**EXPERIMENT NO: 3**

**AIM:** Familiarization of linux commands.

**CO2:** Perform system administration task.

**PROCEDURE:**

pwd - Print the working directory find the path of the current working directory

$pwd



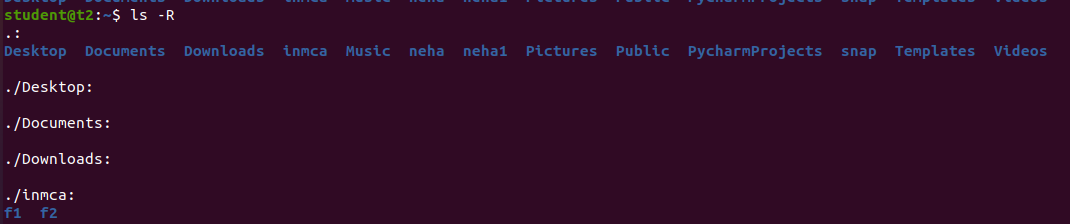
ls – To view the content of the directory

$ls

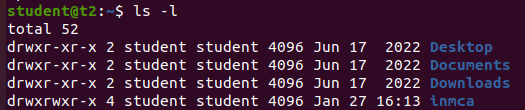


ls -R – To list the contents of sub directory

$ls -R

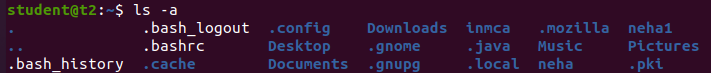


ls -l – Long listing of the contents



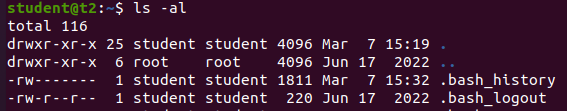
ls -a – To list the all hidden files

$ls -a



ls -al – List the files and directories with detailed information.

$ls -al



ls -t – List the files sorted in the order of last modified.

$ls -t



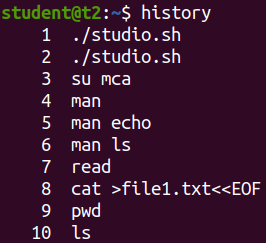
ls -r – To reverse the natural sorting order

$ls -r



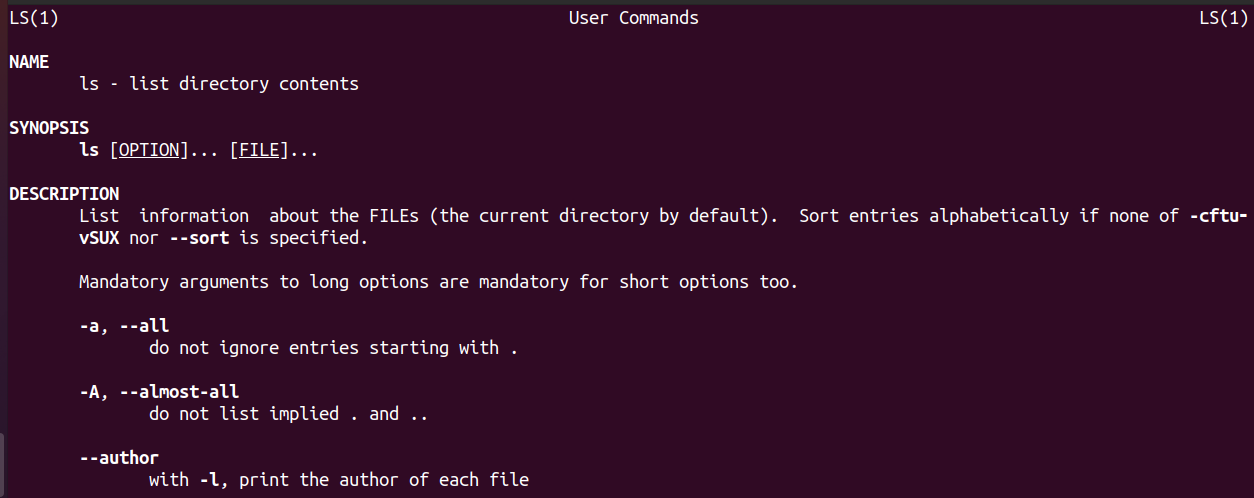
history – To review the command that have been previously executed for a certain period of time.

$history



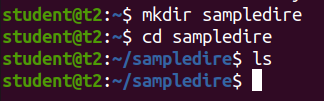
man – learn and understand about different command right from the shell using man command

$man



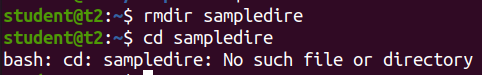
mkdir – To create a new directory

$mkdir



rmdir – To remove a directory

$rmdir



touch – To create new empty file

$touch

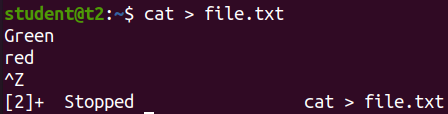


cat – Concatenate the files and print on the standard output

$cat

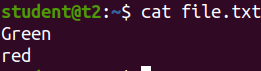
cat > filename.txt – To create a file with inserting contents

$cat > file.txt



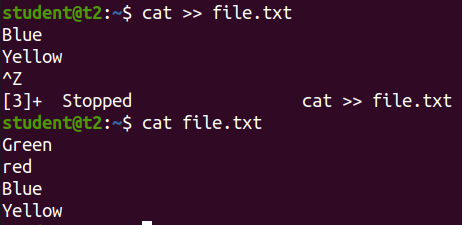
cat filename.txt – To view the content of the file

$cat file.txt



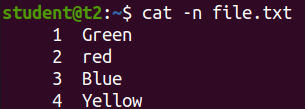
cat >> filename.txt – To append new contents to an existing file

$cat >> file.txt



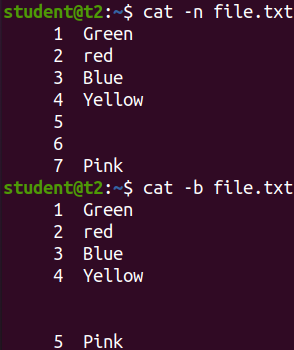
cat -n filename.txt – Number all output lines

$cat -n file.txt



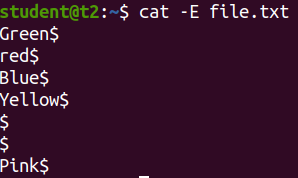
cat -b filename.txt – To remove the empty lines

$cat -b file.txt



cat -E filename.txt – Display $ at end of each line

$cat -E file.txt



**Result**

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

**EXPERIMENT NO :4**

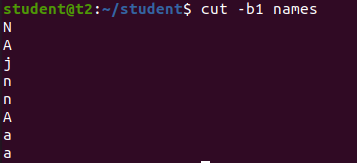
**AIM : Familiarization of linux commands**

**CO2:** Perform system administration task.

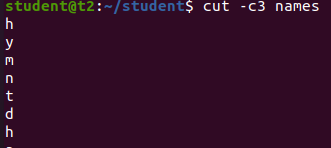
**PROCEDURE:**

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

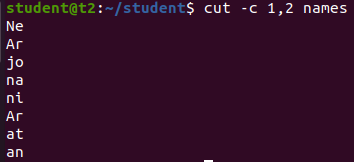
* $cut -b1 names – cut by bytes



* $cut -c3 names – cut by character position



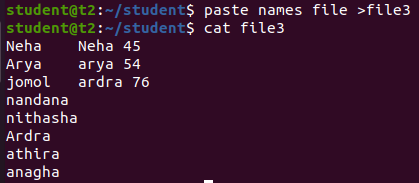
* $cut -d - - f1 file – to show first column
* $cut -c1 1,2 names – To specify the columns



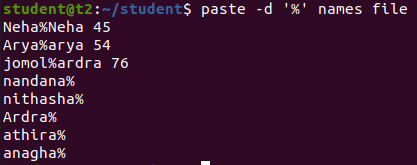
* $cut -d ‘’ -f1 file5 – To cut the space

paste - To join files horizontally (each files consisting of different lines)

* $paste names file>file3



* $paste -d ‘%’ names file

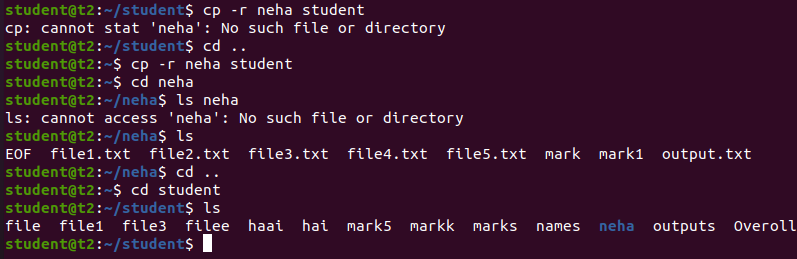


* $paste -s names – To print in single line



copy - To copy the contents of the file

* $cp -r neha student



**Result**

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

**EXPERIMENT NO : 5**

**AIM :** Familiarization of linux commands

**CO2:** Perform system administration task.

**Procedure:**

read command – To read the contents of a line we use ‘read’ command

* $read



REPLY - to print the read line

* $echo $REPLY



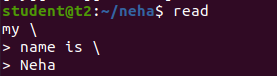
* $read - to store the read content in different variables



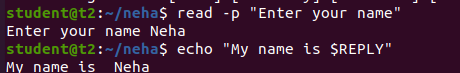
* $echo “[$var1] [var2] [var3]”-to print the variable stored contents



* $read my\ name is\ neha -to read multiple lines using \



* $read -p -to promt the text



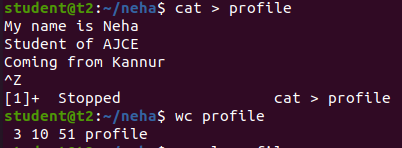
* $read -n 6 -p -to limit the specified text



* $read -s -p -To secure the password



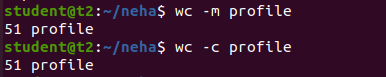
* $wc profile -To display the number of lines,bits,word



* $wc -l -number of lines



* $wc -m -number of bit



* $wc -w -number of word



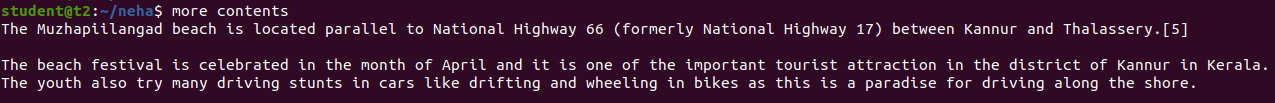
* $wc -L -To display the length of longest line

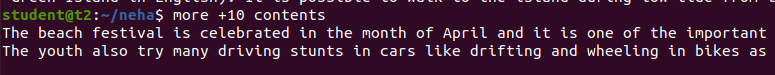


more –‘more’ command is similar to ‘cat’ command to display the contents .The only difference is that in case of larger files ‘cat’ command output will scroll off your screen while ‘more’ command display output once screenful at atime.

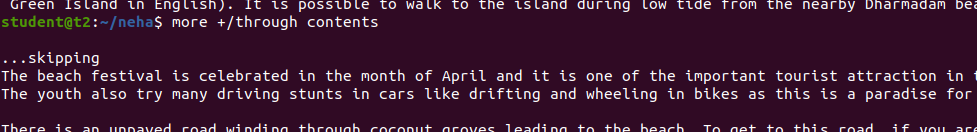
more filename-

* $more contents

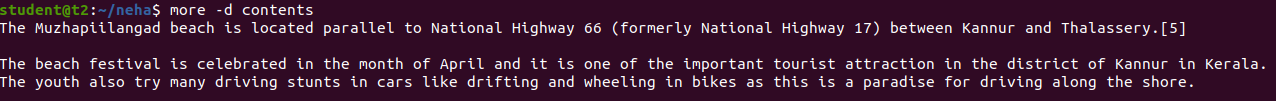


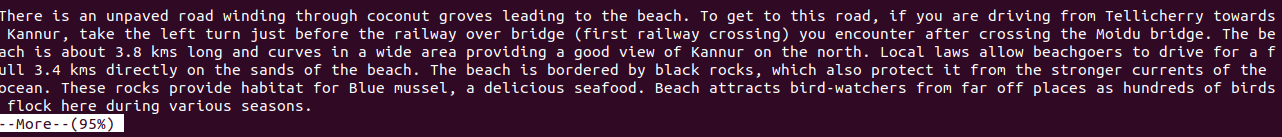


* $more +/through contents – The string contained paragraph will show.



* $read -d contents – It helps the user to navigate according to the instruction





**Result**

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

**EXPERIMENT NO:6**

**AIM:**Familiarization of linux commands

**CO2:**Perform system administration task

**PROCEDURE:**

grep – Filter the content which makes our search easy

* $ grep 43 marks

OUTPUT:

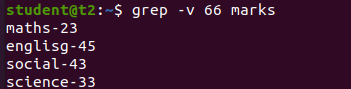


* $ grep -i maths marks – case insensitive search

OUTPUT:



* $ grep -v 66 marks – Display all the contents except the searched contents.



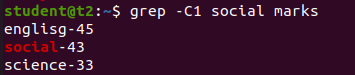
* $ grep -A1 maths marks -Display the contents along with one line after that



* $ grep -B1 malayalam marks – Display the contents along with one line before that

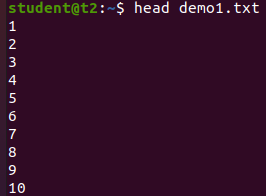


* $ grep -C1 social marks – Display the contents along with one line before and after .



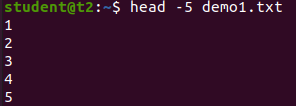
head – Display the first ten lines of the contents

* $ head demo1.txt



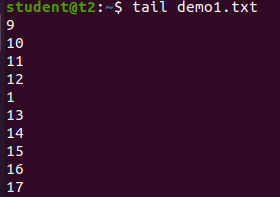
head-5 – Display the firat five contents

* $ head -5 demo1.txt

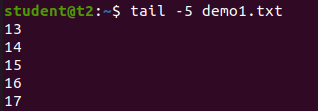


t ail – Display the last ten lines of the content

* $ tail demo1.txt

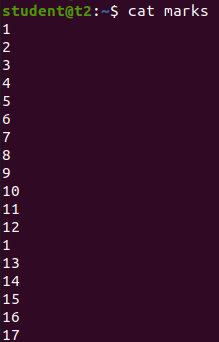


* $ tail -5 demo1.txt



mv – To move the file or directory

* $ mv demo1.txt marks
* $ cat marks



* $ mv -b marks profile – To backup the files.
* $ ls



* $ mv -i profile profile1 – Ask for overwrite the file or what

mv: overwrite 'profile1'? y



**Result**

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

**EXPERIMENT NO :7**

**AIM:** Familiarization of linux commands

**CO2:** Perform system administration task

**PROCEDURE:**

expr -Evaluate the given expression and display the output .

* $ expr 12 + 8



* $ expr 12 – 8



* $ expr 12 \\* 3



* $ expr 12 / 4



* $ read x



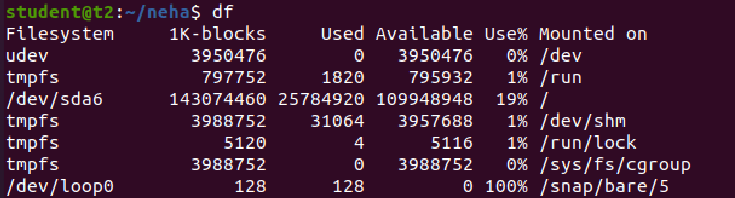
* $ read y



* $ expr $x + $y



* $ df – Get a report on system disk space usage



du – To check how much space of a file or directory takes in current directory

* $ du file1.txt

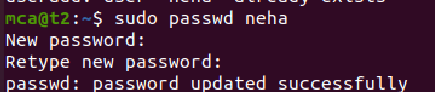


sudo -  allows you to run a command as root

* $sudo useradd neha



* $sudo passwd neha



* sudo groupadd -g 333 mcastaff



* $sudo usermod -G mcastaff neha



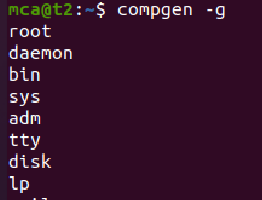
id – Used to find group name and numeric id

* $id neha



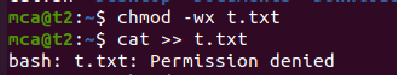
compgen -  It is bash built-in command and it will show all available commands, aliases, and functions for you.

* $compgen -g - Used to display all the groups

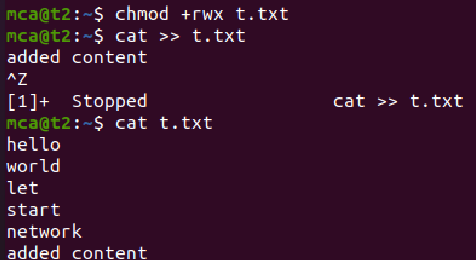


chmod – Used to change access permissions of files and directories .It stands change mode (read (r), write (w ), execute (x)).

* $chmod -wx t.txt

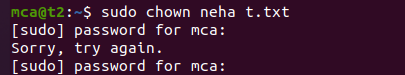


* $chmod rwx t.txt

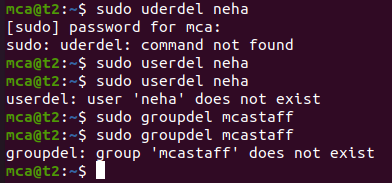


sudo chown – Used to change a file ownership or directory ownership for a user or a group down stands for change owner.

* $sudo chown neha file1.txt



* $sudo userdel



**Result**

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

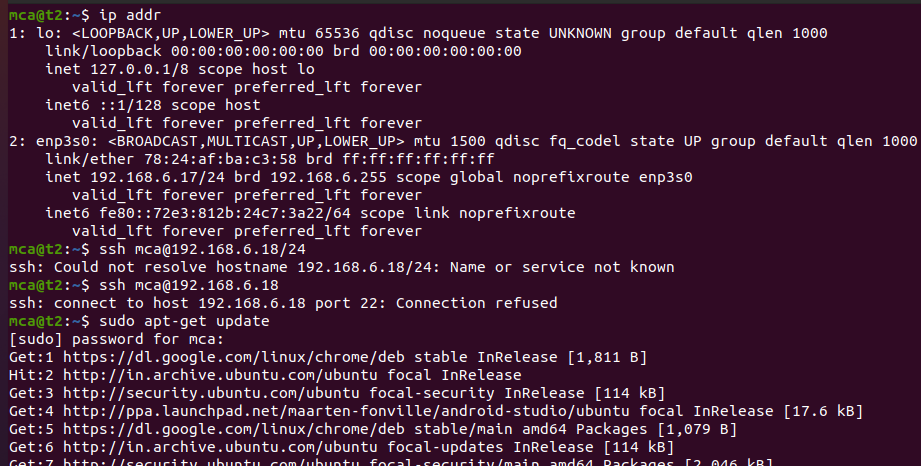
**EXPERIMENT NO: 8**

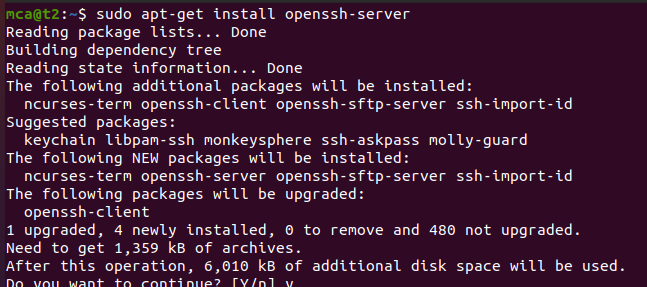
**AIM:** Familiarization of linux commands.

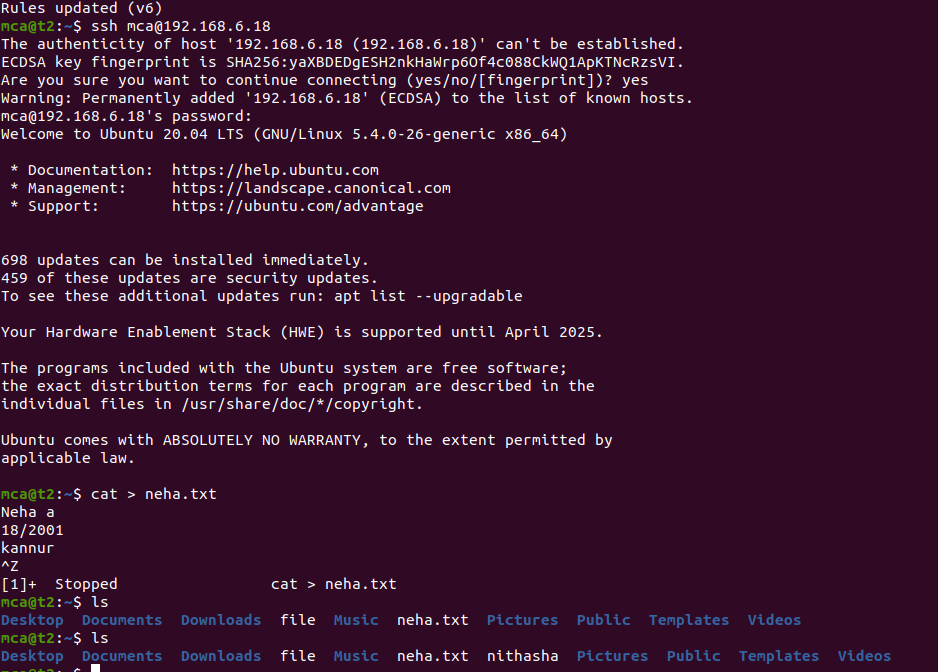
**CO2:** Perform system administration task.

**PROCEDURE**

1. Ip addr :





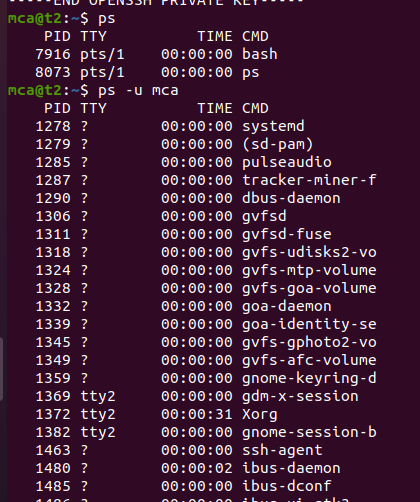


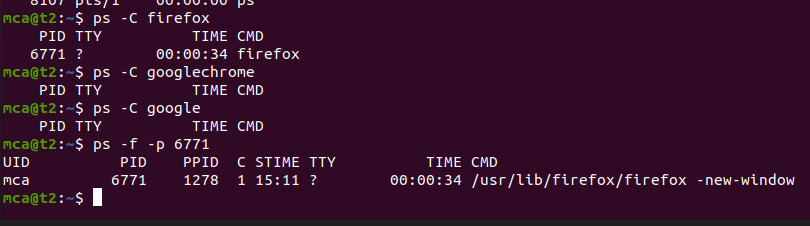
ssh stands for secure shell

$ssh-keygen : generate a key for ssh



Ps : currently running program





**Result :**

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

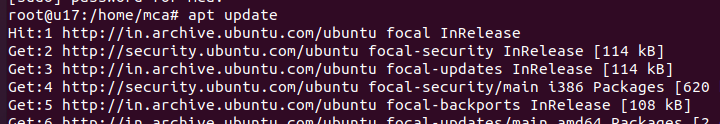
AIM : File system hierarchy

CO2 : Perform system administration task

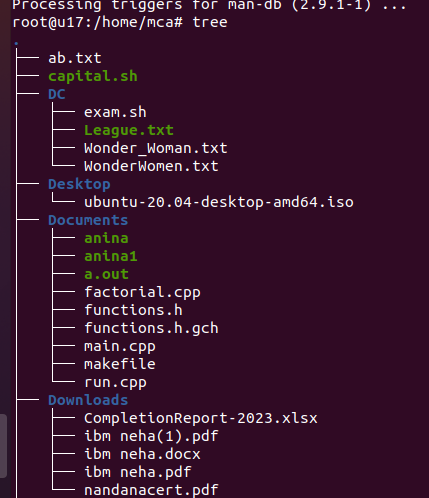
sudo su



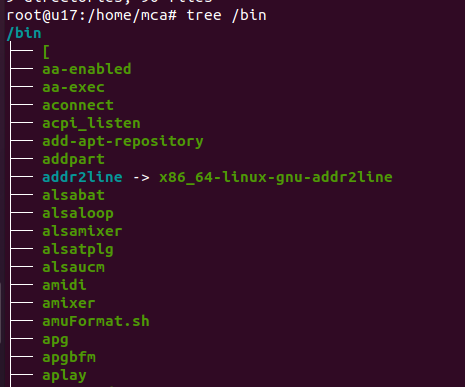
apt update



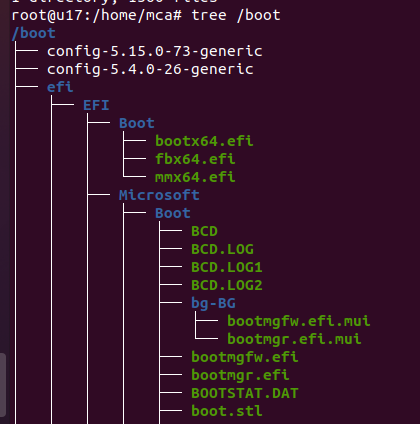
tree



tree /bin



tree /boot



**SHELL SCRIPTING**

**EXPERIMENT NO : 1**

**AIM** : Shell script to display current time

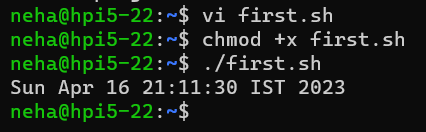
**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

Date

**OUTPUT**

****

**RESULT**

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

**EXPERIMENT NO : 2**

**AIM** : Shell script to display your name

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

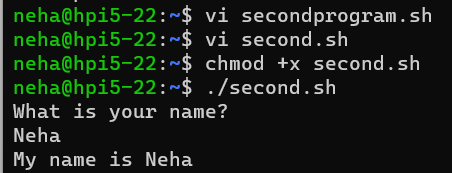
#!/bin/bash

echo "What is your name?"

read name

echo "My name is $name"

**OUTPUT**

****

**RESULT**

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

**EXPERIMENT NO : 3**

**AIM** : Shell script to commands

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

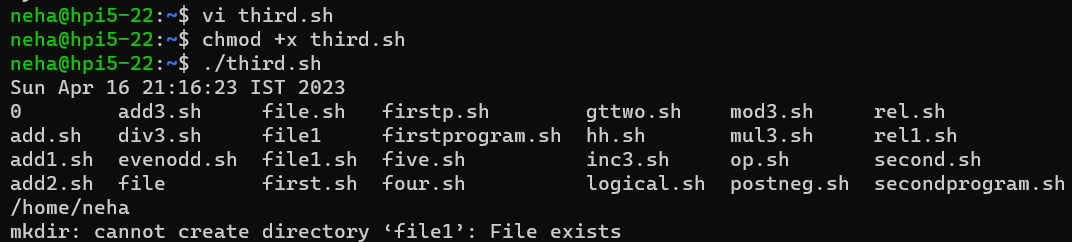
date

ls

pwd

mkdir file1

**OUTPUT**

****

**RESULT**

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

**EXPERIMENT NO : 4**

**AIM** : Shell script to demonstrate variables

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

echo "File name:$0"

echo "first prameter:$1"

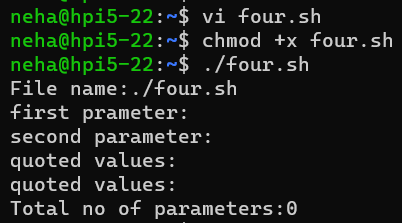
echo "second parameter:$2"

echo "quoted values:$@"

echo "quoted values:$\*"

echo "Total no of parameters:$#"

**OUTPUT**

****

**RESULT**

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

**EXPERIMENT NO : 5**

**AIM** : Shell script to display an array

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

name[0]="neha"

name[1]="arya"

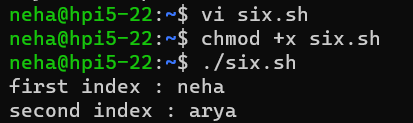
name[2]="ardra"

name[3]="anju"

echo "first index : ${name[0]}"

echo "second index : ${name[1]}"

**OUTPUT**

****

**RESULT**

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

**EXPERIMENT NO : 6**

**AIM :**Shell script to demonstrate arithematic operators

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

read -p "Enter a: " a

read -p "Enter b: " b

add=$(( a + b ))

echo "Sum is: $add"

sub=$(( a - b ))

echo "sub is : $sub"

mul=$(( a \* b ))

echo "mul is : $mul"

div=$(( a / b ))

echo "div is : $div"

mod=$(( a % b ))

echo "mod is : $mod"

if [ $a == $b ]

then

echo "a is qual to b"

fi

if [ $a != $b ]

then

echo "a is not equal to b"

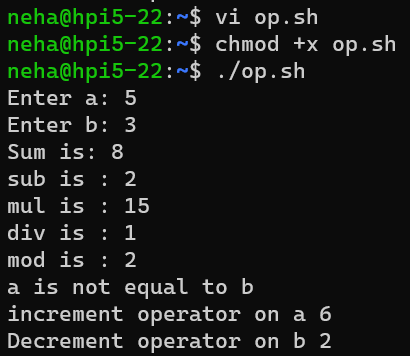
fi

(( ++a ))

echo "increment operator on a $a"

(( --b ))

echo "Decrement operator on b $b"



**RESULT**

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

**EXPERIMENT NO : 7**

**AIM :** Shell script to demonstrate relational operators

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

read -p "Enter a:" a

read -p "Enter b:" b

if(( $a == $b ))

then

echo " a is equal to b"

else

echo "a is not equal to b"

fi

if(( $a != $b ))

then

echo " a is not equal to b"

else

echo "a is equal to b"

fi

if(( $a < $b ))

then

echo " a is less than b"

else

echo "a is not less than b"

fi

if(( $a <= $b ))

then

echo " a is less than or equal to b"

else

echo "a is not less than or equal to b"

fi

if(( $a > $b ))

then

echo " a is greater than b"

else

echo "a is not greater than b"

fi

if(( $a >= $b ))

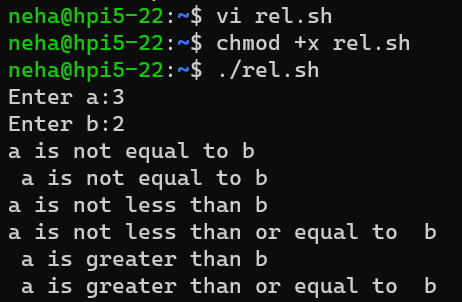
then

echo " a is greater than or equal to b"

else

echo "a is not greater than or equal to b"

fi



**RESULT**

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

**EXPERIMENT NO : 8**

**AIM :** Shell script to demonstrate logical operators

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/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 them is true

else

echo None of them is true

fi

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

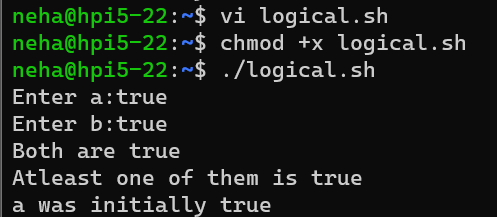
then

echo a was initially false

else

echo a was initially true

fi



**RESULT**

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

**EXPERIMENT NO : 9**

**AIM :** Shell script to demonstrate even and odd numbers

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

read -p "Enter a number :" num

if(( num % 2 == 0 ))

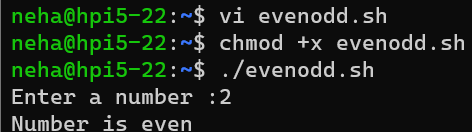
then

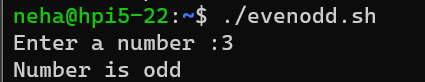
echo "Number is even"

else

echo "Number is odd"

fi





**RESULT**

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

**EXPERIMENT NO : 10**

**AIM :** Shell script to demonstrate simple if loop

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

read -p "Enter first number :" $a

read -p "Enter second number :" $b

if [ $a == $b ]

then

echo "a is equal to b"

fi

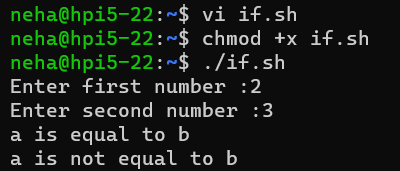
if [ $a != $b ]

then

echo "a is not equal to b"

fi

**OUTPUT**



**RESULT**

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

**EXPERIMENT NO : 11**

**AIM :** Shell script to demonstrate if else loop

**CO4** : Write shell scripts required for system administration

**PROCEDURE**

#!/bin/bash

read -p "Enter number :" num

if (( $num>=0 && $num<=10 ))

then

echo " Number is between 0 and 10"

elif (( $num >= 11 && $num <= 20 ))

then

echo "Number is betwwe 11 and 20"

elif (( $num >= 21 && $num <= 30 ))

then

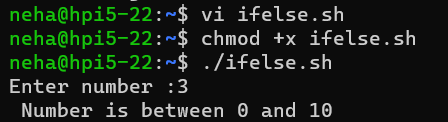
echo "Number is between 21 and 30"

else

echo "Number is greater than 30"

fi

**OUTPUT**



**RESULT**

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

for i in 1 2 3 4 5 6 7 8 9 10

do

if [ $i == 5 ]

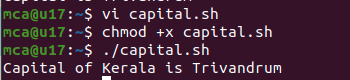
then

break

fi

echo "iteration no $i"

done



#!/bin/bash

read -p "Enter any color in vibgyor :" vibgyor

case "$vibgyor" in

"v") echo "v for violet";;

"i") echo "i for indigo";;

"b") echo "b for blue";;

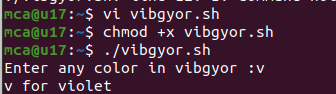
"g") echo "g for green";;

"y") echo "y for yellow";;

"o") echo "o for orange";;

"r") echo "r for red";;

esac



#!/bin/bash

a=0

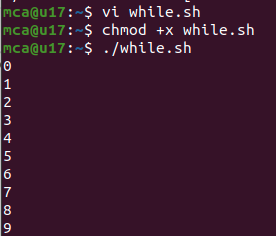
while [ $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done



#!/bin/bash

a=10

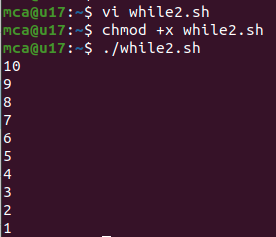
while [ $a -gt 0 ]

do

echo $a

a=`expr $a - 1`

done



#!/bin/bash

echo "Enter a number:"

read n

num=0

on=$n

while [ $n -gt 0 ]

do

num=$(expr $num \\* 10)

k=$(expr $n % 10)

num=$(expr $num + $k)

n=$(expr $n / 10)

done

if [ $num -eq $on ]

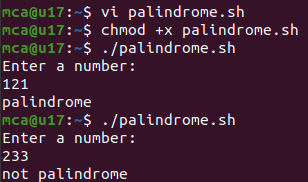
then

echo palindrome

else

echo not palindrome

fi



#!/bin/bash

fruit="apple"

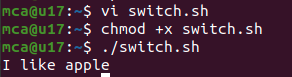
case "$fruit" in

"apple") echo "I like apple";;

"banana") echo "I like banana";;

"orange") echo "I like orange";;

esac



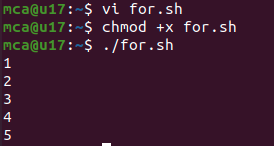
#!/bin/bash

for i in {1..5}

do

echo "$i"

done



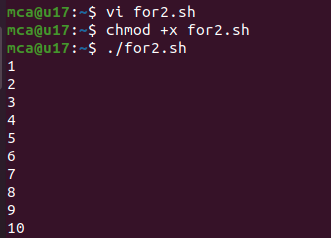
#!/bin/bash

for (( i=1; i<=10; i++ ))

do

echo "$i"

done



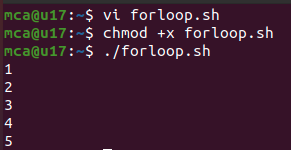
#!/bin/bash

for i in 1 2 3 4 5

do

echo "$i"

done



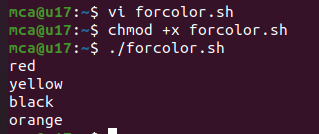
#!/bin/bash

for i in 'red' 'yellow' 'black' 'orange'

do

echo "$i"

done



#!/bin/bash

for i in 1 2 3 4 5 6 7 8 9 10

do

if [ $i == 5 ]

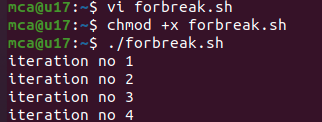
then

break

fi

echo "iteration no $i"

done



#!/bin/bash

for i in 1 2 3 4 5 6 7 8 9 10

do

if [ $i == 5 ]

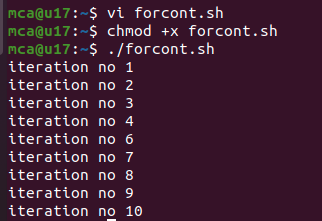
then

continue

fi

echo "iteration no $i"

done



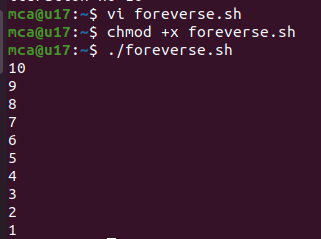
#!/bin/bash

for (( i=10; i>0; i-- ))

do

echo "$i"

done



#!/bin/bash

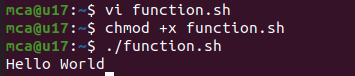
hello()

{

echo "Hello World"

}

hello



#!/bin/bash

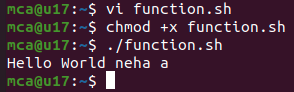
hello()

{

echo "Hello World $1 $2"

}

hello neha a



#!/bin/bash

i=2

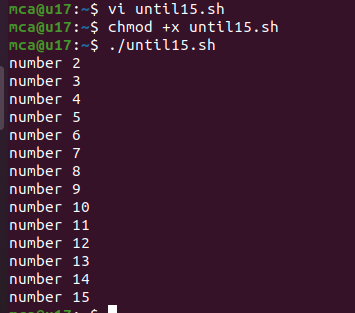
until [ $i -gt 15 ];

do

echo "number $i"

i=$(( i+1 ))

done



**Make**

function1.cpp

#include "functions.h"

int factorial(int n)

{

if(n!=1)

{

return(n\*factorial(n-1));

}

else return 1;

}

function2.cpp

#include<iostream>

#include "functions.h"

void print\_hello()

{

std::cout << "hello world";

}

functions.h

void print\_hello();

int factorial(int n);

main.cpp

#include<iostream.h>

#include "functions.h"

int main()

{

print\_hello();

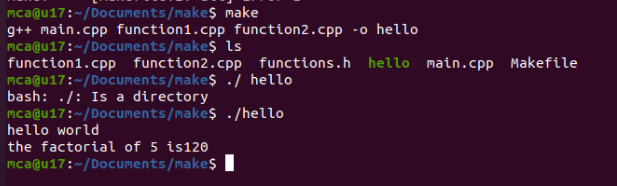
std::cout<< std::endl;

std::cout<< "the factorial of 5 is" <<factorial(5)<<std::endl;

return 0;

}

OUTPUT



**CMake**

add.cp

#include "add.h"

int add(int a, int b)

{

return a + b;

}

add.h

#pragma once

int add(int a,int b);

CmakeLists.txt

cmake\_minimum\_required(VERSION 3.16.3)

project("Hello world")

add\_executable(a.out main.cpp add.cpp)

main.cpp

#include<iostream>

#include "add.h"

int main()

{

std::cout << "Sum of 20 and 30 is :" << add(20,30)<<"\n";

return 0;

}

OUTPUT

