**Experiment No.: 3**

**Aim:**

Familiarisation of Linux commands

**CO2:**

**Procedure:**

1. Pwd - Print working directory.

To print the name of the working directory.

$pwd

**output:**

****

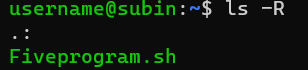
1. ls - To list directory contents

$ls

* 1. ls -R - to list subdirectories recursively

$ls -R

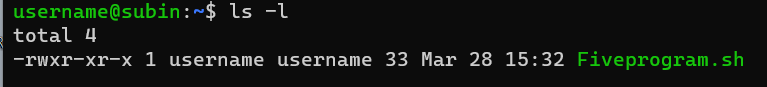
**output:**



* 1. ls -l - use a long listing format

&ls -l

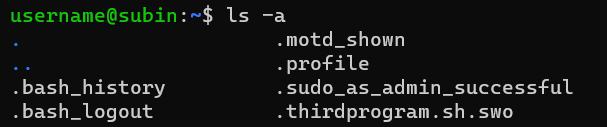
**output:**



* 1. ls -a -list a hidden files

$ls -a

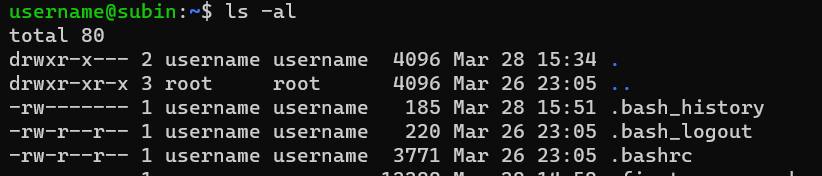
**output:**



* 1. ls -al - list the files and directories with detailed information such as permissions,size and owner

$ls -al

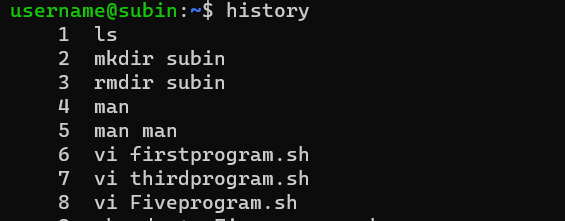
**output:**



1. History - Review all previously executed command right from the shell.

$history

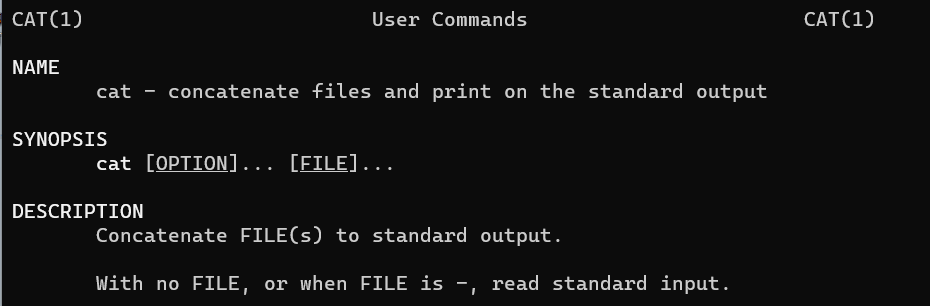
**output:**



1. Man - an interface to the syatem reference manuals

$ man cat

**output:**



1. Cd - Change directory

$cd demo

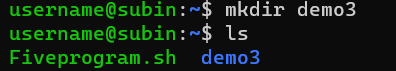
**output:**



1. mkdir - Make Directory

$mkdir demo3

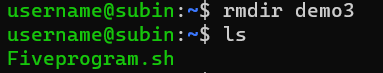
**output:**



1. Rmdir - Remove empty directory

$rmdir demo3

**output:**



1. Touch - create empty file

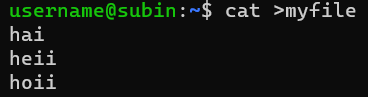
$ touch newfile.txt

1. Cat - concatenate files and print on the standard output

9.1) Cat >myfile -create a new file named myfile and allow to write content to it.

$cat>myfile

**output:**



9.2) cat >> - Cat command can append the contents of one file to the end of another file.

$cat>>myfile2

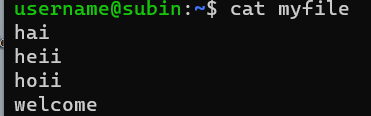
**output:**



9.3)cat – display the content in the file

$cat myfile

**output:**



**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

**Experiment No.: 4**

**Aim:**

Familiarisation of Linux commands

**CO2:**

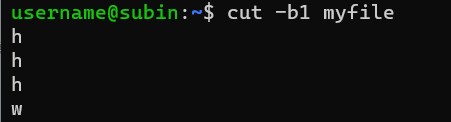
**Procedure:**

1.cut -for cutting out the section from each line of files and writing result to standard

Output.

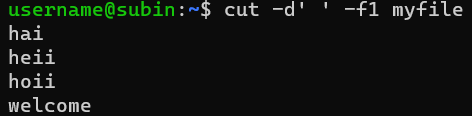
* 1. cut -b1 – cut by byte

$cut -b1 file



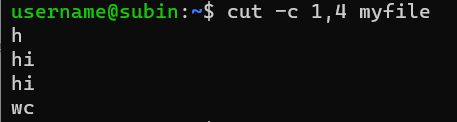
* 1. cut -d --cut by delimiter

$cut -d’ ‘ -f1 file2



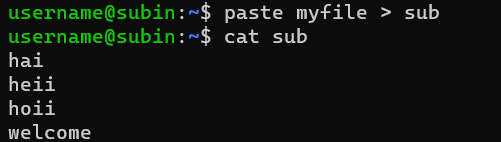
* 1. cut -c –cut by character

$cut -c 1,2 file2



2) paste -paste command is used to join file horizontally.

$paste file2 > sub



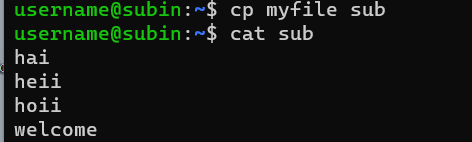
2.1) paste -s -display content in one single line.

$paste -s file2



3)copy – to copy the content from one file to another.

$cp file sub



3.1) cp -r -use to copy the directory along with its sub directory

$cp -r subin new

**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

**Experiment No.: 5**

**Aim:**

Familiarisation of Linux commands

**CO1:**

Perform system administration tasks

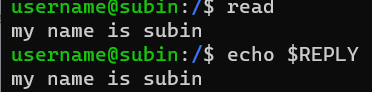
**Procedure:**

1)read-To read the content of the line

$read

$echo REPLY

Output:

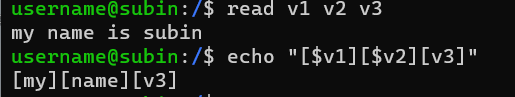


1.1)read var1 var2 var3-To read into specific variable

$read var1 var2 var3

$echo “[$var1][$var2][$var3]”

Output:



1.2)$read -To provide output in multiple lines

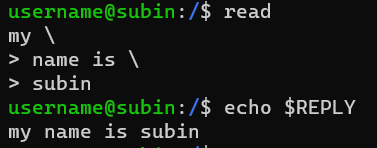
My\

name is\

subin

echo $REPLY

Output:

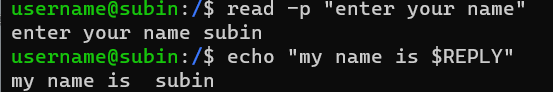


1.3)read -p ->to prompt text from user

$read -p “Enter your name”

$echo my name is $REPLY”

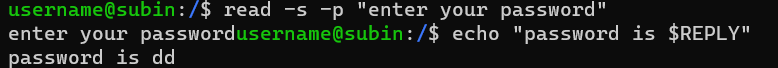
Output:



1.4) read -s : For security

$read -s -p “Enter the password:”

Output:



2. wc - Word count display number of lines,number of lines,number of words,number of bytes in file

$wc fiprofile

Output:



2.1)wc -l : Display number of lines

$wc -l filename.txt

Output:



2.2)wc -m filename.txt -display number of characters

$wc -m profile



2.3)$wc -c profile – display number of bytes



2.4)Wc -L --longest line

$wc -L profile



3.more:it is similar to cat to display the content only diff is that in case of larger files,cat command output will scroll off your screen while more command display output one screenfull at a time

$more content.txt

Output:

Text

Description automatically generated

3.1)more +2 filename.txt :will display the content after specified number of lines

$more +2 content.txt

Output:

search: To search the string inside your text document,you can view all the instances by navigating through the result

$more +/string name content.txt

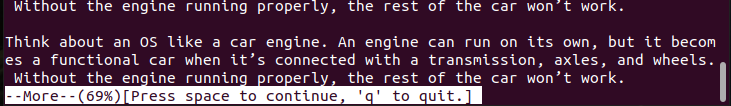
Output:

Text

Description automatically generated

more -d filename.txt : help the user to navigate according to instructions

$more -d content.txt



**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

**Experiment No.: 6**

**Aim:**

Familiarisation of Linux commands

**CO2:**

1. grep -: grep is used to file the content which make search easy.

grep filename

$grep mark1 |grep 67

**output:**



1.1)grep -v –inverted search.All content expect search content display.

$grey -v 56 mark1

**output:**



1.2) grep -A1 –view the content along with online after .

$grep -A1 english mark1

**output:**



1.3)grep -B1 –view the content along with online before.

$grep -B1 maths mark1

**output:**

Text

Description automatically generated

1.4) grep -C1 –Get the content along with one line after and before.

$grep -C1 maths mark1

**output:**

Text

Description automatically generated

1.5) grep -i –case insensitive search.

$grep -i 98 mark1

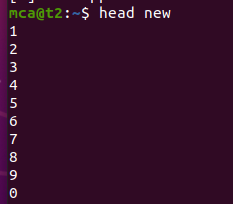
**output:**



1. head – Head command is used to display the content.By default 10 line.

$head new

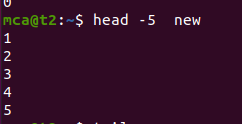
**output:**



2.1)head -5 –To get the first five lines.

$head -5 new

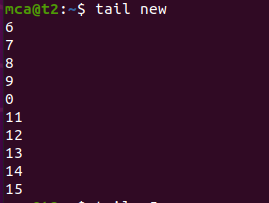
**output:**



1. tail –Used to display last line .By default 10 line.

$tail new

**output:**



3.1)tail -5 –To get the last five lines.

$tail -5 new

**output:**

Text

Description automatically generated

1. mv -moving one file to another by replacing it.
2. **output:**

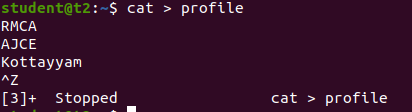
Text

Description automatically generated

4.1) mv -b create a duplicate file.

$mv-new mark1

**output:**



4.2) mv -i –to display a prompt message.

$mv -i profile profile1

**output:**



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

**Experiment 7**

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

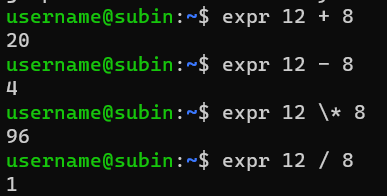
Perform system administration task

**Procedure:**

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

$expr 12 + 8

Output:



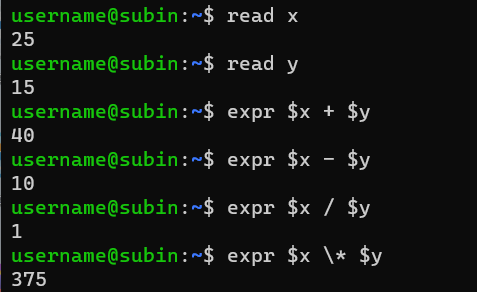
* 1. expr x + y :- Add two variables obtained through read

$read x

$read y

$expr $x + $y

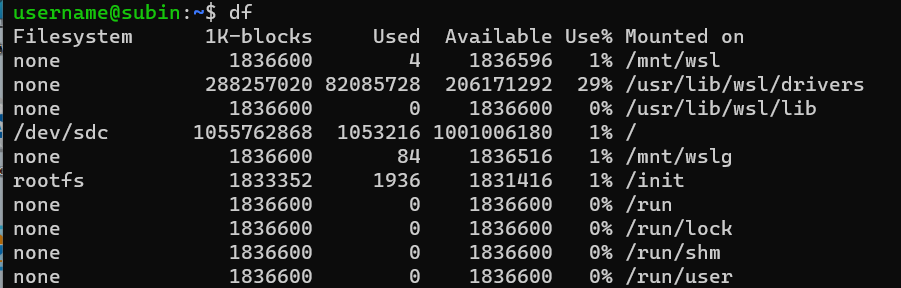
Output:



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

$df

Output:



1. du :- check how much space a file or directory in a given directory

$du file.txt

Output:



1. sudo :- superuser do
   1. sudo useradd user :- Add new user

$sudo useradd Subin

Output:

* 1. sudo passwd user :- Update password of the user

$sudo passwd Subin

Output:

* 1. sudo groupadd -g identifier name:- To create new group

$sudo groupadd -g 269 mcastd

* 1. sudo usermod -G name user :- Add users to group

$sudo usermod -G mcastd Subin

* 1. id user :- Details on group name and numeric id of particular user.

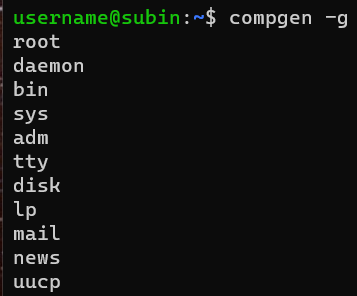
$id Subin

Output:

1. compgen -g :- Display all the groups created

$compgen -g

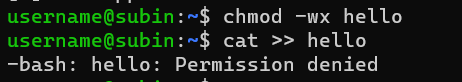
Output:



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

$chmod -wx file

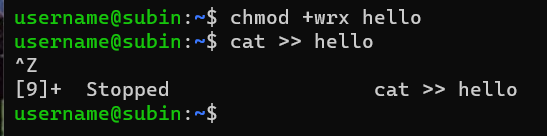
Output:



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

$chmod +wrx file

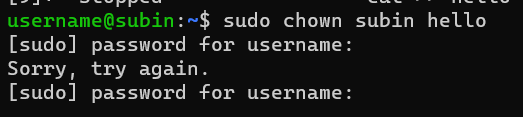
Output:



1. sudo chown :- Used to change ownership of a file or directory for a user or a group. It stands for change owner.

$sudo chown Subin file

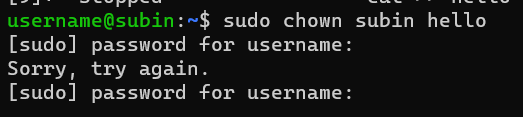
Output:



1. sudo userdel user :- Delete user

$sudo userdel Subin

Output:



1. sudo groupdel name :- Delete group

$sudo groupdel mcastd

Output



**Result:**

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

**Experiment 8**

**Aim:**

Familiarization of Linux Commands

**Course Outcome(CO2):**

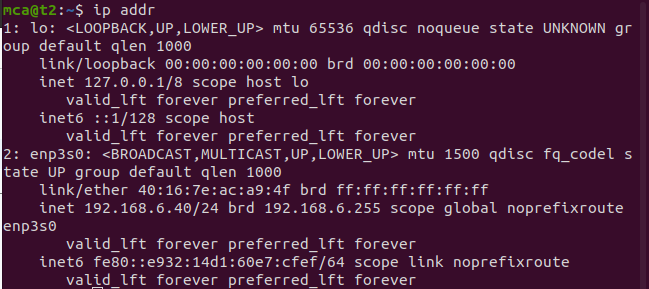
Perform system administration task

**Procedure:**

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

$ip addr

Output:

\

1. 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

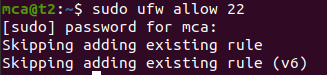
Output:



* 1. sudo apt-get install openssh -server :- Update port
  2. sudo ufw allow 22

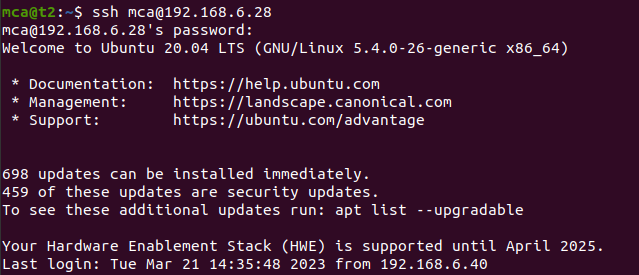
$sudo ufw allow 22

Output:



* 1. $ssh mca@192.168.6.28

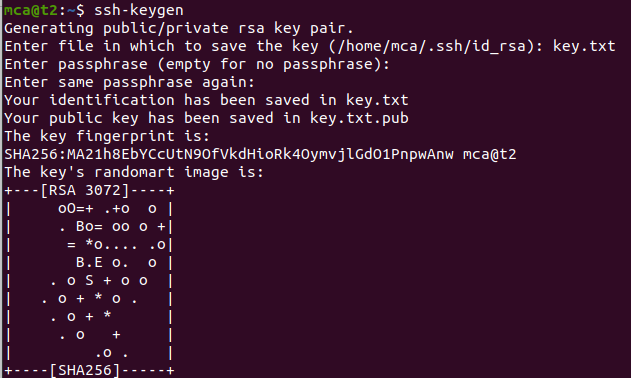
Output:



* 1. ssh-keygen :- Generating a key for secure shell

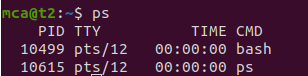
$ssh-keygen

Output:



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

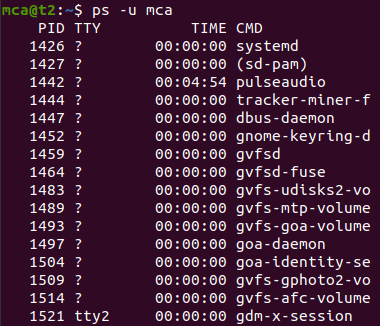
Output:



* 1. ps -u :- Display all running processes of a particular user

$ps -u mca

Output:



* 1. ps -C :- Specific process

$ps -C firefox

Output:



* 1. ps -f –p PID :- List the process by id

$ps -f -p 2762

Output:



**Result:**

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

**Experiment 9**

**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

value=`expr 2 + 13`

echo "Total value : $value"

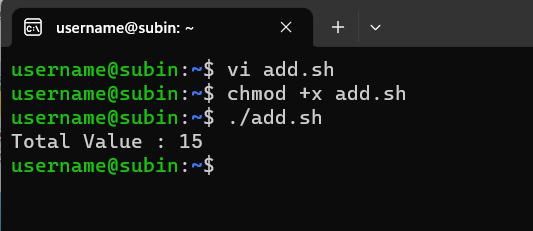
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. 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

num1=12

num2=48

sum=$(( $num1 + $num2 ))

echo "Sum : $sum"

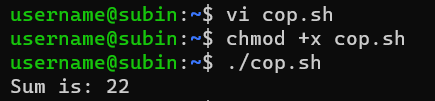
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



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

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

sum=$(( $1 + $2 ))

echo "Sum : $sum"

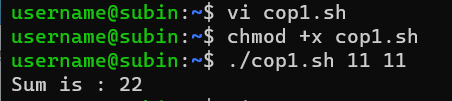
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh num1 num2

Output:



1. 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

read -p "Enter the First number: " num1

read -p "Enter the Second number: " num2

sum=$(( $num1 + $num2 ))

echo "Sum : $sum"

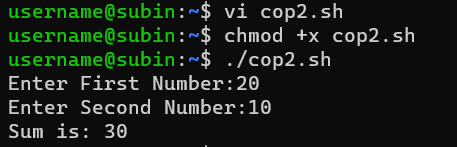
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh num1 num2

Output:



**Result:**

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

**Experiment 10**

**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)

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

num1=54

num2=20

if [ $num1 -eq $num2 ]

then

echo "-eq : $num1 is equal to $num2"

else

echo "-eq: $num1 is not equal to $num2"

fi

if [ $num1 -ne $num2 ]

then

echo "-ne: $num1 is not equal to $num2"

else

echo "-ne: $num1 is equal to $num2"

fi

if [ $num1 -gt $num2 ]

then

echo "-gt: $num1 is greater than $num2"

else

echo "-gt: $num1 is not greater than $num1"

fi

if [ $num1 -lt $num2 ]

then

echo "-lt: $num1 is less than $num2"

else

echo "-lt: $num1 is not less than $num2"

fi

if [ $num1 -ge $num2 ]

then

echo "-ge: $num1 is greater than or equal to $num2"

else

echo "-ge: $num1 is not greater than or equal to $num2"

fi

if [ $num1 -le $num2 ]

then

echo "-le: $num1 is less than or equal to $num2"

else

echo "-le: $num1 is not less than or equal to $num2"

fi

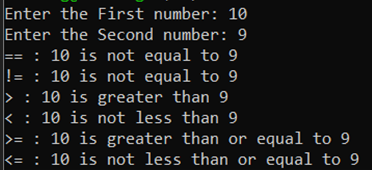
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



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)

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

num1=54

num2=20

if [ $num1 -eq $num2 ]

then

echo "-eq : $num1 is equal to $num2"

else

echo "-eq: $num1 is not equal to $num2"

fi

if [ $num1 -ne $num2 ]

then

echo "-ne: $num1 is not equal to $num2"

else

echo "-ne: $num1 is equal to $num2"

fi

if [ $num1 -gt $num2 ]

then

echo "-gt: $num1 is greater than $num2"

else

echo "-gt: $num1 is not greater than $num1"

fi

if [ $num1 -lt $num2 ]

then

echo "-lt: $num1 is less than $num2"

else

echo "-lt: $num1 is not less than $num2"

fi

if [ $num1 -ge $num2 ]

then

echo "-ge: $num1 is greater than or equal to $num2"

else

echo "-ge: $num1 is not greater than or equal to $num2"

fi

if [ $num1 -le $num2 ]

then

echo "-le: $num1 is less than or equal to $num2"

else

echo "-le: $num1 is not less than or equal to $num2"

fi

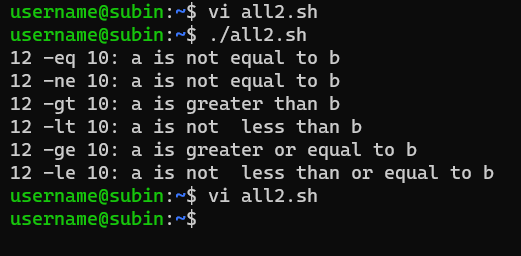
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



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

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the First Boolean: " b1

read -p "Enter the Second Boolean: " b2

if(($b1 == "true" & $b2 == "true" ))

then

echo Both are true

else

echo Both are not true

fi

if(($b1 == "true" || $b2 == "true" ))

then

echo Atleast one of them is true

else

echo None of them are true

fi

if(( ! $b1 == "true" ))

then

echo "b1" was initially false

else

echo "b1" was initially true

fi

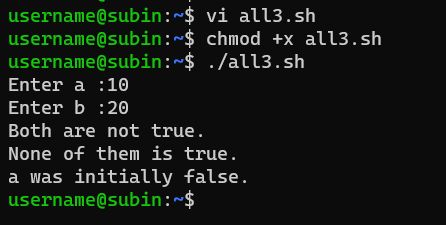
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. 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 a Number: " num1

if(( $num1 == 0 ))

then

echo "$num1 is neither odd nor even number"

elif(( $num1 % 2 == 0 ))

then

echo "$num1 is an even number"

else

echo "$num1 is a odd number"

fi

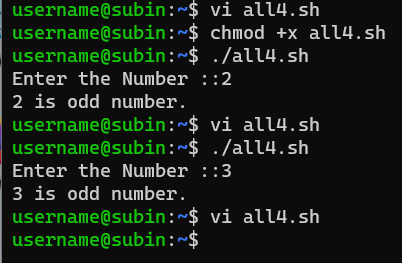
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. 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 a Number: " num1

if(( $num1 > 0 ))

then

echo "$num1 is a positive number"

elif(( $num1 < 0 ))

then

echo "$num1 is a negative number"

else

echo "$num1 is zero"

fi

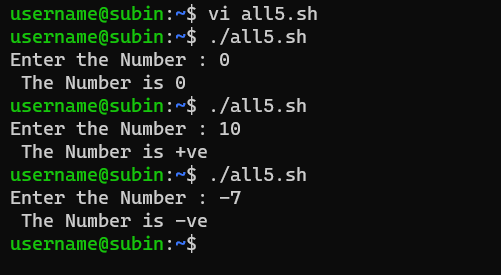
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



**Result:**

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

**Experiment 11**

**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 first string: " str1

read -p "Enter the second string: " str2

if(( $str1=$str2 ))

then

echo "Both strings are equal"

else

echo "Both strings are not equal"

fi

if(( $str1!=$str2 ))

then

echo "Both strings are not equal"

else

echo "Both strings are equal"

fi

if(( -z$str1 ))

then

echo "String size is zero"

else

echo "String size is non-zero"

fi

if(( -n$str1 ))

then

echo "String size is non-zero"

else

echo "String size is zero"

fi

if(( $str1 ))

then

echo "String is Empty"

else

echo "String is not Empty"

fi

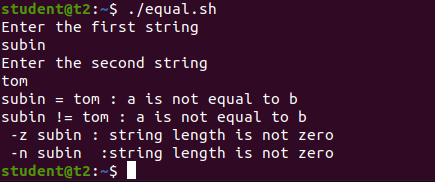
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



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

vi filename.sh

Press ‘i’ to INSERT

#!/bin/bash

read -p "Enter the first value: " bin1

read -p "Enter the second value: " bin2

result=$(( $bin1&$bin2 ))

echo "Bitwise AND: $result"

result=$(( $bin1|$bin2 ))

echo "Bitwise OR: $result"

result=$(( $bin1^$bin2 ))

echo "Bitwise XOR: $result"

result=$(( ~$bin1 ))

echo "Bitwise COMPLEMENT of First Value: $result"

result=$(( $bin1<<2 ))

echo "Bitwise RIGHT SHIFT(4) of First Value: $result"

result=$(( $bin1>>2 ))

echo "Bitwise LEFT SHIFT(4) of Second Value: $result"

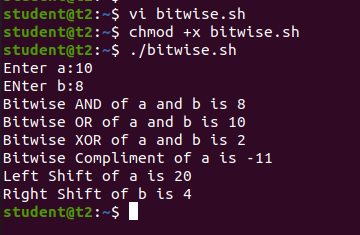
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. 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

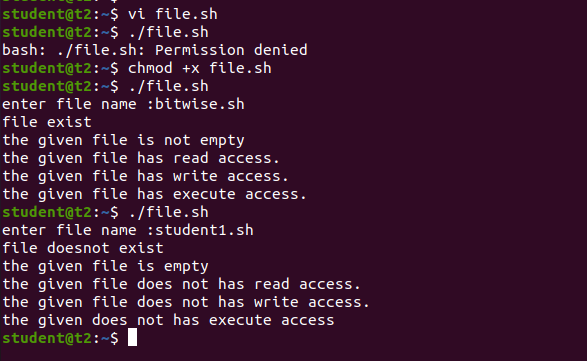
Press ‘Esc’ to end INSERT

:wq!

chmod +x filename.sh

./filename.sh

Output:



1. 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:

