numbers-3,24,29,45
^Z
[1]+  Stopped                 cat > fil1
ubuntu@Tinu:~/newfile$ paste -s fil1
colours-green,red,black,blue    numbers-3,24,29,45
```

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

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

```
ubuntu@Tinu:~/newfile$ cp fil1 fil2
ubuntu@Tinu:~/newfile$ cat fil2
colours-green,red,black,blue
numbers-3,24,29,45
```

b) $cp -r tinu tinu2-to copy the directory along with its sub directories

```
ubuntu@Tinu:~$ cp -r new newfile
ubuntu@Tinu:~$ cd newfile
ubuntu@Tinu:~/newfile$
```

<u>Result</u>

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

Experiment No.: 5 **DATE-13/03/2023**

Aim

Familiarisation of Linux Commands

CO2

Perform System Administration tasks

Procedure

1) read – Too read the content of the line, we use read command. This line read the command into a variable

a)read

```
mca@t2:~$ read
computer network and system administration
mca@t2:~$ echo $REPLY
```

b)read variable1 variable2 variable3 -Declare variables to store data

```
mca@t2:~$ read var1 var2 var3
computer networking and system administration
```

```
mca@t2:~$ echo "[$var1][$var2][$var3]"
[computer][networking][and system administration]
```

c) To read contents through multiple lines we use"\" at the end of each line.

```
mca@t2:~$ read
computer \
> networking and \
> system \
> administration
mca@t2:~$ echo $REPLY
computer networking and system administration
```

c)read -p "[Prompt message]" -Prompt user to enter data

```
mca@t2:~$ read -p "ENTER YOUR NAME"
ENTER YOUR NAME TINU
mca@t2:~$ echo "My name is $REPLY"
My name is  TINU
```

d)read -n limit -p - Specifies the limit

```
mca@t2:~$ read -n 7 -p "Enter 6 characters only"
Enter 6 characters only Emmanumca@t2:~$
```

e)read -s -p - it gives the security(hides the data)

```
mca@t2:~$ read -s -p "Enter the password"
Enter the passwordmca@t2:~$ echo "Password is $REPLY"
Password is 112325
```

2)wc -To display number of lines, number of words, number of bytes, filename

a)wc filename

```
mca@t2:~$ cat > profile
My name is TINU
Student of Amal Jyothi College of Engineering
Koovappally
Kottayam
^Z
[1]+  Stopped                 cat > profile
mca@t2:~$ wc profile
 4 13 83 profile
```

b) wc -l filename

```
mca@t2:~$ wc -l profile
4 profile
```

c) wc -m filename

```
mca@t2:~$ wc -m profile
83 profile
```

d) wc -c filename

```
mca@t2:~$ wc  -c profile
83 profile
```

e)wc -w filename

```
mca@t2:~$ wc -w profile
13 profile
```

f)wc -L filename – Print the length of the longest line

```
mca@t2:~$ wc -L profile
45 profile
```

3)more - The more command is similar to cat command to display the content. The only

difference is that in case of larger files cat command output will scroll of your screen while more command display output one screenful at a time.

a)more filename.txt

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

```
Tasks
Analyses, develops, interprets and evaluates complex system design and architecture specifications, data models and diagrams in the developmen
t, configuration and integration of computer systems.

Researches, analyses, evaluates and monitors network infrastructure to ensure networks are configured to operate at optimal performance.

Assesses and recommends improvements to network operations and integrated hardware, software, communications and operating systems.

Computer Network and Systems Engineers plan, develop, deploy, test and optimise network and system services, taking responsibility for configu
--More--(52%)
```

b)more +[number] filename.txt

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

```
A bachelor or postgraduate degree in a related information technology field (such as computer science, network engineering or computer systems
) is usually needed to work as a Computer Network and Systems Engineer. Some workers have Vocational Education and Training (VET) qualificatio
ns. There are also a wide range of vendor and industry certifications available that may substitute for formal qualifications.

Tasks
--More--(91%)
```

c)more +/ [word] [filename.txt] - This option is used to search the string inside your text document. We can view all the instances by 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
```

d)more –d filename.txt – Helps the user to navigate according to the instruction.”Space key” to continue and “q” to quit.

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

```
Researches, analyses, evaluates and monitors network infrastructure to ensure networks are configured to operate at optimal performance.

Assesses and recommends improvements to network operations and integrated hardware, software, communications and operating systems.

Computer Network and Systems Engineers plan, develop, deploy, test and optimise network and system services, taking responsibility for configu
--More--(52%)[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 **DATE:14/3/2023**

Aim

Familiarisation of Linux Commands

CO2

Perform System administration tasks

Procedure

1)grep – Used to filter the content which makes our search easy

a)grep [search word] [filename] -To search for a specific content

```
mca@t2:~/new$ cat > details.txt
maths 50
english 40
science45
hindi 34
social 49
^Z
[3]+  Stopped                 cat > details.txt
mca@t2:~/new$ grep 40 details.txt
english 40
```

b)grep -i [search word] [filename] -Its a case insensitive search

```
mca@t2:~/new$ grep -i english details.txt
english 40
```

c)grep -v [search word] [filename] – All the contents except the searched content will be displayed

```
mca@t2:~/new$ grep -v 49 details.txt
maths 50
english 40
science45
hindi 34
```

d)grep -A1 [search word] [filename] – View the content with one line after

```
mca@t2:~/new$ grep -A1 english details.txt
english 40
science45
```

e)grep -B1 [search word] [filename] - View the content with one line before

```
mca@t2:~/new$ grep -B1 social details.txt
hindi 34
social 49
```

f)grep -C1 [search word] [filename] -View the content with one line before and after.

```
mca@t2:~/new$ grep -C1 science details.txt
english 40
science45
hindi 34
```

2)head – To display the top contents of the file. By default it displays first 10 lines.

a)head [filename]-To display the first 10 lines of the file

```
mca@t2:~/new$ head demo
1
2
3
4
5
6
7
8
9
```

b) head -[limit] [filename]-To display the contents of the file upto a specified limit

```
mca@t2:~/new$ head -5 demo
1
2
3
4
5
```

3)tail – To display the last contents of the file

a)tail [filename]-To display last 10 lines

```
mca@t2:~/new$ tail demo
7
8
9

10
34
56
67
78
89
```

b)tail -[limit] [filename]-To display the last contents of the file upto a specified limit

```
mca@t2:~/new$ tail -5 demo
34
56
67
78
89
```

4)mv – Used for moving a file from one location to another. Its a way of replacing the file. Already written files will be overwritten.

a)mv [file1] [file2] -To move contents from one file to another

```
mca@t2:~/new$ ls
demo  details  details.txt
mca@t2:~/new$ mv demo details.txt
mca@t2:~/new$ cat details.txt
1
2
3
4
5
6
7
8
9

10
34
56
67
78
89
```

```
mca@t2:~/new$ cat demo
cat: demo: No such file or directory
mca@t2:~/new$ ls
details  details.txt
```

b)mv -b [file1] [file2] – Back-up contents in a file

```
mca@t2:~/new$ cat >profile
hello
how are
you
^Z
[7]+  Stopped                 cat > profile
mca@t2:~/new$ mv -b details.txt profile
mca@t2:~/new$ ls
details  profile  profile~
mca@t2:~/new$ cat profile~
hello
how are
you
```

c) mv -i [file1] [file2] – To display the prompt message

```
mca@t2:~/new$ cat > profile1
Amal Jyothi clg
kanjirappally
kottayam
^Z
[8]+  Stopped                 cat > profile1
mca@t2:~/new$ mv -i profile profile1
mv: overwrite 'profile1'?
```

Result

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

Experiment No.: 7 **DATE: 20/3/2023**

Aim

Familiarisation of Linux Commands

CO2

Perform System administration tasks

Procedure

1. expr-To calculate the expression and print the output

```
mca@t2:~$ expr 12 \* 3
36
mca@t2:~$ expr 12 + 8
20
mca@t2:~$ expr 12 - 8
4
```

```
mca@t2:~$ expr 12 / 4
3
```

2. expr-using user input

```
mca@t2:~$ read x
20
mca@t2:~$ read y
25
mca@t2:~$  expr $x + $y
45
```

3. df-It is used to get a report on system space usage

```
mca@t2:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             3966888        0   3966888   0% /dev
tmpfs             799004     1732    797272   1% /run
/dev/sda6      143135900 23654704 112140604  18% /
tmpfs            3995016    17880   3977136   1% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            3995016        0   3995016   0% /sys/fs/cgroup
/dev/loop0         63488    63488         0 100% /snap/core20/1518
/dev/loop1           128      128         0 100% /snap/bare/5
/dev/loop3         64896    64896         0 100% /snap/core20/1828
```

4. du-It is used to check how much space a file or directory takes in the current directory

```
mca@t2:~$ du file1
4       file1
```

5. sudo useradd [username]-Add or create a user

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

6. sudo passwd [username]- To update password

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

7. sudo groupadd -g [group id] [groupname]- To create new group

```
mca@t2:~$ sudo groupadd -g 741 mcadept
```

8. sudo usermod -G [group name] [username]-To add the user to the specified group

```
mca@t2:~$ sudo usermod -G mcadept tinu
```

9. id [user name]-Used to find out user id, group id, group

```
mca@t2:~$ id tinu
uid=1004(tinu) gid=1005(tinu) groups=1005(tinu),741(mcadept)
```

10. compgen -g – To display all the groups

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

11)chmod -Used to change the access permissions or files and directories. It stands for change mode-(read(r),write(w),execute(x))

Here writing permission is denied

```
ubuntu@Tinu:~/newfile$ chmod -wx fil1
ubuntu@Tinu:~/newfile$ cat >> fil1
-bash: fil1: Permission denied
```

Here writing permission is granted

```
ubuntu@Tinu:~/newfile$ chmod +wx fil1
ubuntu@Tinu:~/newfile$ cat >> fil1
name-tinu sara riya jenny
^Z
[3]+  Stopped                 cat >> fil1
```

12)sudo chown [username] [filename]-Used to change a file ownership or directory ownership for a user or a group . Chown stands for change owner.

```
mca@t2:~$ sudo chown tinu new2.txt
```

```
mca@t2:~$ chmod +rwx new2.txt
chmod: changing permissions of 'new2.txt': Operation not permitted
mca@t2:~$ ls -l new2.txt
-rw-rw-r-- 1 tinu mca 37 Mar 13 11:50 new2.txt
```

13) $sudo userdel username-to delete the user

```
mca@t2:~$ sudo userdel tinu
[sudo] password for mca:
```

```
mca@t2:~$ sudo userdel tinu
userdel: user 'tinu' does not exist
```

14)$sudo groupdel groupname-to delete the group

```
mca@t2:~$ sudo groupdel mcadept
mca@t2:~$ sudo groupdel mcadept
groupdel: group 'mcadept' does not exist
```

Result

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

Experiment No.:8 **Date -21-03-2023**

Aim

Familiarisation of Linux Commands

CO2

Perform System Aministration tasks

Procedure

1)ip addr - Get ip address of the system

$ip addr

```
mca@t2:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 40:16:7e:ac:a5:b2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.6.25/24 brd 192.168.6.255 scope global noprefixroute enp3s0
       valid_lft forever preferred_lft forever
    inet6 fe80::cd53:5b58:cc7a:ea37/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

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@192.168.6.39

```
mca@t2:~$ ssh mca@192.168.6.29
ssh: connect to host 192.168.6.29 port 22: Connection refused
```

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

```
mca@t2:~$ sudo apt-get update
[sudo] password for mca:
Get:1 https://dl.google.com/linux/chrome/deb stable InRelease [1,811 B]
Get:2 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal InRelease [17.6 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,079 B]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:7 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal/main amd64 Packages [2,052 B]
Get:8 http://ppa.launchpad.net/maarten-fonville/android-studio/ubuntu focal/main Translation-en [324 B]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2,428 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [568 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [797 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [415 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [275 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [16.2 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu focal-updates/restricted i386 Packages [30.5 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1,671 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [235 kB]
Get:19 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2,046 kB]
Get:20 http://in.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [620 B]
Get:21 http://in.archive.ubuntu.com/ubuntu focal-updates/universe i386 Packages [718 kB]
Get:22 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [333 kB]
Get:23 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [59.8 kB]
Get:24 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [12.3 kB]
Get:25 http://security.ubuntu.com/ubuntu focal-security/restricted i386 Packages [29.1 kB]
Get:26 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1,556 kB]
Get:27 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1,038 kB]
Get:28 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [219 kB]
Get:29 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 c-n-f Metadata [624 B]
Get:30 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [588 kB]
Get:31 http://in.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [243 kB]
Get:32 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [410 kB]
Get:33 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [23.8 kB]
```

```
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-client amd64 1:8.2p1-4ubuntu0.5 [671 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/main amd64 ncurses-term all 6.2-0ubuntu2 [249 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-sftp-server amd64 1:8.2p1-4ubuntu0.5 [51.5 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssh-server amd64 1:8.2p1-4ubuntu0.5 [377 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal/main amd64 ssh-import-id all 5.10-0ubuntu1 [10.0 kB]
Fetched 1,359 kB in 5s (281 kB/s)
Preconfiguring packages ...
(Reading database ... 157949 files and directories currently installed.)
Preparing to unpack .../openssh-client_1%3a8.2p1-4ubuntu0.5_amd64.deb ...
Unpacking openssh-client (1:8.2p1-4ubuntu0.5) over (1:8.2p1-4) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../ncurses-term_6.2-0ubuntu2_all.deb ...
Unpacking ncurses-term (6.2-0ubuntu2) ...
Selecting previously unselected package openssh-sftp-server.
Preparing to unpack .../openssh-sftp-server_1%3a8.2p1-4ubuntu0.5_amd64.deb ...
```

b. sudo ufw allow 22

$sudo ufw allow 22

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

```
mca@t2:~$ ssh mca@192.168.6.23
The authenticity of host '192.168.6.23 (192.168.6.23)' can't be established.
ECDSA key fingerprint is SHA256:4lKDpb+NxGk0v+HcC752LY9CAJEPhNt1GbQHiROYXKI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.6.23' (ECDSA) to the list of known hosts.
mca@192.168.6.23's password:
Connection closed by 192.168.6.23 port 22
```

c. $ssh mca@192.168.6.28

```
mca@t2:~$ ssh mca@192.168.6.23
mca@192.168.6.23'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

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

```
mca@t2:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  rijul  Templates  Videos
mca@t2:~$
```

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): abc.txt
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in abc.txt
Your public key has been saved in abc.txt.pub
The key fingerprint is:
SHA256:d2kI+wkB9Z1VikEHqJB1xeote2eXKJ0IwWAymAxC4ow mca@t2
The key's randomart image is:
+---[RSA 3072]----+
|+oo o .+o .==.o..|
|*  + oo+ o...* . |
|Eo    +.=...+ .  |
|        .*.. .   |
|        S.+.+    |
|         =o+.    |
|          +oo o .|
|          .o.+o..|
|           ..o . |
+----[SHA256]-----+
mca@t2:~$
```

```
mca@t2:~$ cat abc.txt
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBn3gJ1hP
Y3u7LU1xKUtnOiAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDXTThbZ47C
P6eFPLn55/TFNxogWrHeuuTOEpKShf7753abXKNlVMMlrHRrEQ3cKMjsZMZ5siPMcP/Fhk
95NMvkvMEwAAQAsQrsQW6it1PVFT1Ti52yP08et2J5SwvpjQNuvXi5imsQ2QCQlzeayD1L
AaTuilr8IMp8mMHo6BbjAlE2p26bpTG/ff9ppYV/mWDmtbBlLb/Y6HjfybvolgTsYfkvih
3JNMAn487rlXDHKgWGPZt3JwEgRdRGOmHmmz2UwxipHZp8PkP++BfF6UGwCAWE8BDC3Y1u
TJrNY2WDnAb6pVx8IzsNHksk/poZ1ebk6v9XIGYkKiEgEWOn8BB+GvnSBtCWVMGjA5nFuZ
gpyYjcaQy8fa7+xRdk5BrbzeWjhLWKscNntdWSfGmzV+tsLLVYM1R2BS86IXFc/r5kgZHQ
LbZG6UCRqSlRhNZPIZFMRLsvqMIKDCtbOl5OnbwvaWThHbGw4DjonrItz+mzY56/FQ0q2w
VG98uNTa3u1R0AAAWAISJpbrK56rYoj+toR6KZH7fRKCLnZqQh7/Yqyg1wYyf9zsamZNIb
Yia1BMaPtgFYmfmozwzcuWojgwgG8F+ZbBfhxuRP1TUjqzpxQGqpPOz7zAKKNXp63zgxCa
hlAdKkgd5JUhjadt9CUHKXFt1Owq6iH0zRztgQhuhZWDNTL02Ut5rqUFs6reu1B33dNuzy
FoKALnRqssIdjLt46g+Z7vsAwNO3d+QiLMTSFoU8C5Bk84urM8kWj+XaYD9DP3UMSeliHr
5vcNy/43qVzrfeRKBoaYqbNHIunK6IUEeJqoUZx1epWp/5XI7YM940iVgGLJ0I2S9cXy3k
l4NdeJHJKBwAq+pPAd4l1BV3w9w/vcYmOtcs6cC7piytO0inSU4N+qAUmRNaiOWGMO5Fju
ktP03KHmVM/NSw/fcTznVKXBjPosNRqn/uMItBxpXOgcjk34Q853Q91cBqgJXAA4nmW3BZ
i8eykwYK1yR8hzrFATrHDUAyP+GMfHHiKfpzIQ8NfR/Z8qxfDfnvWR9GvL8cIfA6IancTa
07HeKGJh2cPEzGUk97j6nCa635VnDcI9kd+D1N0Z5qxhcjdLqAbWec5Gax1qNwa07JDq9h
Y+WSgBKAxKsjeRQt1Qt5EENFU35jXclC/4yqmsYKOIoJkVSOOmNlBgBs3XXqcxXzQo8mRw
o2n5ISjuI8+tGilogsgBBdvcp3yPmqw7T1T2LdUTHk9aDzDV1SepCsyYgMdeYTHixccdrz
7Z7RfDPwklXNd4g0wYt6w0OtkrmE2+h72F2XsetQv9qj5c/nnV3T+0FJqFRMspWrX9wSp1
Joh8So7UvLo67eKZO15WglbzNaPSQN6soikzIqWHcnSE2GYRunGUq8eD1p78DmvkhB0jXh
lsjjYmaAZVWkEsXVtzD+1zHl//UlSVmkNnYaPEEMrHBPailQkVKjt9Zirs/iFqDb3tMi88
cY5U3OmKDDRSooxeavykvCJYIlJa6sUtjupPQWjhWuBsQaaaUDhwV9YtA9ZQRc1tDUeEhk
Tp/R+ry+ga2y64oDH4TDkCz4cmR43FIaZtZ48WHt52LF6ko98pfosve3HTta9gDlpTYIyE
lYe/Ff09EppBlNnPHurmfsy9ViDZT8hUYUjiNo1LIX8yd2eLZKji5MgJPCggjopEvOCvez
WIX2ivmsGEoC9Q1WA8TRW55W4WhVmxz/tiR1M/x3FyRYL7BXI0yZKKTKtbe60yYvYaMoGV
cQLewe2IWwLltmGMTvc8etuUwq31ko21bjAY+DKtz+NzGPEHC7vYOrAC/rY879GQDbe8Em
A9QZrxR4Tw+u11HDcgBf+H8zbuae72SDodzZ3H7F7vwZ1MhmVig/qZ+gn/a/HWMyxPfscM
twHhjTcj3LwatAYCFIn7yAceKAaQJy/GjnqvfTYasweG+R07EOGkUXt/5kqK6LK8yQFy0H
l1xL+MHINO7y9stFuONh7HEH6VKLbDrM4/oQt2MlbCHSFgtkVEramGqjdzazCjlNibRpWA
Cmatd5MmorsNxM33cfPWPHClWTisIvXc8LRwbBPOT3BBi34o/kUwKe1itfq+vxsvnXSZZA
```

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

a. $ps

```
mca@t2:~$ ps
    PID TTY          TIME CMD
   7271 pts/1    00:00:00 bash
   8446 pts/1    00:00:00 ps
mca@t2:~$
```

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

$ps -u mca

```
mca@t2:~$ ps -u mca
    PID TTY          TIME CMD
   1377 ?        00:00:00 systemd
   1387 ?        00:00:00 (sd-pam)
   1395 ?        00:00:00 pulseaudio
   1397 ?        00:00:00 tracker-miner-f
   1400 ?        00:00:00 dbus-daemon
   1405 ?        00:00:00 gnome-keyring-d
   1408 ?        00:00:00 gvfsd
   1413 ?        00:00:00 gvfsd-fuse
   1432 ?        00:00:00 gvfs-udisks2-vo
   1437 ?        00:00:00 gvfs-mtp-volume
   1442 ?        00:00:00 gvfs-goa-volume
   1446 ?        00:00:00 goa-daemon
   1454 ?        00:00:00 goa-identity-se
   1459 ?        00:00:00 gvfs-gphoto2-vo
   1464 ?        00:00:00 gvfs-afc-volume
   1471 tty2     00:00:00 gdm-x-session
   1473 tty2     00:00:53 Xorg
   1497 tty2     00:00:00 gnome-session-b
   1575 ?        00:00:00 ssh-agent
   1592 ?        00:00:03 ibus-daemon
   1599 ?        00:00:00 ibus-dconf
   1600 ?        00:00:00 ibus-ui-gtk3
   1603 ?        00:00:01 ibus-extension-
   1609 ?        00:00:00 ibus-x11
   1612 ?        00:00:00 ibus-portal
   1613 ?        00:00:00 at-spi-bus-laun
   1622 ?        00:00:00 dbus-daemon
   1631 ?        00:00:00 at-spi2-registr
   1639 ?        00:00:00 xdg-desktop-por
   1644 ?        00:00:00 gnome-session-c
   1652 ?        00:00:00 xdg-document-po
   1655 ?        00:00:00 xdg-permission-
   1664 ?        00:00:00 gnome-session-b
```

c)ps -C :- Specific process

$ps -C chrome

```
mca@t2:~$ ps -C chrome
    PID TTY          TIME CMD
   2979 ?        00:00:48 chrome
   2996 ?        00:00:00 chrome
   2997 ?        00:00:00 chrome
   3001 ?        00:00:00 chrome
   3024 ?        00:00:37 chrome
   3025 ?        00:00:20 chrome
   3049 ?        00:00:00 chrome
   3271 ?        00:00:00 chrome
   3496 ?        00:00:38 chrome
   3508 ?        00:00:56 chrome
   7677 ?        00:00:04 chrome
   7758 ?        00:00:00 chrome
   8127 ?        00:00:00 chrome
```

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

$ps -f -p 2762

```
mca@t2:~$ ps -f -p 2979
UID          PID    PPID  C STIME TTY          TIME CMD
mca         2979    1708  1 14:58 ?        00:00:49 /opt/google/chrome/chrome --enable-crashpad
```

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 the date

```
ubuntu@Tinu:~$ vi firstprogram.sh
ubuntu@Tinu:~$ chmod +x firstprogram.sh
ubuntu@Tinu:~$ ./firstprogram.sh
./firstprogram.sh: line 2: date#!/bin/bash: No such file or directory
Mon Apr  3 07:34:35 IST 2023
```

```
#!/bin/bash
date#!/bin/bash
```

2. Shell Script to display your name

```
ubuntu@Tinu:~$ vi secondprogram.sh
ubuntu@Tinu:~$ chmod +x secondprogram.sh
ubuntu@Tinu:~$ ./secondprogram.sh
What is your name
Tinu
My name is Tinu
```

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

3. Shell Script to display date , pwd , ls(multiple commands)

```
ubuntu@Tinu:~$ vi program3.sh
ubuntu@Tinu:~$ chmod +x program3.sh
ubuntu@Tinu:~$ .//program3.sh
Mon Apr  3 07:50:23 IST 2023
1stprgm.sh  firstprgm.sh     newfile          secondprogram.sh
file1       firstprogram.sh  program3.sh      thirdprogram.sh
file2       new              programthird.sh
/home/ubuntu
mkdir: cannot create directory 'new': File exists
Mon Apr  3 07:50:23 IST 2023
1stprgm.sh  firstprgm.sh     newfile          secondprogram.sh
file1       firstprogram.sh  program3.sh      thirdprogram.sh
file2       new              programthird.sh
/home/ubuntu
mkdir: cannot create directory 'new': File exists
```

```
#!/bin/bash
date
ls
pwd
mkdir file1
```

4.Shell Script to demonstrate special variables

```
ubuntu@Tinu:~$ chmod +x spclvariable.sh
ubuntu@Tinu:~$ ./spclvariable.sh Tinu clara Emmanuel
File name: ./spclvariable.sh
First parameter: Tinu
Second parameter: clara
Quoted values: Tinu clara Emmanuel
Quoted values: Tinu clara Emmanuel
Total parameters: 3
File name: ./spclvariable.sh
First parameter: Tinu
Second parameter: clara
Quoted values: Tinu clara Emmanuel
Quoted values: Tinu clara Emmanuel
Total parameters: 3
```

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

5.Shell Script to count lines and words of the file

```
ubuntu@Tinu:~$ readlink -f secondprogram.sh
/home/ubuntu/secondprogram.sh
```

```
ubuntu@Tinu:~$ vi countwords2.sh
ubuntu@Tinu:~$ chmod +x countwords2.sh
ubuntu@Tinu:~$ ./countwords2.sh
Number of lines:4
Number of words: 13
```

```
file_path=/home/ubuntu/secondprogram.sh
countlines=`wc --lines < $file_path`
countwords=`wc --word <$file_path`
echo "Number of lines:$countlines"
echo "Number of words:$countwords"
```

6.Shell Script to display array index

```
ubuntu@Tinu:~$ vi array.sh
ubuntu@Tinu:~$ chmod +x array.sh
ubuntu@Tinu:~$ ./array.sh
First index: Tinu
Second index: Sara
```

```
Name[0]="Tinu"
Name[1]="Sara"
Name[2]="Jenny"
Name[3]="Riya"
Name[4]="sebin"
echo "First index: ${Name[0]}"
echo "Second index: ${Name[1]}"
```

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 2 numbers

```
ubuntu@Tinu:~$ vi add.sh
ubuntu@Tinu:~$ chmod +x add.sh
ubuntu@Tinu:~$ ./add.sh
Total value: 15
```

```
#!/bin/bash
a=` expr 2 + 13`
echo "Total value: $a"
```

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

```
ubuntu@Tinu:~$ vi add1.sh
ubuntu@Tinu:~$ chmod +x add1.sh
ubuntu@Tinu:~$ ./add1.sh
sum: 30
ubuntu@Tinu:~$
```

```
a=20
b=10
sum=$(( $a + $b ))
echo "sum: $sum"
```

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

```
ubuntu@Tinu:~$ vi add_command_line.sh
ubuntu@Tinu:~$ chmod +x add_command_line.sh
ubuntu@Tinu:~$ ./add_command_line.sh 10 10
Sum: 20
```

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

4. Shell script which take input from the user at run time, then calculate the sum of given numbers and store to the variable and show the results

```
ubuntu@Tinu:~$ vi add_user_input.sh
ubuntu@Tinu:~$ chmod +x add_user_input.sh
ubuntu@Tinu:~$ ./add_user_input.sh
Enter first number: 10
Enter second number:  20
Sum: 30
```

```
#!/bin/bash
#!/bin/bash
read -p "Enter first number: " num1
read -p "Enter second number:  " num2
sum=$(( $num1 + $num2 ))
echo "Sum: $sum"
```

1.Shell Script to subtract 2 numbers

```
ubuntu@Tinu:~$ vi subt.sh
ubuntu@Tinu:~$ chmod +x subt.sh
ubuntu@Tinu:~$ ./subt.sh
Difference= 10
```

```
#!/bin/bash
diff=` expr 20 - 10`
echo "Difference= $diff"
```

2. Write a shell script to initialise two numeric variables then perform an subtraction operation on both the values and store the result in the third variable.

```
ubuntu@Tinu:~$ vi subt_initialise.sh
ubuntu@Tinu:~$ chmod +x subt_initialise.sh
ubuntu@Tinu:~$ ./subt_initialise.sh
Difference= 11
```

```
#!/bin/bash
a=20
b=9
diff=$(( $a - $b ))
echo "Difference= $diff"
```

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

```
ubuntu@Tinu:~$ vi subt_command_line.sh
ubuntu@Tinu:~$ chmod +x subt_command_line.sh
ubuntu@Tinu:~$ ./subt_command_line.sh 30 9
Difference: 21
```

```
#!/bin/bash
diff=$(( $1 -$2 ))
echo "Difference: $diff"
```

4. Shell script which take input from the user at run time, then calculate the difference of given numbers and store to the variable and show the results

```
ubuntu@Tinu:~$ vi subt_user_input.sh
ubuntu@Tinu:~$ chmod +x subt_user_input.sh
ubuntu@Tinu:~$ ./subt_user_input.sh
Enter number 1: 40
Enter number 2: 10
Difference = 30
```

```
#!/bin/bash
read -p "Enter number 1: " num1
read -p "Enter number 2: " num2
diff=$(( $num1 - $num2 ))
echo "Difference = $diff"
```

1.Shell Script to multiply 2 numbers

```
ubuntu@Tinu:~$ vi prdct.sh
ubuntu@Tinu:~$ chmod +x prdct.sh
ubuntu@Tinu:~$ ./prdct.sh
Product= 40
```

```
#!/bin/bash
p=`expr 20 \* 2`
echo "Product= $p"
```

2. Write a shell script to initialise two numeric variables then perform an multiplication operation on both the values and store the result in the third variable.

```
ubuntu@Tinu:~$ vi prdct_initialise.sh
ubuntu@Tinu:~$ chmod +x prdct_initialise.sh
ubuntu@Tinu:~$ ./prdct_initialise.sh
Product= 60
```

```
#!/bin/bash
a=20
b=3
p=$(( $a * $b ))
echo "Product= $p"
```

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

```
ubuntu@Tinu:~$ vi prdct_command_line.sh
ubuntu@Tinu:~$ chmod +x prdct_command_line.sh
ubuntu@Tinu:~$ ./prdct_command_line.sh 60 3
Product= 180
```

```
#!/bin/bash
prdct=$(( $1 * $2 ))
echo "Product= $prdct"
```

4. Shell script which take input from the user at run time, then calculate the product of given numbers and store to the variable and show the results

```
ubuntu@Tinu:~$ vi prdct_user_input.sh
ubuntu@Tinu:~$ chmod +x prdct_user_input.sh
ubuntu@Tinu:~$ ./prdct_user_input.sh
Enter first number: 20
Enter second number: 3
Product= 60
```

```
#!/bin/bash
read -p "Enter first number: " num1
read -p "Enter second number: " num2
p=$(( $num1 * $num2 ))
echo "Product= $p"
```

1.Shell Script to divide 2 numbers

```
ubuntu@Tinu:~$ vi quo.sh
ubuntu@Tinu:~$ chmod +x quo.sh
ubuntu@Tinu:~$ ./quo.sh
Quotient= 6
```

```
#!/bin/bash
q=`expr 20 / 3`
echo "Quotient= $q"
```

2. Write a shell script to initialise two numeric variables then perform an division operation on both the values and store the result in the third variable.

```
ubuntu@Tinu:~$ vi quo_initialise.sh
ubuntu@Tinu:~$ chmod +x quo_initialise.sh
ubuntu@Tinu:~$ ./quo_initialise.sh
Quotient= 30
```

```
#!/bin/bash
a=60
b=2
q=$(( $a / $b ))
echo "Quotient= $q"
```

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

```
ubuntu@Tinu:~$ vi quo_command_line.sh
ubuntu@Tinu:~$ chmod +x quo_command_line.sh
ubuntu@Tinu:~$ ./quo_command_line.sh 6 3
Quotient= 2
```

```
#!/bin/bash
q=$(( $1 / $2 ))
echo "Quotient= $q"
```

4. Shell script which take input from the user at run time, then calculate the quotient of given numbers and store to the variable and show the results

```
ubuntu@Tinu:~$ vi quo_user_inpt.sh
ubuntu@Tinu:~$ chmod +x quo_user_inpt.sh
ubuntu@Tinu:~$ ./quo_user_inpt.sh
Enter first number: 6
Enter second number: 3
Quotient= 2
```

```
#!/bin/bash
read -p "Enter first number: " num1
read -p "Enter second number: " num2
q=$(( num1 / num2 ))
echo "Quotient= $q"
```

1.Shell Script to find modulo of 2 numbers

```
ubuntu@Tinu:~$ vi modulo.sh
ubuntu@Tinu:~$ chmod +x modulo.sh
ubuntu@Tinu:~$ ./modulo.sh
Remainder= 0
```

2. Write a shell script to initialise two numeric variables then perform modulus operation on both the values and store the result in the third variable.

```
ubuntu@Tinu:~$ vi modulo_initialise.sh
ubuntu@Tinu:~$ chmod +x modulo_initialise.sh
ubuntu@Tinu:~$ ./modulo_initialise.sh
Remainder= 1
```

```
#!/bin/bash
a=6
b=5
r=$(( $a % $b ))
echo "Remainder= $r"
```

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

```
ubuntu@Tinu:~$ vi modulo_command_line.sh
ubuntu@Tinu:~$ chmod +x modulo_command_line.sh
ubuntu@Tinu:~$ ./modulo_command_line.sh 6 3
Remainder= 0
```

```
#!/bin/bash
r=$(( $1 % $2 ))
echo "Remainder= $r"
```

4. Shell script which take input from the user at run time, then calculate the modulus of given numbers and store to the variable and show the results

```
ubuntu@Tinu:~$ vi modulo_user_input.sh
ubuntu@Tinu:~$ chmod +x modulo_user_input.sh
ubuntu@Tinu:~$ ./modulo_user_input.sh
Enter a: 20
Enter b: 3
Remainder= 2
```

```
#!/bin/bash
read -p "Enter a: " a
read -p "Enter b: " b
r=$(($a % $b ))
echo "Remainder= $r"
```

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 arithmetic operations

```
ubuntu@Tinu:~$ vi operations.sh
ubuntu@Tinu:~$ chmod +x operations.sh
ubuntu@Tinu:~$ ./operations.sh
Enter a number: 13
Enter another number: 10
Sum: 23
Difference: 3
Product: 130
Quotient: 1
Remainder: 3
a is not equal to b
Increment operator on a is 14
Decrement operator on b is 9
```

```
#!/bin/bash
read -p "Enter a number: " a
read -p "Enter another number: " b
add=$(( a + b ))
echo "Sum: $add"
diff=$(( a - b ))
echo "Difference: $diff"
prduct=$(( a * b ))
echo "Product: $prduct"
quo=$(( a / b ))
echo "Quotient: $quo"
mod=$(( a % b ))
echo "Remainder: $mod"
if [ $a == $b ]
then
        echo "a is equal to b"
fi
if [ $a != $b ]
then
        echo "a is not equal to b"
fi
(( ++a ))
echo "Increment operator on a is $a"
(( --b ))
echo "Decrement operator on b is $b"
```

2.Shell script to demonstrate relational operation

```
ubuntu@Tinu:~$ vi relation.sh
ubuntu@Tinu:~$ ./relation.sh
Enter a: 20
Enter b: 10
a not equal to b
a not equal to b
a not less than b
a not less than or equal to b
a greater than b
a greater than or equal to b
```

```
#!/bin/bash
read -p "Enter a: " a
read -p "Enter b: " b
if(($a == $b))
then
        echo "a = b"
else
        echo "a not equal to b"
fi
if(($a != $b))
then
        echo "a not equal to b"
else
        echo "a equal to b"
fi
if(($a < $b))
then
        echo "a less than b"
else
        echo "a not less than b"
fi
if(($a <= $b))
then
         echo "a less than or equal to b"
else
          echo "a not less than or equal to b"
fi
if(($a > $b))
then
       echo "a greater than b"
else
       echo "a not greater than b"
fi
if(($a >= $b))
then
       echo "a greater than or equal to b"
else
        echo "a not geater than or equal to b"
fi
~
```

3.Shell script to demonstrate relational operation(another method)

```
ubuntu@Tinu:~$ vi relation1.sh
ubuntu@Tinu:~$ chmod +x relation1.sh
ubuntu@Tinu:~$ ./relation1.sh
a not equal to b
a not equal to b
a less than b
a less than or equal to b
a not greater than b
a not geater than or equal to b
```

```
#!/bin/bash
a=10
b=20
if [ $a -eq  $b ]
then
            echo "a equal b"
else
            echo "a not equal to b"
fi
if [ $a -ne $b ]
then
            echo "a not equal to b"
else
           echo "a equal to b"
fi
if [ $a -lt  $b ]
then
           echo "a less than b"
else
            echo "a not less than b"
fi
if [ $a -le $b ]
then
           echo "a less than or equal to b"
else
           echo "a not less than or equal to b"
fi
if [ $a  -gt $b ]
then
            echo "a greater than b"
else
            echo "a not greater than b"
fi
if [ $a -ge $b ]
then
            echo "a greater than or equal to b"
else
           echo "a not geater than or equal to b"
fi
```

4.Shell script to demonstrate logical operations

```
ubuntu@Tinu:~$ vi logical.sh
ubuntu@Tinu:~$ chmod +x logical.sh
ubuntu@Tinu:~$ ./logical.sh
Enter a: true
Enter b: true
Both are true
One of them is true
a was initially true
```

```
#!/bin/bash
read -p "Enter a: " a
read -p "Enter b: " b
if(($a == "true" & $b == "true"))
then
        echo "Both are true"
else
        echo "Both are not true"
fi
if(($a == "true" || $b == "true"))
then
           echo "One of them is true"
else
          echo "None is true"
fi
if [ !a ==  "true" ]
then
          echo "a was initially false"
else
        echo "a was initially true"
fi
```

5.Write a shell script to check whether a number is odd or even

```
ubuntu@Tinu:~$ vi odd_or_even.sh
ubuntu@Tinu:~$ chmod +x odd_or_even.sh
ubuntu@Tinu:~$ ./odd_or_even.sh
Enter a number: 20
Number is even
ubuntu@Tinu:~$ vi odd_or_even.sh
ubuntu@Tinu:~$ ./odd_or_even.sh
Enter a number: 13
Number is odd
```

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

6.Write a shell script to check whether a number is positive, negative or zero

```
ubuntu@Tinu:~$ vi positive_or_not.sh
ubuntu@Tinu:~$ chmod +x positive_or_not.sh
ubuntu@Tinu:~$ ./positive_or_not.sh
Enter a number: 20
Number is positive
ubuntu@Tinu:~$ ./positive_or_not.sh
Enter a number: -2
Number is negative
ubuntu@Tinu:~$ ./positive_or_not.sh
Enter a number: 0
The number is zero
```

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

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

```
ubuntu@Tinu:~$ vi largest_two.sh
ubuntu@Tinu:~$ chmod +x largest_two.sh
ubuntu@Tinu:~$ ./largest_two.sh
Enter a: 20
Enter b: 10
20 is greater
```

```
#!/bin/bash
read  -p "Enter a: " a
read -p "Enter b: " b
if [ $a > $b ]
then
        echo "$a is greater"
else
        echo "$b is greater"
fi
```

8.Write a shell script to find largest among three numbers

```
ubuntu@Tinu:~$ vi largest_three.sh
ubuntu@Tinu:~$ ./largest_three.sh
Enter a: 20
Enter b: 10
Enter c: 5
20 is greater
```

```
#!/bin/bash
read -p "Enter a: " a
read -p "Enter b: " b
read -p "Enter c: " c
if(($a > $b && $a > $c))
then
        echo "$a is greater"
elif(($b > $a && $b > $c))
then
        echo "$b is greater"
else
        echo "$c is greater"
fi
```

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

```
ubuntu@Tinu:~$ vi string.sh
ubuntu@Tinu:~$ chmod +x string.sh
ubuntu@Tinu:~$ ./string.sh
Enter 1st string: anu
Enter 2nd string: tinu
Both strings are not equal
Both strings are equal
String size is not zero
String size is zero
String is not empty
```

```
#!/bin/bash
read -p "Enter 1st string: " a
read -p "Enter 2nd string: " b
if(( $a=$b ))
then
        echo "Both strings are equal"
else
        echo "Both strings are not equal"
fi
if(( $a!=$b ))
then
        echo "Both strings are not equal"
else
        echo "Both strings are equal"
fi
if(( -z$a ))
then
        echo "String size is zero"
else
        echo "String size is not zero"
fi
if(( -n$a ))
then
        echo "String size is non zero"
else
        echo "String size is zero"
fi
if (( $a ))
then
        echo "String is empty"
else
        echo "String is not empty"
fi
```

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

```
ubuntu@Tinu:~$ vi bitwiseop.sh
ubuntu@Tinu:~$ chmod +x bitwiseop.sh
ubuntu@Tinu:~$ ./bitwiseop.sh
Enter 1st value: 10
Enter 2nd value: 8
Bitwise AND: 8
Bitwise OR: 10
Bitwise XOR: 2
Bitwise Complement of 1st value: -11
Bitwise right shift(4) of ist value: 40
Bitwise left shift(4) of 1st value: 2
```

```
#!/bin/bash
read -p "Enter 1st value: " a
read -p "Enter 2nd value: " b
result=$(( $a&$b ))
echo "Bitwise AND: $result"
result=$(( $a|$b ))
echo "Bitwise OR: $result"
result=$(( $a^$b ))
echo "Bitwise XOR: $result"
result=$(( ~$a ))
echo "Bitwise Complement of 1st value: $result"
result=$(( $a<<2 ))
echo "Bitwise right shift(4) of ist value: $result"
result=$(( $a>>2 ))
echo "Bitwise left shift(4) of 1st value: $result"

~
```

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

```
ubuntu@Tinu:~$ vi filetest.sh
ubuntu@Tinu:~$ chmod +x filetest.sh
ubuntu@Tinu:~$ ./filetest.sh
Enter file name: new
new does not exist
new is not empty
new has read permission
new has execute permission
new has write permission
```

```
#!/bin/bash
read -p "Enter file name: " f
if(( -e$f ))
then
        echo "$f exist"
else
        echo "$f does not exist"
fi
if [ -s$f ]
then
        echo "$f is not empty"
else
        echo "$f is empty"
fi
if [ -r$f ]
then
        echo "$f has read permission"
else
        echo "$f does not have read permission"
fi
if [ -x$f ]
then
        echo "$f has execute permission"
else
        echo "$f does not have execute permission"
fi
if [ -w$f ]
then
        echo "$f has write permission"
else
        echo "$f does not have write permission"
fi

~
```

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

```
ubuntu@Tinu:~$ vi equal.sh
ubuntu@Tinu:~$ chmod +x equal.sh
ubuntu@Tinu:~$ ./equal.sh
enter 1st number: 12
Enter 2nd number: 16
Both numbers are not equal
```

```
#!/bin/bash
read -p "enter 1st number: " a
read -p "Enter 2nd number: " b
if(( $a==$b ))
then
        echo "Both numbers are equal"
fi
if(( $a!=$b ))
then
        echo "Both numbers are not equal"
fi
```

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

```
ubuntu@Tinu:~$ vi range.sh
ubuntu@Tinu:~$ chmod +x range.sh
ubuntu@Tinu:~$ ./range.sh
Enter the number between 0 and 150: 60
60 is between 51 and 100
```

```
#!/bin/bash
read -p "Enter the number between 0 and 150: " a
if(( $a>0&&$a<50 ))
then
        echo "$a is between 1 and 50"
elif(( $a>51&&$a<100 ))
then
        echo "$a is between 51 and 100"
else
        echo "$a is between 101 and 150"
fi
~
```

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

```
ubuntu@Tinu:~$ vi grade.sh
ubuntu@Tinu:~$ chmod +x grade.sh
ubuntu@Tinu:~$ ./grade.sh
Enter the mark: 90
B1
```

```
#!/bin/bash
read -p "Enter the mark: " a
if(( $a<=100&&$a>=91 ))
then
        echo "A1"
elif(( $a<=90&&$a>=81 ))
then
        echo "B1"
elif(( $a<=80&&$a>=71 ))
then
        echo "C1"
elif(( $a<=70&&$a>=61 ))
then
        echo "D1"
else
        echo "FAIL"
fi

~
```

Result

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