Experiment 3

Date: 06/03/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO2):

Perform system administration task

Procedure:

1. pwd :- print working directory
   $pwd
   Output:

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

2. ls :- list directory content
   $ls
   Output:

```
student@t2:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  PycharmProjects  snap  temp  Templates  Videos
student@t2:~$
```

a) ls -R :- list subdirectories recursively
   $ls -R
   Output:

```
student@t2:~$ ls -R
.:
Desktop  Documents  Downloads  Music  Pictures  Public  PycharmProjects  snap  temp  Templates  Videos

./Desktop:

./Documents:

./Downloads:
'NSA _Content.pdf'

./Music:

./Pictures:
 ls.png  'ls -t.png'   pwd.png
```

b) ls -l :- Use long listing format

$ls -l

Output:

```
nithin@DESKTOP-389ELIT:~$ ls -l
total 32
drwxr-xr-x 3 nithin nithin 4096 Apr 12 06:48 String
drwxr-xr-x 2 nithin nithin 4096 Apr  4 13:42 addition
drwxr-xr-x 2 nithin nithin 4096 Apr  4 17:12 basics
drwxr-xr-x 2 nithin nithin 4096 Apr  4 13:51 division
drwxr-xr-x 2 nithin nithin 4096 Apr  4 13:54 modulus
drwxr-xr-x 2 nithin nithin 4096 Apr  4 13:57 multiplication
drwxr-xr-x 2 nithin nithin 4096 Apr  4 21:54 old
drwxr-xr-x 2 nithin nithin 4096 Apr  4 13:59 subtract
nithin@DESKTOP-389ELIT:~$ ~
```

c) ls -al :- list files and directories with detailed information such as permissions, size and owner

$ls -al

Output:

```
student@t2:~$ ls -al
total 104
drwxr-xr-x 22 student student 4096 Mar  7 15:46 .
drwxr-xr-x  6 root    root    4096 Jun 17  2022 ..
-rw-------  1 student student 1911 Mar  7 15:50 .bash_history
-rw-r--r--  1 student student  220 Jun 17  2022 .bash_logout
-rw-r--r--  1 student student 3771 Jun 17  2022 .bashrc
drwxrwxr-x 23 student student 4096 Mar  7 15:36 .cache
drwxr-xr-x 19 student student 4096 Mar  7 15:38 .config
drwxr-xr-x  2 student student 4096 Jun 17  2022 Desktop
drwxr-xr-x  2 student student 4096 Mar  7 15:50 Documents
drwxr-xr-x  2 student student 4096 Mar  7 15:30 Downloads
drwx------  3 student student 4096 Mar  7 14:36 .gnupg
drwxrwxr-x  4 student student 4096 Jun 17  2022 .java
drwxr-xr-x  3 student student 4096 Jun 17  2022 .local
drwx------  4 student student 4096 Mar  6 11:19 .mozilla
drwxr-xr-x  2 student student 4096 Jun 17  2022 Music
drwxr-xr-x  2 student student 4096 Mar  7 16:03 Pictures
drwx------  3 student student 4096 Mar  6 11:20 .pki
-rw-r--r--  1 student student  807 Jun 17  2022 .profile
drwxr-xr-x  2 student student 4096 Jun 17  2022 Public
drwxrwxr-x  3 student student 4096 Jun 17  2022 PycharmProjects
drwx------  4 student student 4096 Mar  6 11:19 snap
drwx------  2 student student 4096 Jun 17  2022 .ssh
drwxrwxr-x  2 student student 4096 Mar  7 15:10 temp
drwxr-xr-x  2 student student 4096 Jun 17  2022 Templates
drwx------  6 student student 4096 Mar  6 11:21 .thunderbird
drwxr-xr-x  2 student student 4096 Jun 17  2022 Videos
student@t2:~$
```

d) ls -a :- List hidden files

$ls -a

Output:

```
nithin@DESKTOP-389ELIT:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .motd_shown  .profile  .viminfo  String  addition  basics  division  modulus  multiplication  old  subtract
nithin@DESKTOP-389ELIT:~$
```

e) ls -t :- Sort by modification time, newest first

$ls -t

Output:

```
student@t2:~$ ls -t
Pictures  Documents  Downloads  temp  snap  PycharmProjects  Desktop  Music  Public  Templates  Videos
student@t2:~$
```

f) ls -r :- Reverse order while sorting

$ls -r

Output:

```
student@t2:~$ ls -r
Videos  Templates  temp  snap  PycharmProjects  Public  Pictures  Music  Downloads  Documents  Desktop
student@t2:~$
```

3. history : - Review all previously executed commands right from the shell

$history

Output:

```
student@t2:~$ history
    1  ./studio.sh
    2  ./studio.sh
    3  su mca
    4  cd mca
    5  pwd
    6  ls
    7  ls-R
    8  ls -R
    9  ls -l
   10  ls -a
   11  ls -al
   12  ls -t
   13  ls -r
   14  hidtory
   15  history
```

4. man :- An interface to system reference manuals

$man ls

Output:

```
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 ..

       --author
              with -l, print the author of each file
```

5. cd :- Change directory

   $cd

   Output:

```
nithin@DESKTOP-389ELIT:~$ cd addition
nithin@DESKTOP-389ELIT:~/addition$
```

6. mkdir :- Make directory

   $mkdir test

   Output:

```
nithin@DESKTOP-389ELIT:~/addition$ mkdir test
nithin@DESKTOP-389ELIT:~/addition$ ls
add1.sh  add2.sh  add3.sh  add4.sh  test
```

7. rmdir :- Remove empty directories

   $rmdir test

   Output:

```
nithin@DESKTOP-389ELIT:~/addition$ rmdir test
nithin@DESKTOP-389ELIT:~/addition$ ls
add1.sh  add2.sh  add3.sh  add4.sh
```

8. touch :- Create empty file

   $touch

   Output:

```
nithin@DESKTOP-389ELIT:~/addition$ touch testfile.txt
nithin@DESKTOP-389ELIT:~/addition$ ls
add1.sh  add2.sh  add3.sh  add4.sh  testfile.txt
```

9. cat :- Concatenate files and print on the standard output

a) cat > sample.txt :- Create and write in new file

   $cat > sample.txt

   Output:

```
student@t2:~/nithin$ cat >sample2.txt
Nithin Jose
Manjadiyil House
Chirakkadavu
^Z
[1]+  Stopped                 cat > sample2.txt
student@t2:~/nithin$ ls
sample2.txt  sample.txt
student@t2:~/nithin$
```

b) cat sample.txt :- Print contents of the file

   $cat sample2.txt

   Output:

```
student@t2:~/nithin$ cat sample2.txt
Nithin Jose
Manjadiyil House
Chirakkadavu
student@t2:~/nithin$
```

c) cat >> sr.txt :- Append information in already existing file

   $cat >> sr.txt

   Output:

```
student@t2:~/nithin$ cat >>sample
Nithin Rajappan
Puthenpurackal House
Vagamon
^Z
[2]+  Stopped                 cat >> sample
student@t2:~/nithin$ cat sample
Nithin Rajappan
Puthenpurackal House
Vagamon
student@t2:~/nithin$
```

d) cat sr.txt file.txt > output.txt :- Copy contents of two files to a third new file
$cat sample sample2 >students.txt

Output:

```
student@t2:~/nithin$ cat sample
Nithin Rajappan
Puthenpurackal House
Vagamon
student@t2:~/nithin$ cat sample2
Nithin Jose
Manjadiyil House
Chirakkadavu
student@t2:~/nithin$ cat sample sample2 >students.txt
student@t2:~/nithin$ cat students.txt
Nithin Rajappan
Puthenpurackal House
Vagamon
Nithin Jose
Manjadiyil House
Chirakkadavu
student@t2:~/nithin$
```

e) cat -n output.txt :- Number all output lines
$cat -n students.txt

Output:

```
student@t2:~/nithin$ cat -n students
     1  Nithin Jose
     2  Manjadiyil House
     3  Chirakkadavu
     4  Nithin Rajappan
     5  Puthenpurackal House
     6  Vagomon
student@t2:~/nithin$
```

f) cat -b sr.txt :- Remove numbering for empty lines
$cat -b sr.txt

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cat -b students.txt
     1  Nithin Jose
     2  Nithin rajappan
     3  Philip Antony
```

g) cat -e students :- Display $ at end of each line
$cat -e students.

Output:

```
student@t2:~/nithin$ cat -e students
Nithin Jose$
Manjadiyil House$
Chirakkadavu$
Nithin Rajappan$
Puthenpurackal House$
Vagomon$
student@t2:~/nithin$
```

h) cat << EOF :- Displays an end marker at the end of a file.

$cat > students.txt <<EOF

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cat > students.txt <<EOF
> Nithin Jose
> Philip Antony
> Nithin Rajappan
> EOF
nithin@DESKTOP-389ELIT:~/addition$ cat students.txt
Nithin Jose
Philip Antony
Nithin Rajappan
```

i) cat file.txt | tr a-z A-Z > output1.txt :- To change content to uppercase

$cat file1.txt | tr a-z A-Z > output.txt

Output:

```
student@t2:~/nithin$ cat students |tr a-z A-Z >output.txt
student@t2:~/nithin$ cat output.txt
NITHIN JOSE
MANJADIYIL HOUSE
CHIRAKKADAVU
NITHIN RAJAPPAN
PUTHENPURACKAL HOUSE
VAGOMON
student@t2:~/nithin$
```

Result:

Output displayed successfully and CO2 was obtained.

Experiment 4: Date: 07/03/2023

Aim:
Familiarization of Linux Commands

Course Outcome(CO2):
Perform system administration task

Procedure:

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

a. cut -b1 :- Cut by first byte position
$cut -b1 file1.txt
Output:

```
student@t2:~/nithin$ cat > dc_comics
Batman
Superman
Wonderwoman
Aquaman
^Z
[1]+  Stopped                 cat > dc_comics
student@t2:~/nithin$ cut -b1 dc_comics
B
S
W
A
student@t2:~/nithin$
```

b. cut -c3 :- Cut by third character
$cut -c3 dc_comics.txt
Output:

```
student@t2:~/nithin$ cut -c3 dc_comics
t
p
n
u
student@t2:~/nithin$
```

c. cut -d - -f1 file.txt :- Cut by delimiter
$cut -d - -f1 sample.txt
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cat>sample.txt
Nithin -90
Rajappan -95
Philip -98
^Z
[3]+  Stopped                 cat > sample.txt
nithin@DESKTOP-389ELIT:~/addition$ cut -d - -f1 file3.txt
cut: file3.txt: No such file or directory
nithin@DESKTOP-389ELIT:~/addition$ cut -d - -f1 sample.txt
Nithin
Rajappan
Philip
```

d. cut -c :- Select only these characters
$cut -c 1,3,5 sample.txt
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cut -c 1,3,5 sample.txt
Nti
Rjp
Pii
```

2. Paste :- Merge lines of files
$paste sr.txt file1.txt
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cat >file1
10
11
12
13
^Z
[4]+  Stopped                 cat > file1
nithin@DESKTOP-389ELIT:~/addition$ cat >file2
Nithin
Rajappan
Philip
Pranav
^Z
[5]+  Stopped                 cat > file2
nithin@DESKTOP-389ELIT:~/addition$ paste file1 file2
10      Nithin
11      Rajappan
12      Philip
13      Pranav
```

a. paste file1.txt output.txt> output3.txt :- Paste the merged content to new file
$paste file1 file2 > file3
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ paste file1 file2 > file3
nithin@DESKTOP-389ELIT:~/addition$ cat file3
10      Nithin
11      Rajappan
12      Philip
13      Pranav
```

b. paste -d ‘%’ file3.txt output.txt :- Separate the merged parts using a symbol(%)
$paste -d ‘%’ file1 file2
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ paste -d ‘%’ file1 file2
10◆Nithin
11◆Rajappan
12◆Philip
13◆Pranav
```

c. paste -s output.txt :- Display output in a single line
$paste -s file3
Output:

```
nithin@DESKTOP-389ELIT:~/addition$ paste -s file3
10      Nithin  11      Rajappan        12      Philip  13      Pranav
```

3. cp :- Copy the content

a. cp file3.txt sr.txt : -Overwrite existing file

$cp file1 file2

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cp file1 file2
nithin@DESKTOP-389ELIT:~/addition$ cat file2
10
11
12
13
```

b. cp sr.txt output5.txt :- Copy into new file

$cp file3 file4

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ cp file3 file4
nithin@DESKTOP-389ELIT:~/addition$ cat file4
10      Nithin
11      Rajappan
12      Philip
13      Pranav
```

c. cp -r class mca :- Copy directories and subdirectories from existing directory to a new one

$cp -r addition mca

Output:

```
nithin@DESKTOP-389ELIT:~$ cp -r addition mca
nithin@DESKTOP-389ELIT:~$ ls
String  addition  basics  division  mca  modulus  multiplication  old  subtract
```

d. cp newfile class:- Copy file from one directory to another

$cp -r mca class

Output:

```
nithin@DESKTOP-389ELIT:~$ cp -r mca class
nithin@DESKTOP-389ELIT:~$ ls
String  addition  basics  class  division  mca  modulus  multiplication  old  subtract
nithin@DESKTOP-389ELIT:~$ cd class
nithin@DESKTOP-389ELIT:~/class$ ls
add1.sh  add2.sh  add3.sh  add4.sh  file1  file2  file3  file4  sample.txt  students.txt  students2.txt  testfile.txt
```

Result:

Output displayed successfully and CO2 was obtained.

Experiment 5 **Date**: 13/03/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO2):

Perform system administration task

Procedure:

1. read :- Read content of one line of input into a variable
   $read
   echo $REPLY :- To print the input from the default variable
   echo $REPLY :- To print the input from the default variable
   Output:

```
nithin@DESKTOP-389ELIT:~/class$ read
My name is Nithin
nithin@DESKTOP-389ELIT:~/class$ echo $REPLY
My name is Nithin
```

a. read var1 var2 var3 :- To read into specific variables
   $read var1 var2 var3
   $echo "[$var1][$var2][$var3]"
   Output:

```
nithin@DESKTOP-389ELIT:~/class$ read var1 var2 var3
Nithin Jose Manjadiyil
nithin@DESKTOP-389ELIT:~/class$ echo "[$var1][$var2][$var3]"
"[Nithin][Jose][Manjadiyil]"
```

b. read input \ :- To read multiple lines
   $read My \ name is \ Sreerag
   Output:

```
nithin@DESKTOP-389ELIT:~/class$ read
My\
> Name is\
> Nithin Jose
nithin@DESKTOP-389ELIT:~/class$ echo $REPLY
MyName isNithin Jose
```

c. read -p :- Prompt text from user
$read -p “Enter your name”
$echo “my name is $REPLY”
Output:

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

d. read -n :- Specify limit
$read -n 8 -p “Enetr your name”
Output:

```
student@t2:~$ read -n 8 -p "Enter your name"
Enter your nameNithin Jstudent@t2:~$ echo $REPLY
Nithin J
```

e. read -s :-For security. Hides input
$read -s -p “Enter the password: “
Output:

```
student@t2:~$ read -s -p "Enter the Password"
Enter the Passwordstudent@t2:~$ echo "Password is "$REPLY
Password is Manadiyil
```

2. wc :- Word count display number of lines, number of words, number of bytes and file name
$wc profile
Output:

```
student@t2:~$ cat > profile
Nithin Jose
Manjadiyil House
Chirakkadavu
^Z
[1]+  Stopped                 cat > profile
student@t2:~$ wc profile
 3  5 42 profile
```

a. wc -l :- Display number of lines
b. wc -m :-Display number of bytes
c. wc -c :- Display number of characters
d. wc -w :-Display number of words

$ wc -l profile
$ wc -m profile
$ wc -c profile
$ wc -w profile
Output:

```
student@t2:~$ wc -l profile
3 profile
student@t2:~$ wc -m profile
42 profile
student@t2:~$ wc -c profile
42 profile
student@t2:~$ wc -w profile
5 profile
```

e. wc -L :- Displays length of longest line

$ wc -L profile

Output:

```
student@t2:~$ wc -L profile
16 profile
```

3. more :- It is similar to cat to display the content. The difference is that in case of larger files, cat command output will scroll off your screen while more command display output one screenful at a time.

$ more longtext.txt

Output:

```
    The time had come for Nancy to say goodbye. She had been dreading this moment for a good
six months, and it had finally arrived despite her best efforts to forestall it. No matter ho
w hard she tried, she couldn't keep the inevitable from happening. So the time had come for a
 normal person to say goodbye and move on. It was at this moment that Nancy decided not to be
 a normal person. After all the time and effort she had expended, she couldn't bring herself
to do it.
    Debbie put her hand into the hole, sliding her hand down as far as her arm could reach. S
--More--(7%)
```

a. more +15 file.txt :- Will display content after the specified number of lines

$ more +22 lontext.txt

Output:

```
student@t2:~$ more +22 longtext.txt
    She was in a hurry. Not the standard hurry when you're in a rush to get someplace, but a franti
c hurry. The type of hurry where a few seconds could mean life or death. She raced down the road ig
noring speed limits and weaving between cars. She was only a few minutes away when traffic came to
a dead standstill on the road ahead.
```

b. more +/pattern file.txt :- Search and navigate towards a particular string and view all the instances.

$ more +/someplace longtext.txt

Output:

```
student@t2:~$ more +/someplace longtext.txt

...skipping
    Time is all relative based on age and experience. When you are a child an hour is a long time to wait
but a very short time when that’s all the time you are allowed on your iPad. As a teenager time goes faste
r the more deadlines you have and the more you procrastinate. As a young adult, you think you have forever
 to live and don’t appreciate the time you spend with others. As a middle-aged adult, time flies by as you
 watch your children grow up. And finally, as you get old and you have fewer responsibilities and fewer de
mands on you, time slows. You appreciate each day and are thankful you are alive. An hour is the same amou
nt of time for everyone yet it can feel so different in how it goes by.
    She was in a hurry. Not the standard hurry when you're in a rush to get someplace, but a frantic hurry
```

c. more -p file.txt :- Clear the whole screen and then display the text.

$ more -p longtext.txt

Output:

```
here was little doubt that the bridge was unsafe. All one had to do was look at it to know that with certainty. Yet Bob didn't see anot
her option. He may have been able to work one out if he had a bit of time to think things through, but time was something he didn't hav
e. A choice needed to be made, and it needed to be made quickly.
He hid under the covers hoping that nobody would notice him there. It really didn't make much sense since it would be obvious to anyone
 who walked into the room there was someone hiding there, but he still held out hope. He heard footsteps coming down the hall and stop
in front in front of the bedroom door. He heard the squeak of the door hinges and someone opened the bedroom door. He held his breath w
aiting for whoever was about to discover him, but they never did.

(END)
```

d. more -d file.txt :- Helps the user to navigate according to instructions, [space to continue and ‘q’ to quit]

$ more -d longtext.txt

Output:

```
    Stormi is a dog. She is dark grey and has long legs. Her eyes are expressive and are able to let her h
umans know what she is thinking. Her tongue is long, pink, and wet. Her long legs allow her to sprint afte
r other dogs, people or bunnies. She can be a good dog, but also very bad. Her tail wags when happy or exc
ited and hides between her back legs when she is bad. Stormi is a dog I love.
    It all started with the computer. Had he known what was to follow, he would have never logged on that
--More--(27%)[Press space to continue, 'q' to quit.]
```

Result:

Output displayed successfully and CO2 was obtained.

Experiment 6: Date: 14/03/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO2):

Perform system administration task

Procedure:

1. grep :- Filtering and Searching content easily

   $ grep 90 marks

   Output:

```
mca@S47:~/Nithin$ grep marks | grep 90
^Z
[2]+  Stopped                 grep --color=auto marks | grep --color=auto 90
mca@S47:~/Nithin$ grep 90 marks
```

a. grep -i :- Case insensitive search of a particular content

   $ grep -i English marks

   Output:

```
mca@S47:~/Nithin$ grep -i English marks
English 50
```

b. grep -v :- Exclude that content during search

   $ grep -v English marks

   Output:

```
mca@S47:~/Nithin$ grep -v 80 marks
English 50
Maths 70
Programming 90
Malayalam 90
```

c. grep -A1 :- Specific content and one line after the content

   $ grep -A1 English marks

   Output:

```
mca@S47:~/Nithin$ grep -A1 English marks
English 50
Maths 70
```

d. grep -B1 :- Specific content and one line before the content

   $ grep -B1 English marks

   Output:

```
mca@S47:~/Nithin$ grep -B1 Science marks
Maths 70
Science 80
```

e. grep -C1 :- Specific content and one lone before and after the content.

$ grep -C1 English marks

Output:

```
mca@S47:~/Nithin$ grep -C1 Science marks
Maths 70
Science 80
Programming 90
```

2. head :- Used to display the first content of the file(Top 10 lines by default)

$ head demo

Output:-

```
mca@S47:~/Nithin$ head demo
1
2
3
4
5
6
7
8
9
10
```

a. head -number filename:- Specific number of lines

$ head -5 demo

Output:

```
mca@S47:~/Nithin$ head -5 demo
1
2
3
4
5
```

3. tail:- Used to display last contents of a file(last 10 by default)

$ tail demo

Output:

```
mca@S47:~/Nithin$ tail demo
7
8
9
10
11
12
13
1
15
```

a. tail -number filename :- Specific number of content from last
$ tail -5 demo
Output:

```
mca@S47:~/Nithin$ tail -5 demo
12
13
1
15
```

4. mv :- move from one location to another or it can be used to rename a file. Content will be overwritten.
$ mv demo marks
Output:-

```
mca@S47:~/Nithin$ mv demo marks
mca@S47:~/Nithin$ cat marks
1
2
3
4
5
6
7
8
9
10
11
12
13
1
15
```

a. mv -b :- To take backup of a file while moving.
$ mv -b marks profile
Output:

```
mca@S47:~/Nithin$ mv -b marks profile
mca@S47:~/Nithin$ ls
Malayalam  profile  profile~
mca@S47:~/Nithin$ cat profile~
Nithin Jose
Manjadiyil House
Chirakkadavu PO
```

b. mv -i :- Prompt confirmation from user before overwriting.
$ mv -i profile profile1
Output:

```
mca@S47:~/Nithin$ mv -i profile profile1
mv: overwrite 'profile1'? y
mca@S47:~/Nithin$ ls
Malayalam  profile~  profile1
```

Result:

Output displayed successfully and CO2 was obtained.

Experiment 7

Date: 20/03/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO2):

Perform system administration task

Procedure:

1. expr :- Evaluate the given expression and display the output.

   $expr 12 + 8

   Output:

```
student@t2:~$ expr 12 + 8
20
student@t2:~$ expr 12 - 8
4
student@t2:~$ expr 12 / 4
3
student@t2:~$ expr 12 \* 2
24
```

a. expr x + y :- Add two variables obtained through read

   $read x

   $read y

   $expr $x + $y

   Output:

```
student@t2:~$ read x
25
student@t2:~$ read y
35
student@t2:~$ expr $x + $y
60
```

2. df :- Get a report on disk utilization of the system

   $df

   Output:

```
student@t2:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             3950480        0   3950480   0% /dev
tmpfs             797752     1808    795944   1% /run
/dev/sda6      143074460 25052284 110681584  19% /
tmpfs            3988752        0   3988752   0% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            3988752        0   3988752   0% /sys/fs/cgroup
```

3. du :- check how much space a file or directory in a given directory
   $du dc_comics
   Output:

```
student@t2:~/nithin$ du dc_comics
4       dc_comics
```

4. sudo :- superuser do

a. sudo useradd user :- Add new user
   $sudo useradd nithin
   Output:

```
mca@t2:~$ sudo useradd nithin
[sudo] password for mca:
```

b. sudo passwd user :- Update password of the user
   $sudo passwd nithin
   Output:

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

c. sudo groupadd -g identifier name:- To create new group
   $sudo groupadd -g 765 mcastd

```
mca@t2:~$ sudo groupadd -g 765 mcastudent
```

d. sudo usermod -G name user :- Add users to group
   $sudo usermod -G mcastd nithin

```
mca@t2:~$ sudo usermod -G mcastudent nithin
```

e. id user :- Details on group name and numeric id of particular user.
   $id nithin
   Output:

```
mca@t2:~$ id nithin
uid=2004(nithin) gid=2006(nithin) groups=2006(nithin),765(mcastudent)
```

   compgen -g :- Display all the groups created
   $compgen -g
   Output:

```
mca@t2:~$ compgen -g
root
daemon
bin
sys
adm
```

5. chmod :- Used to change the access permissions of files and directories. It stands for change mod namely, read(r), write(w), execute(x)

a. chmod -wx file :- deny permission to write and execute for file

$chmod -wx nithin

Output:

```
mca@t2:~$ chmod -wx nithin
mca@t2:~$ cat >> nithin
bash: nithin: Permission denied
```

b. chmod +wrx file :- give permission to write, read and execute for a file
$chmod +wrx file

Output:

```
mca@t2:~$ chmod +rwx nithin
mca@t2:~$ cat >> nithin
Kottayam
^Z
[2]+  Stopped                 cat >> nithin
```

6. sudo chown :- Used to change ownership of a file or directory for a user or a group. It stands for change owner.
$sudo chown nithin address.txt
Output:

```
mca@t2:~$ sudo chown nithin address.txt
[sudo] password for mca:
mca@t2:~$ chmod +rwx address.txt
chmod: changing permissions of 'address.txt': Operation not permitted
mca@t2:~$ ls -l address.txt
-rw-rw-r-- 1 nithin mca 29 Mar 20 12:07 address.txt
```

7. sudo userdel user :- Delete user
$sudo userdel nithin

Output:

```
mca@t2:~$ sudo userdel nithin
[sudo] password for mca:
mca@t2:~$ sudo userdel nithin
userdel: user 'nithin' does not exist
```

8. sudo groupdel name :- Delete group
$sudo groupdel mcastudent

Output

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

Result:

Output displayed successfully and CO2 was obtained.

Experiment 8 **Date:** 21/03/2023

Aim:
Familiarization of Linux Commands

Course Outcome(CO2):
Perform system administration task

Procedure:

1. ip addr:- Get ip address of the system
   $ip addr
   Output:

```
mca@t2:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gr
oup default 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 s
tate UP group default qlen 1000
    link/ether 40:16:7e:ac:a9:4f brd ff:ff:ff:ff:ff:ff
    inet 192.168.6.40/24 brd 192.168.6.255 scope global noprefixroute
enp3s0
       valid_lft forever preferred_lft forever
    inet6 fe80::e932:14d1:60e7:cfef/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
   Output:

```
mca@t2:~$ ssh mca@192.168.6.39
ssh: connect to host 192.168.6.39 port 22: Connection refused
```

a. sudo apt-get install openssh -server :- Update port
b. sudo ufw allow 22
   $sudo ufw allow 22
   Output:

```
mca@t2:~$ sudo ufw allow 22
[sudo] password for mca:
Skipping adding existing rule
Skipping adding existing rule (v6)
```

c. $ssh mca@192.168.6.28

Output:

```
mca@t2:~$ ssh mca@192.168.6.28
mca@192.168.6.28'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.
Last login: Tue Mar 21 14:35:48 2023 from 192.168.6.40
```

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

$ssh-keygen

Output:

```
mca@t2:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mca/.ssh/id_rsa): key.txt
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key.txt
Your public key has been saved in key.txt.pub
The key fingerprint is:
SHA256:MA21h8EbYCcUtN9OfVkdHioRk4OymvjlGdO1PnpwAnw mca@t2
The key's randomart image is:
+---[RSA 3072]----+
|     oO=+ .+o  o |
|     . Bo= oo o +|
|      = *o.... .o|
|       B.E o.  o |
|    . o S + o o  |
|   . o + * o .   |
|    . o + *      |
|     . o   +     |
|         .o .    |
+----[SHA256]-----+
```

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

a. $ps

Output:

```
mca@t2:~$ ps
    PID TTY          TIME CMD
  10499 pts/12   00:00:00 bash
  10615 pts/12   00:00:00 ps
```

b. ps -u :- Display all running processes of a particular user
$ps -u mca

Output:

```
mca@t2:~$ ps -u mca
    PID TTY          TIME CMD
   1426 ?        00:00:00 systemd
   1427 ?        00:00:00 (sd-pam)
   1442 ?        00:04:54 pulseaudio
   1444 ?        00:00:00 tracker-miner-f
   1447 ?        00:00:00 dbus-daemon
   1452 ?        00:00:00 gnome-keyring-d
   1459 ?        00:00:00 gvfsd
   1464 ?        00:00:00 gvfsd-fuse
   1483 ?        00:00:00 gvfs-udisks2-vo
   1489 ?        00:00:00 gvfs-mtp-volume
   1493 ?        00:00:00 gvfs-goa-volume
   1497 ?        00:00:00 goa-daemon
   1504 ?        00:00:00 goa-identity-se
   1509 ?        00:00:00 gvfs-gphoto2-vo
   1514 ?        00:00:00 gvfs-afc-volume
   1521 tty2     00:00:00 gdm-x-session
```

c. ps -C :- Specific process
$ps -C firefox

Output:

```
mca@t2:~$ ps -C firefox
    PID TTY          TIME CMD
   2762 ?        00:13:35 firefox
```

d. ps -f –p PID :- List the process by id
$ps -f -p 2762

Output:

```
mca@t2:~$ ps -f -p 2762
UID          PID    PPID  C STIME TTY          TIME CMD
mca         2762    1426  9 13:37 ?        00:13:45 /usr/lib/firefox/f
```

Result:

Output displayed successfully and CO2 was obtained.

Experiment 9

Date: 28/03/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO4):

Write shell scripts required for system administration

Procedure:

1. Shell script to display date:
   $vi filename.sh :- Open Editor by creating a shell script file.
   Press ‘i’ to INSERT
   #!/bin/bash :- To indicate the shell used date
   Press ‘Esc’ to end INSERT
   :wq! :- To save and quit
   chmod +x filename.sh :- To give execution permission
   ./filename.sh :- To execute shell script
   Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi date.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x date.sh
nithin@DESKTOP-389ELIT:~/basics$ ./date.sh
Sun Apr 16 20:58:42 IST 2023
```

2. Shell script to display your name:
   $vi filename.sh Press ‘i’ to INSERT #!/bin/bash
   echo “What is your name?” read name
   echo “My name is $name” Press ‘Esc’ to end INSERT
   :wq!
   chmod +x filename.sh
   ./filename.sh
   Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi name.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x name.sh
nithin@DESKTOP-389ELIT:~/basics$ ./name.sh
What is your name?
Nithin
My name is Nithin
```

3. Multiple Commands (ls, pwd, date, mkdir) in Shell Script:
   $vi filename.sh Press ‘i’ to INSERT #!/bin/bash
   date
   ls
   pwd
   mkdir
   file6
   ls
   Press ‘Esc’ to end INSERT
   :wq!
   chmod +x filename.sh
   ./filename.sh
   Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi Multicommand.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x Multicommand.sh
nithin@DESKTOP-389ELIT:~/basics$ ./Multicommand.sh
Sun Apr 16 21:14:40 IST 2023
Multicommand     arithmetics.sh  file5          ls                    sample
Multicommand.sh  counts.sh       gratest2.sh    name.sh
arith2.sh        date.sh         greatest3.sh   oddeven.sh
arith3.sh        file1           logicalop1.sh  positivenegative.sh
/home/nithin/basics
```

4. Shell script to demonstrate variables
   $vi filename.sh Press ‘i’ to INSERT #!/bin/bash
   echo “Enter your name: “ read name
   echo “Your name is $name”
   echo “File Name: $0”
   echo “First Parameter: $1”
   echo “Second Parameter: $2”
   echo “Quoted Values: $@”
   echo “Quoted Values: $*”
   echo “Total Number of Parameters: $#” Press ‘Esc’ to end INSERT
   :wq!
   chmod +x filename.sh
   ./filename.sh
   Output:

```
nithin@DESKTOP-389ELIT:~/old$ vi specialvar.sh
nithin@DESKTOP-389ELIT:~/old$ chmod +x specialvar.sh
nithin@DESKTOP-389ELIT:~/old$ ./specialvar.sh MCA AJCE KPLY
Filename : ./specialvar.sh
Filrst Parameter : MCA
Second Parameter : AJCE
Quoted Values : MCA AJCE KPLY
Quoted values : MCA AJCE KPLY
Total Parameters : 3
```

5. Shell script to count lines and words in a file
   readlink -f filename :- Get path of required file
   $vi filename.sh Press ‘i’ to INSERT #!/bin/bash
   file_path = “/home/Reqfilename.sh” countlines = `wc –lines < $file_path` countwords = `wc –words < $file_path` echo “Number of lines: $countlines” echo “Number of words: $countwords”
   Press ‘Esc’ to end INSERT
   :wq!
   chmod +x filename.sh
   ./filename.sh
   Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi counts.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x counts.sh
nithin@DESKTOP-389ELIT:~/basics$ ./counts.sh
line count is : 3
word lines is: 5
```

6. Shell script to display array index
   $vi filename.sh Press ‘i’ to INSERT #!/bin/bash
   Name=("name1" "name2" "name3" "name4")
   echo "First Index: ${Name[0]}"
   echo "Second Index: ${Name[1]}"
   Press ‘Esc’ to end INSERT
   :wq!
   chmod +x filename.sh
   ./filename.sh
   Output:

```
nithin@DESKTOP-389ELIT:~/old$ vi array.sh
nithin@DESKTOP-389ELIT:~/old$ chmod +x array.sh
nithin@DESKTOP-389ELIT:~/old$ ./array.sh
First Index: name1
Second Index: name2
```

Result:

Output displayed successfully and CO4 was obtained.

Experiment 10

Date: 03/04/2023

Aim:

Familiarization of Linux Commands

Course Outcome(CO4):

Write shell scripts required for system administration

Procedure:

1. Shell script to add two number: vi filename.sh

Press ‘i’ to INSERT

```
#!/bin/bash
val=`expr 10 + 10`
echo "total is $val"
```

Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ vi add1.sh
nithin@DESKTOP-389ELIT:~/addition$ chmod +x add1.sh
nithin@DESKTOP-389ELIT:~/addition$ ./add1.sh
total is 20
```

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.

vi filename.sh Press ‘i’ to INSERT #!/bin/bash

```
read -p "Enter first number " num1
read -p "Enter second number " num2
sum=$(($num1 + $num2))
echo "sum is: $sum"
```

Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ vi add2.sh
nithin@DESKTOP-389ELIT:~/addition$ chmod +x add2.sh
nithin@DESKTOP-389ELIT:~/addition$ ./add2.sh
Enter first number 23
Enter second number 25
sum is: 48
```

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

   vi filename.sh Press ‘i’ to #!/bin/bash

   a=10

   b=20

   sum=$(($a + $b))
   echo "sum is: $sum"Press ‘Esc’ to end INSERT

   :wq!
   chmod +x filename.sh
   ./filename.sh num1 num2

   Output:

```
nithin@DESKTOP-389ELIT:~/addition$ vi add3.sh
nithin@DESKTOP-389ELIT:~/addition$ chmod +x add3.sh
nithin@DESKTOP-389ELIT:~/addition$ ./add3.sh
sum is: 30
```

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 vi filename.sh Press ‘i’ to INSERT#!/bin/bash sum=$(($1 + $2)) echo "sum is $sum"
   Press ‘Esc’ to end INSERT

   :wq!
   chmod +x filename.sh
   ./filename.sh num1 num2

   Output:

```
nithin@DESKTOP-389ELIT:~/addition$ vi add4.sh
nithin@DESKTOP-389ELIT:~/addition$ chmod +x add4.sh
nithin@DESKTOP-389ELIT:~/addition$ ./add4.sh 12 12
sum is 24
```

5. Shell script to demonstrate Arithmetic operators (addition, subtraction, multiplication, division, modulus, increment, decrement) by taking user input and store to another variable
   vi filename.sh Press ‘i’ to INSERT
   #!/bin/bash

   read -p "Enter the First number: " num1 read -p "Enter the Second number: " num2 sum=$(($num1 + $num2))
   prd=$(($num1 * $num2)) diff=$(($num1 - $num2)) quo=$(($num1 / $num2)) rem=$(($num1 % $num2)) echo "Sum : $sum"
   echo "Product : $prd" echo "Difference : $diff" echo "Quotient : $quo" echo "Remainder : $rem" if [$num1 == $num2] then
   echo "$num1 is equal to $num2"
   fi
   if [$num1 != $num2] then
   echo "$num1 is not equal to $num2"
   fi

((++num1))
echo "Increment operator on first number: $num1" ((--num2))
echo "Decrement operator on second number: $num2" Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/addition$ vi add5.sh
nithin@DESKTOP-389ELIT:~/addition$ chmod +x add5.sh
nithin@DESKTOP-389ELIT:~/addition$ ./add5.sh
Enter the First number: 25
Enter the Second number: 25
Sum : 50
Product : 625
Difference : 0
Quotient : 1
Remainder : 0
25 is equal to 25
Increment operator on first number: 26
Decrement operator on second number: 24
```

Result:

Output displayed successfully and CO4 was obtained.

Experiment 11 **Date:** 04/04/2023

Aim:
Familiarization of Linux Commands

Course Outcome(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

```
vi filename.sh   Press 'i' to INSERT  #!/bin/bash
read -p "Enter First : " i
read -p "Enter Second : " j
if (( $i == $j ))
then
        echo "Both Numbers are equal"
else
        echo "Both Numbers are different"
fi
if (( $i != $j ))
then
      echo "Both Numbers are different"
else
      echo "Both Numbers are Equal"
fi
if (( $i < $j ))
then
        echo "First is less than Second Number"
else
        echo "First is not less than Second Number"
fi
if (( $i <= $j ))
then
        echo "First is less than or equal to Second Number"
else
      echo "First is not less than or equal to Second Number"
fi
```

```
        if (( $i > $j ))
        then
            echo "First is greater than Second Number"
        else
            echo "First is not greater than Second Number"
        fi
        if (( $i >= $j ))
        then
            echo "First is greater than or equal to Second Number"
        else
            echo "First is not graeter than or equal to Second Number"
        fi
echo "Invalid Inputs"
```

Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi arith2.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x arith2.sh
nithin@DESKTOP-389ELIT:~/basics$ ./arith2.sh
Enter First : 25
Enter Second : 36
Both Numbers are different
Both Numbers are different
First is less than Second Number
First is less than or equal to Second Number
First is not greater than Second Number
First is not graeter than or equal to Second Number
Invalid Inputs
```

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)
   vi filename.sh Press ‘i’ to INSERT

```
#!/bin/bash
a=12
b=10
if [ $a -eq $b ]
then
    echo "$a -eq $b : a is equal to b"
else
    echo "$a -eq $b : a is not equal to b"
fi
```

```
if [ $a -ne $b ]
then
        echo "$a -ne $b : a is not equal to b"
else
      echo "$a -ne $b : a is equal to b"
fi

if [ $a -gt $b ]
then
        echo "$a -gt $b : a is greater than to b"
else
      echo "$a -gt $b : a is not graeter than b"
fi

if [ $a -lt $b ]
then
        echo "$a -lt $b : a is less than b"
else
      echo "$a -lt $b : a is not less than b"
fi

if [ $a -ge $b ]
then
        echo "$a -ge $b : a is greater than or equal to b"
else
      echo "$a -ge $b : a is not greater than or equal to b"
fi

if [ $a -le $b ]
then
        echo "$a -le $b : a is less than or equal to b"
else
      echo "$a -le $b : a is not less than or equal to b"
fi
```

Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi arith3.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x arith3.sh
nithin@DESKTOP-389ELIT:~/basics$ ./arith3.sh
12 -eq 10 : a is not equal to b
12 -ne 10 : a is not equal to b
12 -gt 10 : a is greater than to b
12 -lt 10 : a is not less than b
12 -ge 10 : a is greater than or equal to b
12 -le 10 : a is not less than or equal to b
```

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

```
#!/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 "Atleast one of the input is true"
else
        echo "None of the input are true"
fi

if(( ! $a == "true" ))
then
        echo "a was initially false"
```

```
else
        echo "a was initially true"
fi
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi logicalop1.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x logicalop1.sh
nithin@DESKTOP-389ELIT:~/basics$ ./logicalop1.sh
Enter a : true
Enter b : true
Both are true
Atleast one of the input is true
a was initially true
```

4. Write a shell script to check if a number is even or odd. vi filename.sh
Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter the Number to check : " num
if(( $num == 0 ))
then
    echo "The Entered Number is Zero"
else
    if(( num % 2 == 0 ))
    then
        echo "$num is even"
    else
        echo "$num is odd"
    fi
fi
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi oddeven.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x oddeven.sh
nithin@DESKTOP-389ELIT:~/basics$ ./oddeven.sh
Enter the Number to check : 25
25 is odd
nithin@DESKTOP-389ELIT:~/basics$
```

5. Write a shell script to check whether a number is positive or negative vi filename.sh
Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter the Number to check : " num
if(( $num > 0 ))
then
    echo "The Entered Number is Positive"
elif(( $num < 0 ))
then
    echo "The entered Number is Negative"
elif(( $num ==0 ))
then
    echo "The Eneterd Number is Zero"
else
    echo "Please enter a valid Number"
fi
```

./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi positivenegative.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x positivenegative.sh
nithin@DESKTOP-389ELIT:~/basics$ ./positivenegative.sh
Enter the Number to check : -18
The entered Number is Negative
```

6. Write a shell script to find the greatest of two numbers vi filename.sh
Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter the Number 1 : " num1
read -p "Enter the Number 2 : " num2
```

```
        if(( $num1 > $num2 ))
        then
                echo "$num1 is greater than $num2"
        elif(( $num2 > $num1 ))
        then
                echo "$num2 is greater than $num1"
        else
                echo "$num1 and $num2 are equal"
fi
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/basics$ ./gratest2.sh
Enter the Number 1 : 25
Enter the Number 2 : 26
26 is greater than 25
```

7. Write a shell script to find the greatest of three numbers vi filename.sh
Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter Number 1 : " num1
read -p "Enter Number 2 : " num2
read -p "Enter Number 3 : " num3

if(( $num1 > $num2 ))
then
	if(( $num1 > $num3 ))
	then
		echo "$num1 is the greatest of three numbers entered"
	else(( $num3 > $num1 ))
		echo "$num3 is the greatest of three numbers entered"
	fi
elif(( $num2 > $num1 ))
then
	if(( $num2 > $num3 ))
	then
		echo "$num2 is the greatest of three numbers entered"
	else(( $num3 < $num2 ))
		echo "$num3 is the greatest of three numbers entered"
	fi
else
	if(( $num1 > $num3 ))
	then
		echo "Both Number1 and Number2 are the greatest Numbers"
	else(( $num1 < $num3 ))
```

```
                echo "$num3 is the greatest among three numbers"
            fi
        fi
```

:wq!
chmod +x filename.sh
./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/basics$ vi greatest3.sh
nithin@DESKTOP-389ELIT:~/basics$ chmod +x greatest3.sh
nithin@DESKTOP-389ELIT:~/basics$ ./greatest3.sh
Enter Number 1 : 24
Enter Number 2 : 25
Enter Number 3 : 26
26 is the greatest of three numbers entered
```

Result:

Output displayed successfully and CO4 was obtained.

Experiment 12 **Date:** 11/04/2023

Aim:
Familiarization of Linux Commands

Course Outcome(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
   vi filename.sh Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter the string 1: " a
read -p "Enter the string 2: " b

if [ $a == $b ]
then
        echo "Both strings are the same"
else
        echo "Both strings are not the same"
fi

if [ $a != $b ]
then
        echo "Both Strings are not the same"
else
        echo "Both strings are the same"
fi

if [ -z $a ]
then
        echo "-z $a :Entered string is null"
else
        echo "-z $a :Entered string was not null string"
fi
```

```
if [ -n $a ]
then
        echo "-n $a : Entered string is not null"
else
        echo "-n $a : Entered string is null"
fi

if [ $a ]
then
        echo " $a : Entered string is not null"
else
        echo " $a : Entered string is null"
fi
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/String$ vi strings.sh
nithin@DESKTOP-389ELIT:~/String$ chmod +x strings.sh
nithin@DESKTOP-389ELIT:~/String$ ./strings.sh
Enter the string 1: nithin
Enter the string 2: manjadiyil
Both strings are not the same
Both Strings are not the same
-z nithin :Entered string was not null string
-n nithin : Entered string is not null
 nithin : Entered string is not null
```

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

vi filename.sh Press ‘i’

```
#!/bin/bash
read -p "Enter first no: " a
read -p "Enter second no: " b
bitwiseAND=$(( num1&num2 ))
echo "BITWISE AND : $bitwiseAND"
bitwiseOR=$(( a|b ))
echo "BITWISE OR : $bitwiseOR"
bitwiseXOR=$(( a^b ))
echo "BITWISE XOR : $bitwiseXOR"
complement=$(( ~a ))
echo "BITWISE COMPLEMENT : $complement"
leftshift=$(( a<<1 ))
echo "LEFT SHIFT : $leftshift"
rightshift=$(( b>>1 ))
echo "RIGHT SHIFT : $rightshift"
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/String$ ./bitwise.sh
Enter first no: 10
Enter second no: 12
BITWISE AND : 0
BITWISE OR : 14
BITWISE XOR : 6
BITWISE COMPLEMENT : -11
LEFT SHIFT : 20
RIGHT SHIFT : 6
```

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

vi filename.sh

Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter file name: " f1
if [ -e $f1 ]
then
echo "$f1 exist"
else
echo "$f1 does not exist"
fi
if [ -s $f1 ]
then
echo "$f1 is not empty"
else
echo "$f1 is empty"
fi
if [ -r $f1 ]
then
echo "$f1 has read permission"
else
echo "$f1 does not have read permission"
fi
if [ -x $f1 ]
then
echo "$f1 has execute permission"
else
echo "$f1 does not have execute permission"
```

```
fi
if [ -w $f1 ]
then
echo "$f1 has write permission"
else
echo "$f1 does not have write permission"
fi
```

Output:

```
nithin@DESKTOP-389ELIT:~/String$ vi FileT.sh
nithin@DESKTOP-389ELIT:~/String$ chmod +x FileT.sh
nithin@DESKTOP-389ELIT:~/String$ ./FileT.sh
Enter file name: sample
sample exist
sample is not empty
sample has read permission
sample does not have execute permission
sample has write permission
```

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

vi filename.sh

Press ‘i’ to INSERT

```
#!/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
```

Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:

```
nithin@DESKTOP-389ELIT:~/String$ vi FileT2.sh
nithin@DESKTOP-389ELIT:~/String$ chmod +x FileT2.sh
nithin@DESKTOP-389ELIT:~/String$ ./FileT2.sh
Enter the first number: 12
Enter the second number: 12
Both numbers are equal
```

5. Shell Script to check the range of a number if numbers using else if ladder
vi filename.sh
Press ‘i’ to INSERT

```
#!/bin/bash
read -p "Enter the number(b/w 0-50): " num1
if (( $num1 >= 0 && $num1 <= 10 ))
then
echo "$num1 is between 0 and 10"
elif (( $num1 >= 11 && $num1 <= 20 ))
then
echo "$num1 is between 10 and 20"
elif (( $num1 >= 21 && $num1 <= 30 ))
then
echo "$num1 is between 20 and 30"
elif (( $num1 >= 31 && $num1 <= 40 ))
then
echo "$num1 is between 30 and 40"
elif (( $num1 >= 41 && $num1 <= 50 ))
then
echo "$num1 is between 40 and 50"
fi
```

Press ‘Esc’ to end INSERT
:wq!
chmod +x filename.sh
./filename.sh
Output:

```
nithin@DESKTOP-389ELIT:~/String$ vi FileT3.sh
nithin@DESKTOP-389ELIT:~/String$ chmod +x FileT3.sh
nithin@DESKTOP-389ELIT:~/String$ ./FileT3.sh
Enter the number(b/w 0-50): 25
25 is between 20 and 30
```