**Experiment No.: 3**

**Aim:** Familiarization of linux commands.

**CO2:** Perform system administration task.

**Procedure:**

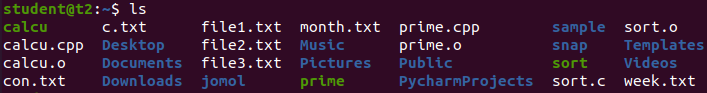
1.pwd - Print the working directory find the path of the current working directory

$pwd

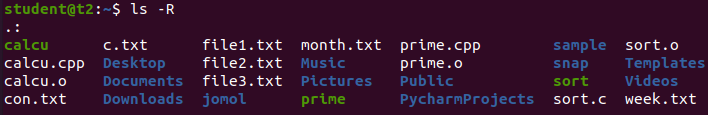


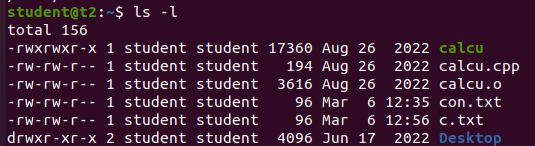
2.ls – To view the content of the directory

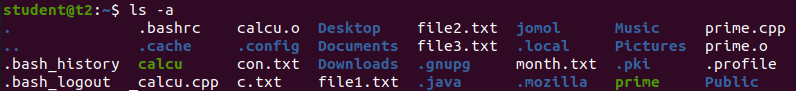
$ls

1.ls -R – To list the contents of sub directory

$ls -R

2.ls -l – Long listing of the contents

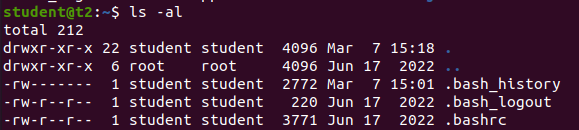


3.ls -a – To list the all hidden files

$ls -a

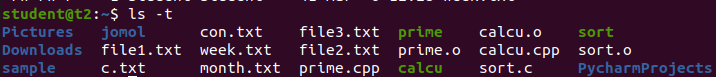
4.ls -al – List the files and directorys with detaild information.

$ls -al

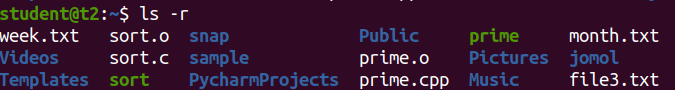


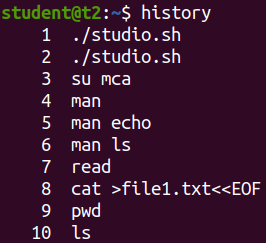
5.ls -t – List the files sorted in the order of last modified.

$ls -t

6.ls -r – To reverse the natural sorting order

$ls -r

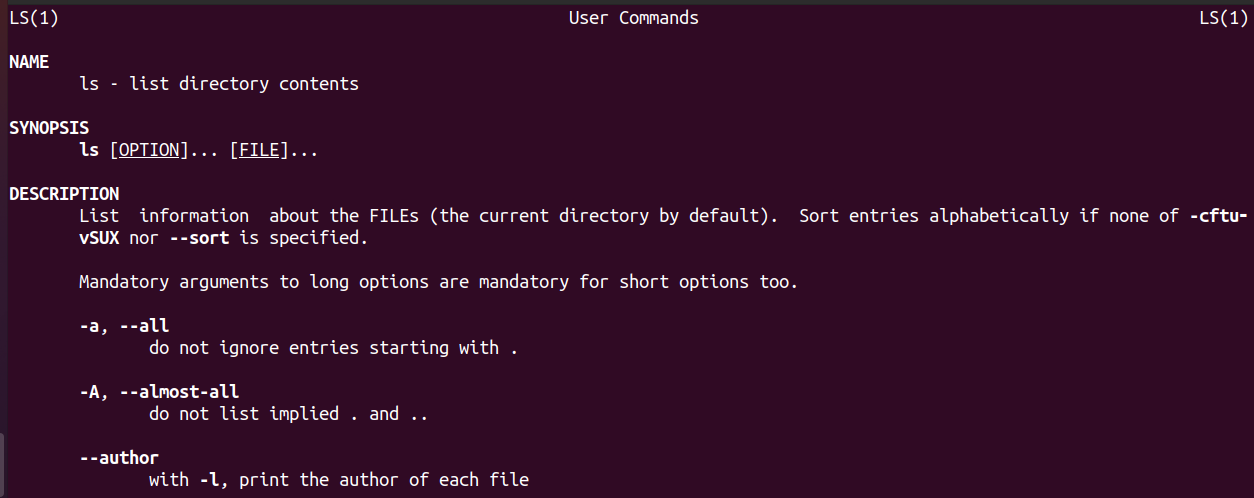


3.history – To review the command that have been previously executed for a certain period of time.

$history

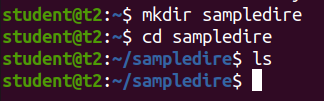
4.man – learn and understand about different command right from the shell using man command

$man



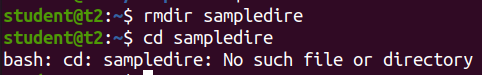
5.mkdir – To create a new directory

$mkdir



6.rmdir – To remove a directory

$rmdir



7.touch – To create new empty file

$touch

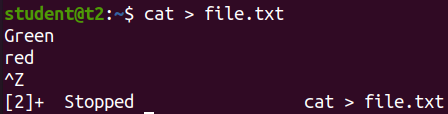


8.cat – Concatenate the files and print on the standard output

$cat

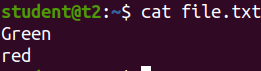
a. cat > filename.txt – To create a file with inserting contents

$cat > file.txt



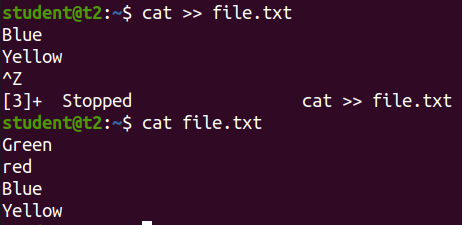
b. cat filename.txt – To view the content of the file

$cat file.txt



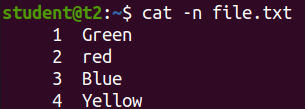
c. cat >> filename.txt – To append new contents to an existing file

$cat >> file.txt



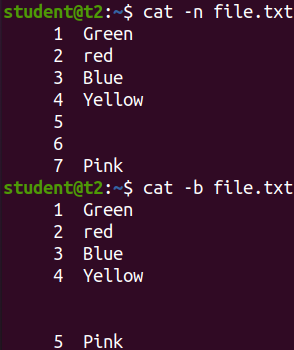
d. cat -n filename.txt – Number all output lines

$cat -n file.txt



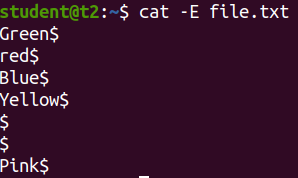
e. cat -b filename.txt – To remove the empty lines

$cat -b file.txt



f. 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:**

$read command



REPLAY - to print the read line



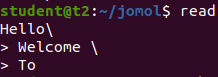
$read - to store the read content in different variables

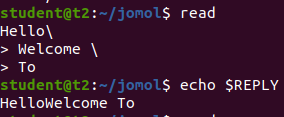


$echo “[$var1] [var2] [var3]” - to print the variable stored contents

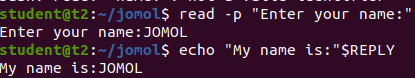


$read my\ name is\ - to read multiple lines using \





$read -p - to promt the text



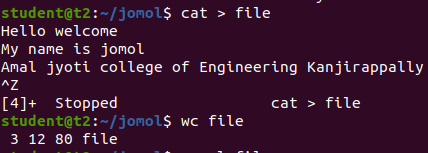
$read -n 6 -p - to limit the specified text



$read -s -p - To secure the password



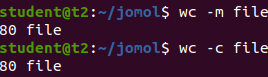
$wc profile - To display the number of lines,bits,word



$wc [filename] – to display the number of lines



$wc -m – to display the number of bit



$wc -w – to display the number of word



$wcf -L - To display the length of longest line

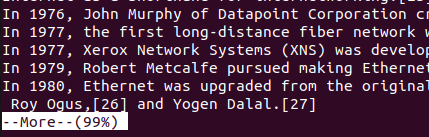


more –‘more’ command is similar to ‘cat’ command to display the contents .The only difference is that in case of larger files ‘cat’ command output will scroll off your screen while ‘more’ command display output once screenful at atime.

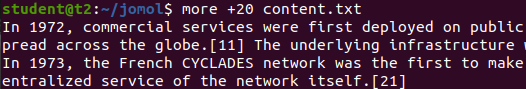
more filename-

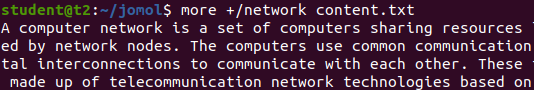
* $more contents





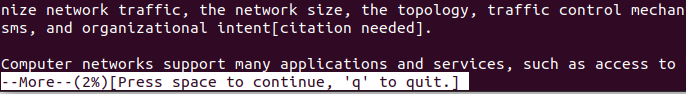
* $more +/through contents – The string contained paragraph will show.





* $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

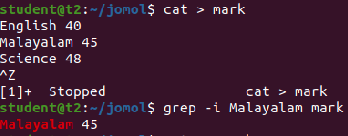
**PROCEDURE:**

Experiment :6

14-03-23

Grep is used filter the content which makes our search easy.

$grep -i - Case insensitive search



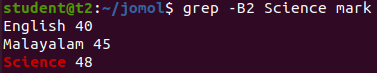
$grep -v display the all the content expect the searched content



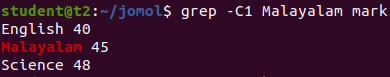
$grep -A1 - View the content along with one line after that.



$grep -B2 - View the content along with one line before that.

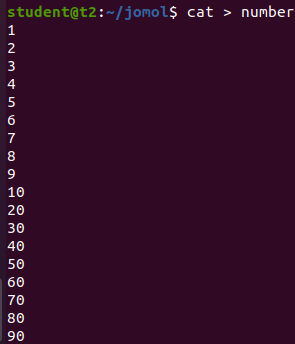


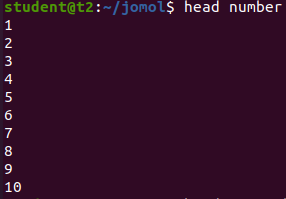
$grep -c1 - view the content along with one



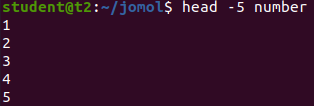
Head

$head  filename - Display the lines of files by default it display the top 10 lines

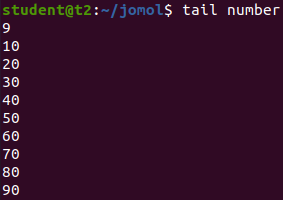




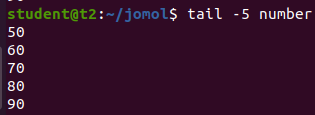
$head  -5 filename -  Display the content with limited range



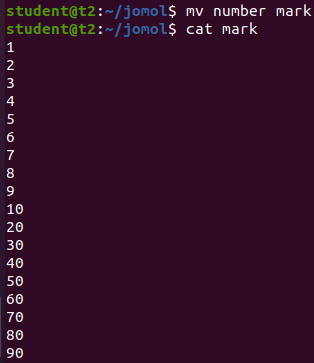
Tail - Display the last content of the by default it will display the last 10 lines.



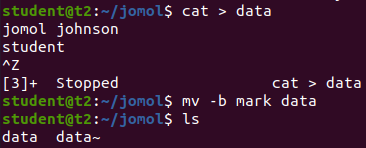
tail  -5  filename - Display the last 5 lines.



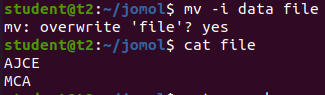
MV - To move the file to one file to another location and rewrite the content



mv  -b filename - Create a backup of the file.



mv  -i filename



**Result**

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

**EXPERIMENT NO :7**

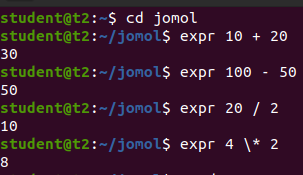
**AIM:** Familiarization of linux commands

**CO2:** Perform system administration task

**PROCEDURE:**

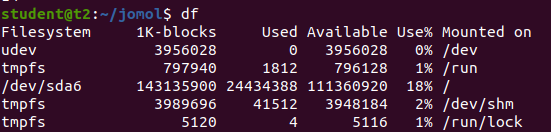
Expr - Used to calculate the expression and display the output

$expr



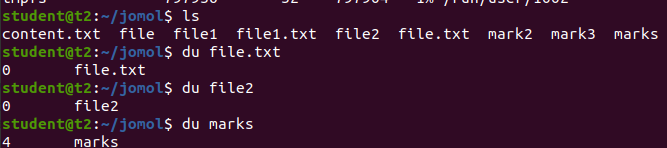
Df - Is used to get a report on system disk

$df



Du - is udes to check how much space a file or directory takes in current directory.

$du



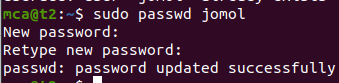
Sudo - To add super user

$sudo useradd [name]



Sudo - To change the password of the superuser

$sudo passwd [name]



Sudo groupadd -g [number][group name] - to add a new group

$sudo groupadd -g 432 mcastudent



Sudo usermod -G [Name][username] - To add new user to the group

$sudo usermod \_g mcastudent jomol



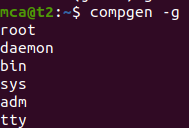
Id [name] - To show the details of the file.

$id jomol



Compgen -g - To display the all group

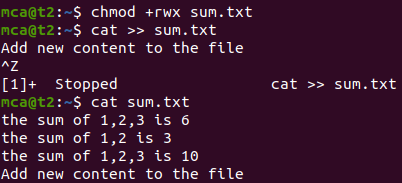
$compgen -g



Chmod [access specifier][filename] - Used to change the access permission of files and directories.it stands for change mod .Read(r),Write(w),Execute(e).

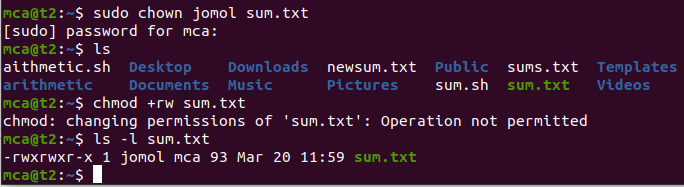
$chmod -wx sum.txt





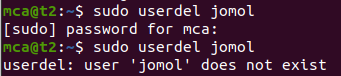
Chown - User to change the ownership or directory ownership for a user or a group. And it stands for change directory.

$sudo chown [username][filename]



Sudo userdel [name] -  To delete the User.

$sudo userdel jomol



**Result**

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

**SHELL SCRIPTING**

**EXPERIMENT NO : 1**

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

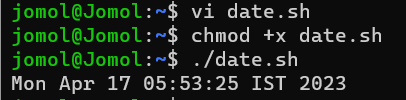
**CO4** : Write shell scripts required for system administration

**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**

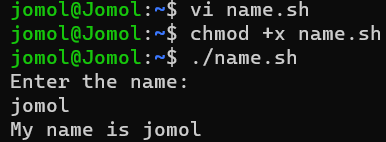
#!/bin/bash

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

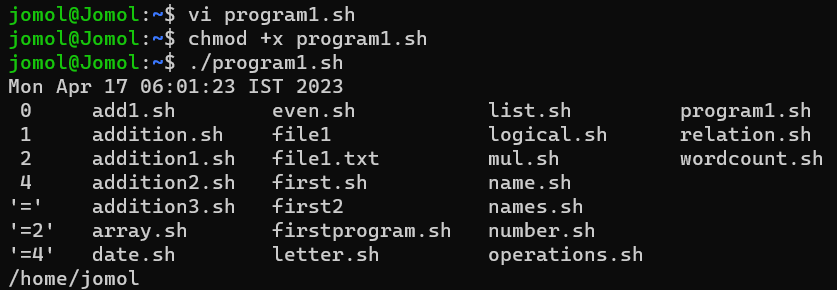
date

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"

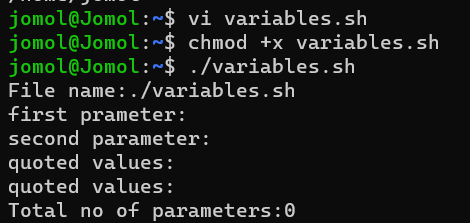
echo "second parameter:$2"

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]="Jomol"

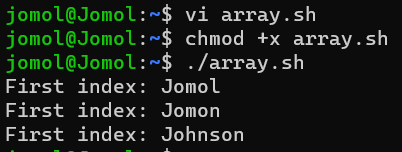
name[1]="Jomon"

name[2]="Johnson"

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

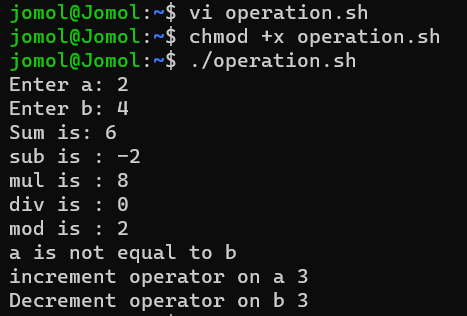
(( ++a ))

echo "increment operator on a $a"

(( --b ))

echo "Decrement operator on b $b"

**OUTPUT**



**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

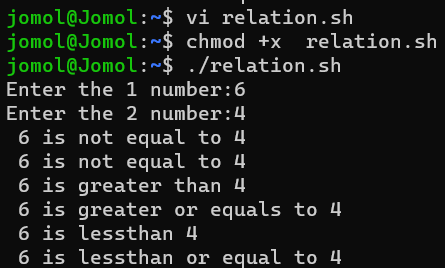
echo " a is greater than or equal to b"

else

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

fi

**OUTPUT**



**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

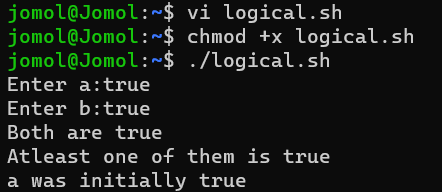
echo a was initially false

else

echo a was initially true

fi

**OUTPUT**



**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

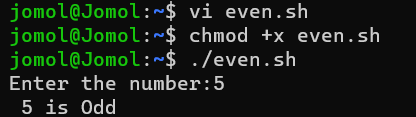
echo "Number is even"

else

echo "Number is odd"

fi

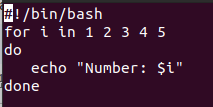
**OUTPUT**

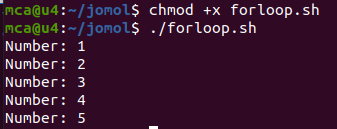


**RESULT**

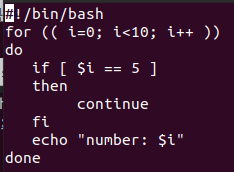
The program was executed and the result was successfully obtained. Thus CO4 was obtained.

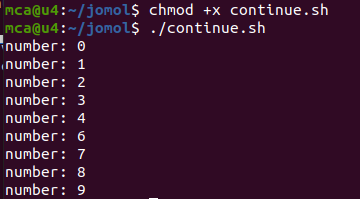
**For Loop**



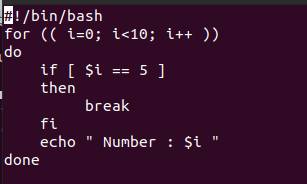


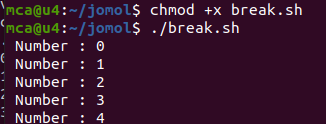
1.Write a shell script to implement for loop with continu statements



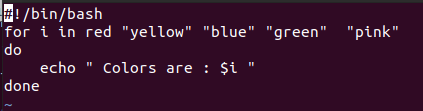


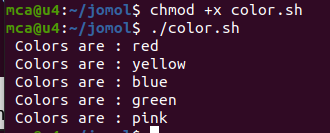
2.Write a shell script to implement for loop with brake statements



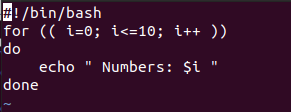


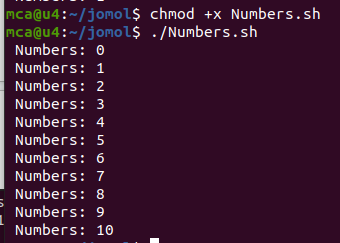
3.Write a shell Script to display colors using forloop



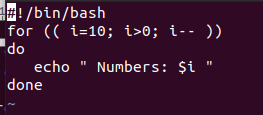


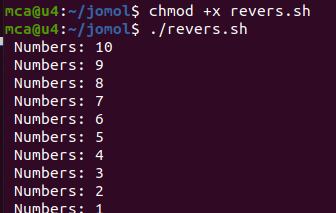
4.Write a shell script to display numbers upto 10



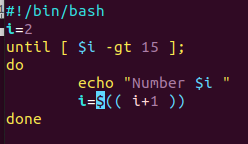


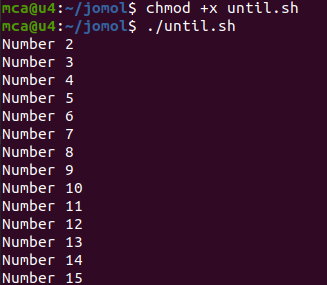
5.Write a shell script to display numbers in reverse direction





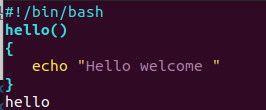
6.Write a shell script to display numbers from 2 to 15 using until loop.

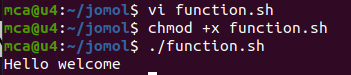




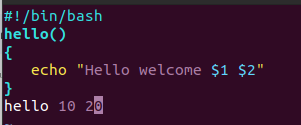
**FUNCTION**

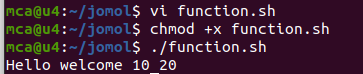
1.Display Hello World

****

****

2.parameterised function





04-07-23

Make and CMake

Main.cpp

#include<iostream>

#include 'fuction.h'

int main()

print\_hello();

std::cout << std::endl;

std::cout << 'The factorial of 5 is'<<factorial(5)<<std::endl;

return 0;

Function.h

void print\_hello();

void factorial(int n);

Function1.cpp

#include "function.h"

int factorial(int n){

if(n!=1){

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

}

else return 1;

}

Function2.cpp

#include<iostrem>

#include "function.h"

void print\_hello(){

std::cout << 'hello world';

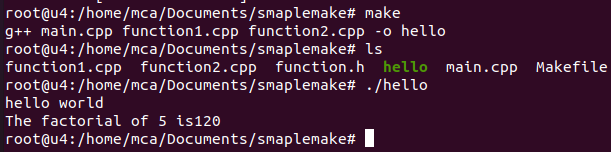
}

Makefile

all:

g++ main.cpp function1.cpp function2.cpp -o hello

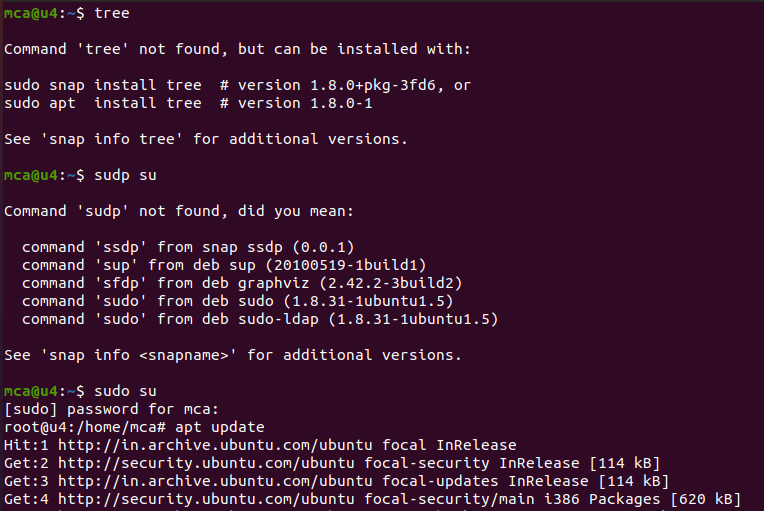
**Output:**

****

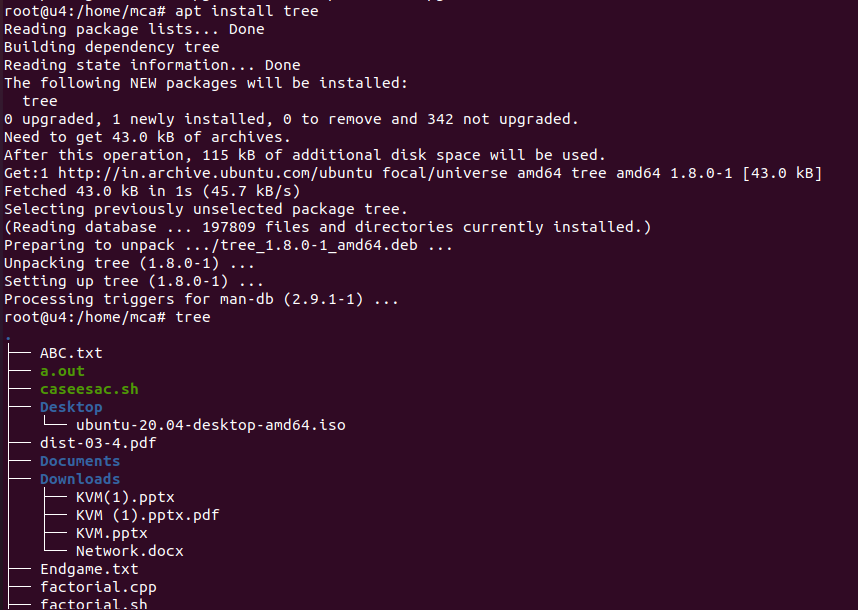
**10-07-23**

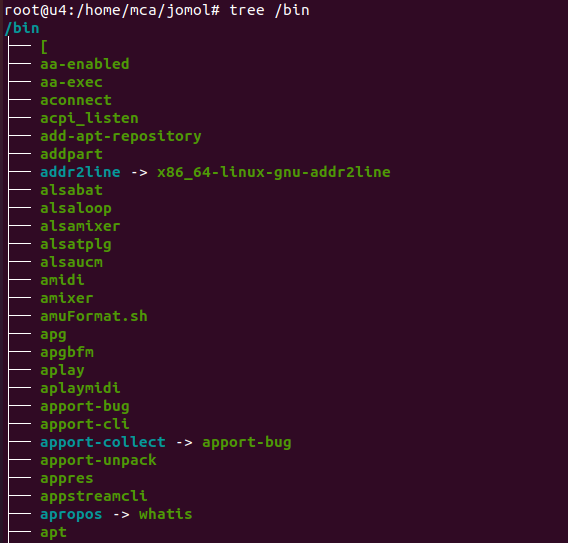
**Aim:- File System Hierarchy**

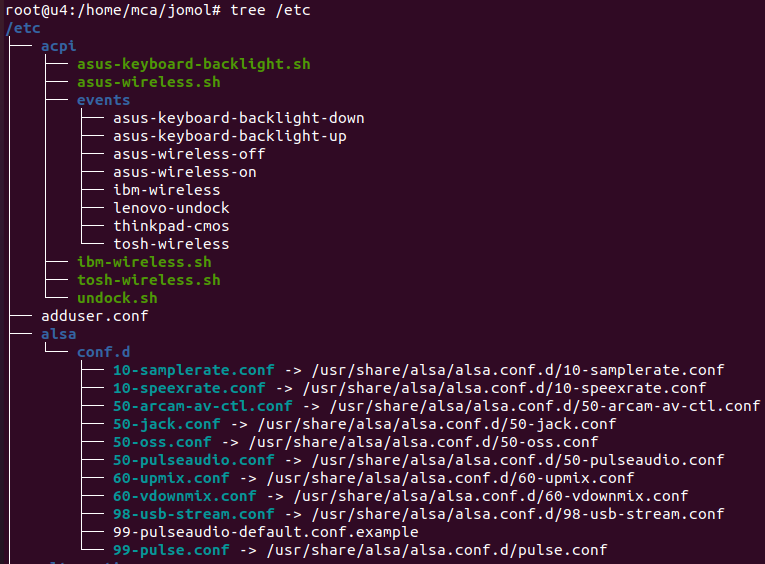
Tree

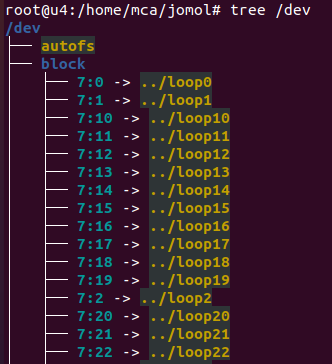


Apt install tree









**Cmake**

main.cpp

#include<iostream>

#include "add.h"

int main()

{

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

return 0;

}

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.16.3)

project("Hello world")

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

add.cpp

#include "add.h"

int add(int a, int b)

{

return a + b;

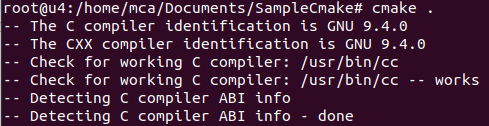
}

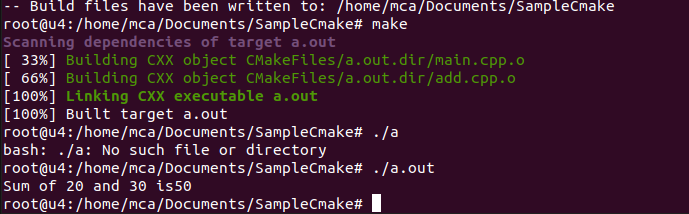
add.h

#pragma once

int add(int a, int b);

**OUTPUT:**

****

****