OPERATING SYSTEMS LAB FILE ETCS – 352

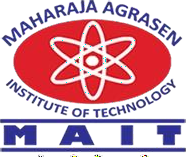
Faculty: Dr. Sandeep Tayal

Name: Muskan

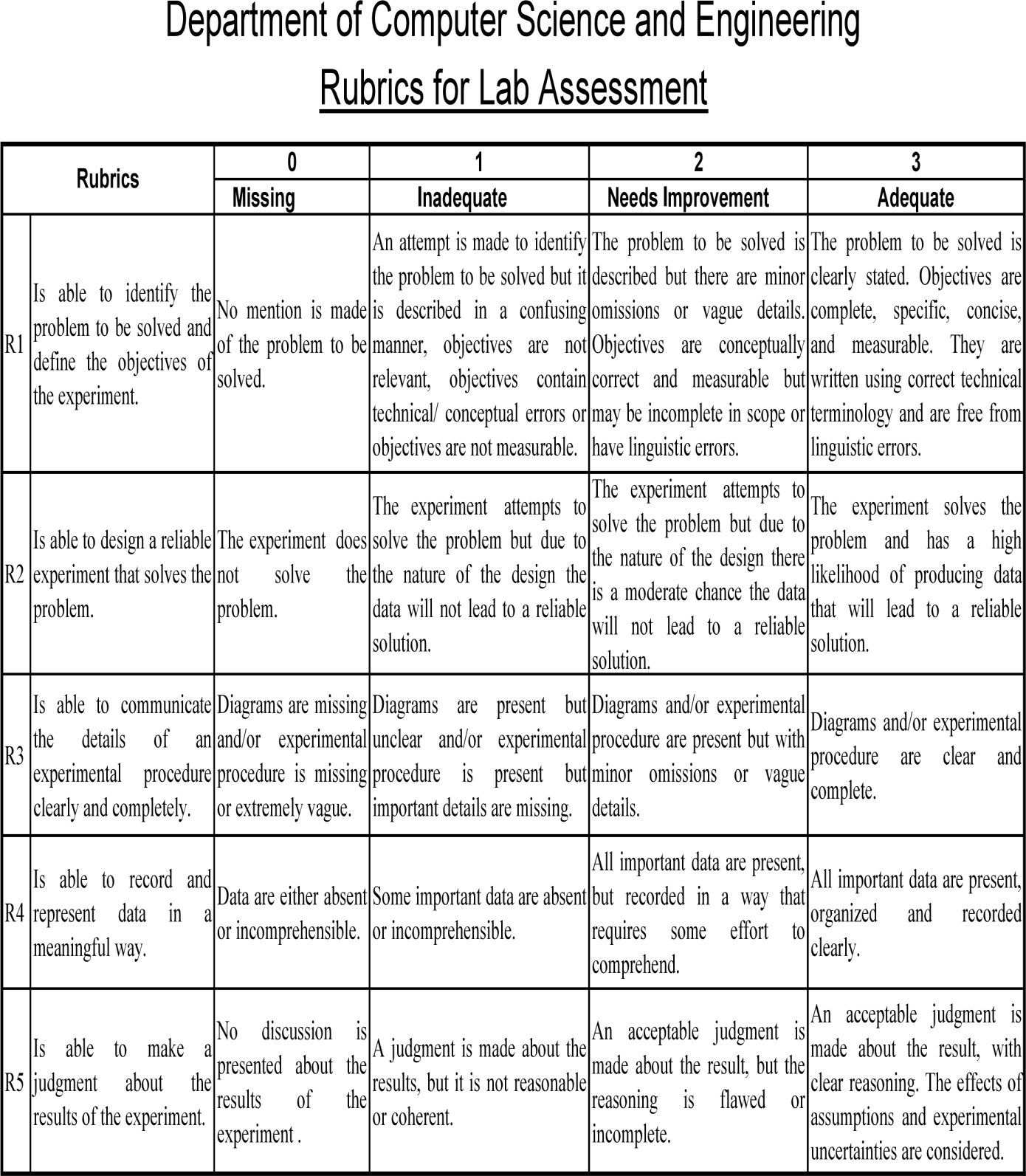
Enroll No: 02614802719

Group: 6C2

Branch: CSE



Maharaja Agrasen Institute of Technology, PSP Area, Sector – 22, Rohini, New Delhi – 110085

1. 

### Operating System Lab ETCS- 352

1. Student Name: Muskan Group: 6C2 Roll no-: 02614802719

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No | Experiment  Name |  | | | | |  | Signature | Date |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |

1. Overall Comments:
3. Signature
4. Faculty Name: Dr Sandeep Tayal

**1. Introduction to Linux and Installation of Linux Operating System**

## Introduction

Linux is a family of free and open-source operating systems based on the Linux kernel. Operating systems based on Linux are known as Linux distributions or distros. Examples include Debian, Ubuntu, Fedora, CentOS, Gentoo, Arch Linux, and many others.

The Linux kernel has been under active development since 1991, and has proven to be extremely versatile and adaptable. You can find computers that run Linux in a wide variety of contexts all over the world, from web servers to cell phones. Today, 90% of all cloud infrastructure and 74% of the world’s smartphones are powered by Linux.

However, newcomers to Linux may find it somewhat difficult to approach, as Linux filesystems have a different structure than those found on Windows or MacOS. Additionally, Linux-based operating systems depend heavily on working with the command line interface, while most personal computers rely on graphical interfaces.

This guide serves as an introduction to important command line concepts and skills and equips newcomers to learn more about Linux.

We need access to a computer running a Linux-based operating system. This can either be a virtual private server which you’ve connected to with SSH or your local machine. Note that this tutorial was validated using a Linux server running Ubuntu 20.04, but the examples given should work on a computer running any version of any Linux distribution.

### Linux Distributions

Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software. And you can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers, etc. Around 600 + Linux Distributions are available and some of the popular Linux distributions are:

* MX Linux
* Manjaro
* Linux Mint
* elementary
* Ubuntu
* Debian
* Solus
* Fedora

## Installation of Ubuntu 20.04

### System Requirements

Ubuntu 20.04 Server Edition provides a common, minimalist base for a variety of server applications, such as file/print services, web hosting, email hosting, etc. This version supports four 64-bit architectures:

* amd64 (Intel/AMD 64-bit)
* arm64 (64-bit ARM)
* ppc64el (POWER8 and POWER9)
* s390x (IBM Z and LinuxONE)

### Download Installation Media

**Step 1.** Download the Installation Media, In a web browser, visit the Ubuntu download page and pick the Ubuntu version suitable for your machine. The most popular versions include:

1. Ubuntu Desktop
2. Ubuntu Server
3. Ubuntu Derivatives

**Step 2.** Once we find the version you need, click the green Download button. You’ll be taken to a thank-you page, and your download should start. (We will download and install Ubuntu 20.04 for desktops.) The download is an .iso file. We can use it to create a bootable USB drive.

**Step 3.** Save the file to a location of your choice.

### Create Bootable USB

Use Balena Etcher to burn the .iso file to the USB drive.

1. Open a search dialog, and type create startup.
2. If it’s not installed, the Software Center will offer the option to install it – choose the option for USB drive, then open the utility.
3. In the top pane, click Other, then browse and select the Ubuntu 20.04 .iso file you downloaded.
4. In the bottom pane, select your USB drive.
5. Click Make startup disk.

### Boot up Ubuntu from USB

1. Turn off your system. Make sure you remove all other USB devices, such as printers, memory cards, etc.

2. Insert the Ubuntu USB drive into the system and turn on your machine.

There are two possible scenarios:

The computer boots the USB drive automatically.

You need to manually configure USB booting in the Boot Menu or BIOS/UEFI.

3. To manually configure the boot order, tap the boot menu key about once or twice per second as soon as the computer powers on.

### Install Ubuntu Using the guided installer

1. Click install Ubuntu
2. Select Language and Keyboard Layout.
   1. Normal Installation – This is the full Ubuntu Desktop experience, with office software, games, and media players.
   2. Minimal Installation – Choose this to save disk space, especially if you won’t be using media players or productivity software.
3. Select disk partition for installation
4. Select timezone
5. Create user account and password
6. Click continue and wait till the installation completes successfully.
7. Restart the PC and start using Ubuntu 20.04

**2. Introduction of basic files, directories, system Commands of Linux.**

## 

## Linux Files

A Linux file system is a structured collection of files on a disk drive or a partition. A partition is a segment of memory and contains some specific data. In our machine, there can be various partitions of the memory. Generally, every partition contains a file system.

The general-purpose computer system needs to store data systematically so that we can easily access the files in less time. It stores the data on hard disks (HDD) or some equivalent storage type. There may be below reasons for maintaining the file system:

* Primarily the computer saves data to the RAM storage; it may lose the data if it gets turned off. However, there is non-volatile RAM (Flash RAM and SSD) that is available to maintain the data after the power interruption.
* Data storage is preferred on hard drives as compared to standard RAM as RAM costs more than disk space. The hard disk costs are dropping gradually compared to the RAM.

The Linux file system contains the following sections:

* The root directory (/)
* A specific data storage format (EXT3, EXT4, BTRFS, XFS and so on)
* A partition or logical volume having a particular file system.

Linux file system is generally a built-in layer of a Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file.

If we have an unsupported file format in our file system, we can download software to deal with it.

Linux file system has a hierarchal file structure as it contains a root directory and its subdirectories. All other directories can be accessed from the root directory. A partition usually has only one file system, but it may have more than one file system.

A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology. The namespace defines the naming process, length of the file name, or a subset of characters that can be used for the file name. It also defines the logical structure of files on a memory segment, such as the use of directories for organizing the specific files. Once a namespace is described, a Metadata description must be defined for that particular file.

The data structure needs to support a hierarchical directory structure; this structure is used to describe the available and used disk space for a particular block. It also has the other details about the files such as file size, date & time of creation, update, and last modified.

Also, it stores advanced information about the section of the disk, such as partitions and volumes.

The advanced data and the structures that it represents contain the information about the file system stored on the drive; it is distinct and independent of the file system metadata.

## Linux Directories

1. General Files – It is also called ordinary files. It may be an image, video, program, or simple text files. These types of files can be in ASCII or Binary format. It is the most commonly used file in the Linux system.
2. Directory Files – These types of files are a warehouse for other file types. It may be a directory file within a directory (subdirectory).
3. Device Files – In a Windows-like operating system, devices like CD-ROM, and hard drives are represented as drive letters like F: G: H whereas in the Linux system device are represented as files. As for example, /dev/sda1, /dev/sda2 and so on.

These are the common top-level directories associated with the root directory:

* /bin – binary or executable programs.
* /etc – system configuration files.
* /home – home directory. It is the default current directory.
* /opt – optional or third-party software.
* /tmp – temporary space, typically cleared on reboot.
* /usr – User related programs.
* /var – log files.

Some other directories in the Linux system:

* /boot- It contains all the boot-related information files and folders such as conf, grub, etc.
* /dev – It is the location of the device files such as dev/sda1, dev/sda2, etc.
* /lib – It contains kernel modules and a shared library.
* /lost+found – It is used to find recovered bits of corrupted files.
* /media – It contains subdirectories where removal media devices inserted.
* /mnt – It contains temporary mount directories for mounting the file system.
* /proc – It is a virtual and pseudo-file system to contains info about the running processes with a specific process ID or PID.
* /run – It stores volatile runtime data.
* /sbin – binary executable programs for an administrator.
* /srv – It contains server-specific and server-related files.
* /sys – It is a virtual filesystem for modern Linux distributions to store and allows modification of the devices connected to the system.

## System Commands of Linux

1. **curl:** curl transfers a URL. Use this command to test an application's endpoint or connectivity to an upstream service endpoint. curl can be useful for determining if your application can reach another service, such as a database, or checking if your service is healthy.

2. **python -m json.tool / jq:** After you issue curl, the output of the API call may be difficult to read. Sometimes, you want to pretty-print the JSON output to find a specific entry. Python has a built-in JSON library that can help with this. You use python -m json.tool to indent and organize the JSON. To use Python's JSON module, pipe the output of a JSON file into the python -m json.tool command.

3. **ls:** ls lists files in a directory. Sysadmins and developers issue this command quite often. In the container space, this command can help determine your container image's directory and files. Besides looking up your files, ls can help you examine your permissions. In the example below, you can't run myapp because of a permissions issue. When you check the permissions using ls -l, you realize that the permissions do not have an "x" in -rw-r--r--, which are read and write only.

4. **tail:** tail displays the last part of a file. You usually don't need every log line to troubleshoot. Instead, you want to check what your logs say about the most recent request to your application. For example, you can use tail to check what happens in the logs when you make a request to your Apache HTTP server.

5. **cat:** cat concatenates and prints files. You might issue cat to check the contents of your dependencies file or to confirm the version of the application that you have already built locally.

6. **grep:** grep searches file patterns. If you are looking for a specific pattern in the output of another command, grep highlights the relevant lines.

7. **ps:** The ps command, part of the procps-ng package which provides useful commands for investigating process IDs, shows the status of a running process.

8. **env:** env allows you to set or print the environment variables.

9. **top:** top displays and updates sorted process information. Use this monitoring tool to determine which processes are running and how much memory and CPU they consume.

10. **netstat:** netstat shows the network status. This command shows network ports in use and their incoming connections.

**3. Write a script to find greatest of three numbers (numbers passed as command line parameters)**

## 

## Script

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo $num1

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

then

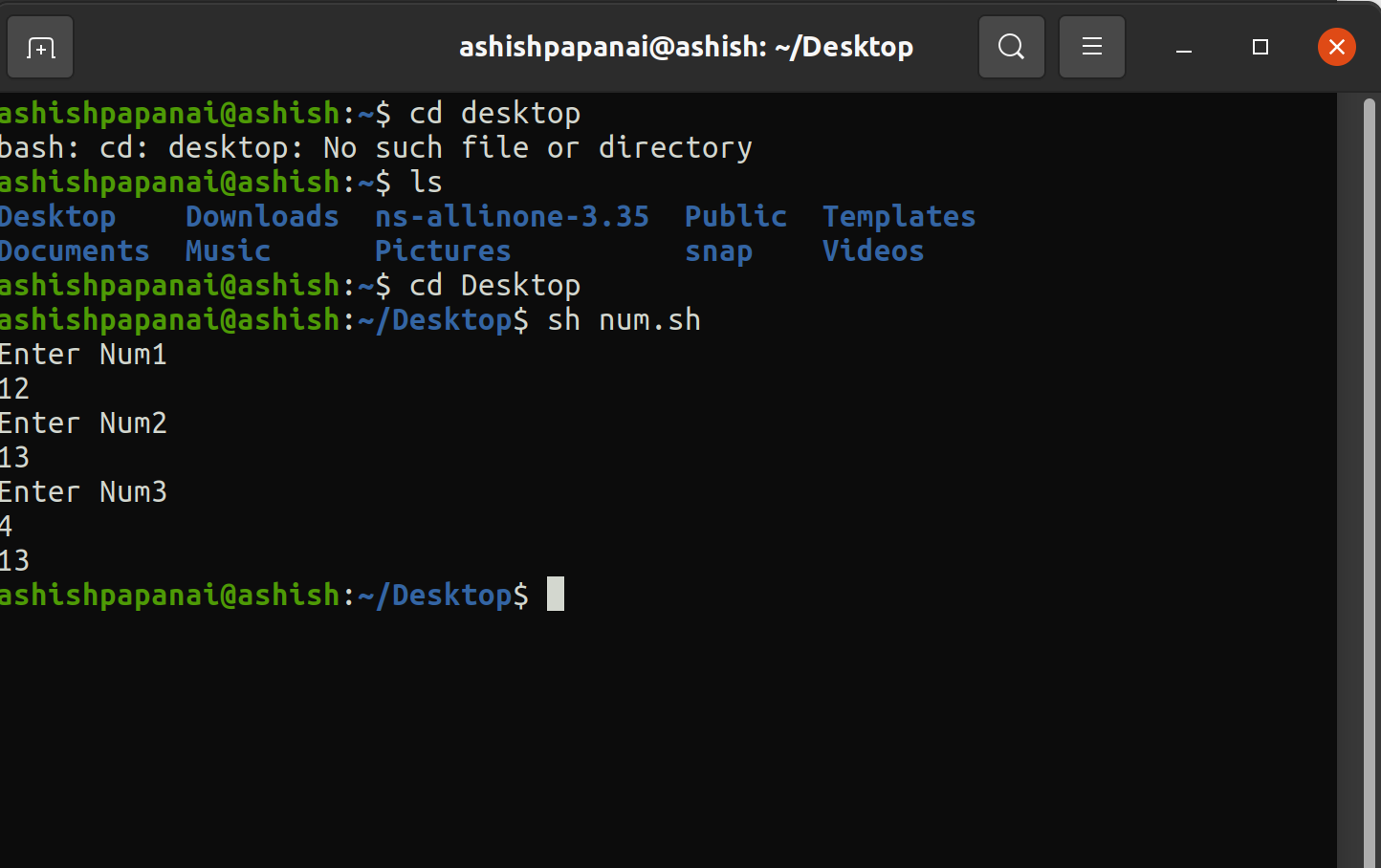
echo $num2

else

echo $num3

fi

## Output



**4. Write a script to check whether the given number is even/odd.**

## Script

read -p "Enter a number: " number

if [ $((number%2)) -eq 0]

then

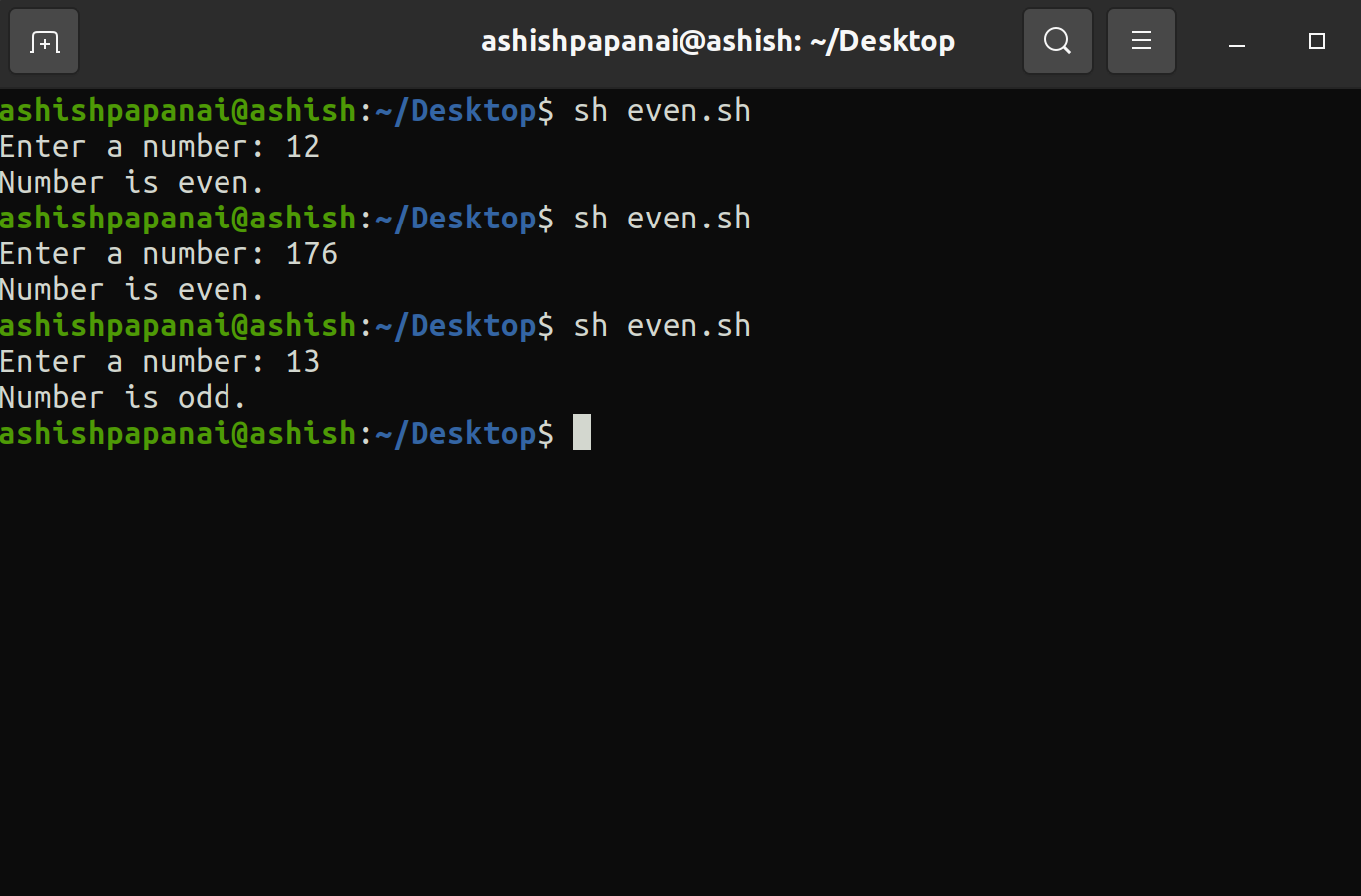
echo "Number is even."

else

echo "Number is odd."

fi

## Output



**5. Write a script to check whether the given number is prime or not.**

## Script

#!/bin/bash

echo -e "Enter Number : \c"

read n

for((i=2; i<=$n/2; i++))

do

ans=$(( n%i ))

if [ $ans -eq 0 ]

then

echo "$n is not a prime number."

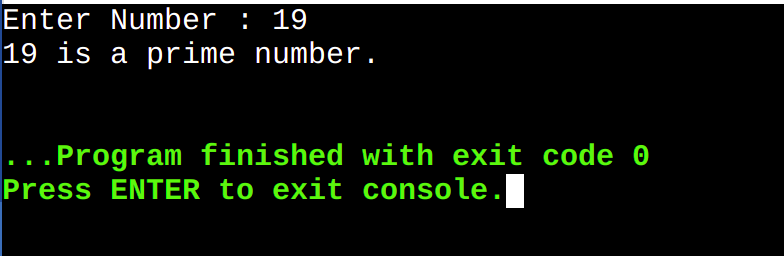
exit 0

fi

done

echo "$n is a prime number."

## Output



**6. Write a program to implement CPU scheduling for first come first serve.**

## Script

# Bash script to implement first come first served CPU scheduling algorithm.

sort(){

for ((i = 0; i<$n; i++))

do

for((j = 0; j<`expr $n - $i - 1`; j++))

do

if [ ${arrival\_time[j]} -gt ${arrival\_time[$((j+1))]} ]

then

# swap

temp=${arrival\_time[j]}

arrival\_time[$j]=${arrival\_time[$((j+1))]}

arrival\_time[$((j+1))]=$temp

temp=${burst\_time[j]}

burst\_time[$j]=${burst\_time[$((j+1))]}

burst\_time[$((j+1))]=$temp

temp=${pid[j]}

pid[$j]=${pid[$((j+1))]}

pid[$((j+1))]=$temp

elif [ ${arrival\_time[j]} -eq ${arrival\_time[$((j+1))]} ]

then

if [ ${pid[j]} -eq ${pid[$((j+1))]} ]

then

temp=${arrival\_time[j]}

arrival\_time[$j]=${arrival\_time[$((j+1))]}

arrival\_time[$((j+1))]=$temp

temp=${burst\_time[j]}

burst\_time[$j]=${burst\_time[$((j+1))]}

burst\_time[$((j+1))]=$temp

temp=${pid[j]}

pid[$j]=${pid[$((j+1))]}

pid[$((j+1))]=$temp

fi

fi

done

done

}

border(){

z=121

for ((i=0; i<$z; i++))

do

echo -n "-"

done

echo ""

}

findWaitingTime(){

service\_time[0]=0

waiting\_time[0]=0

for ((i=1; i<$n; i++))

do

z=1

y=`expr $i - $z`

service\_time[$i]=`expr ${service\_time[$y]} + ${burst\_time[$y]} `

waiting\_time[$i]=`expr ${service\_time[$i]} - ${arrival\_time[$i]}`

if [ ${waiting\_time[$i]} -lt 0 ]

then

waiting\_time[$i]=0

fi

done

}

findTurnAroundTime(){

for ((i=0; i<$n; i++))

do

tat[$i]=`expr ${waiting\_time[$i]} + ${burst\_time[$i]}`

done

}

findAverageTime(){

sort

findWaitingTime

findTurnAroundTime

total\_wt=0

total\_tat=0

border

printf "|%-18s|%-20s|%-18s|%-20s|%-18s|%-20s|\n" "Process Id" "Burst time" "Arrival time" "Waiting time" "Turn around time" "Completion time"

border

for ((i=0; i<$n; i++))

do

total\_wt=`expr $total\_wt + ${waiting\_time[$i]}`

total\_tat=`expr ${tat[$i]} + $total\_tat`

completion\_time=`expr ${arrival\_time[$i]} + ${tat[$i]}`

printf "|%-18s|%-20s|%-18s|%-20s|%-18s|%-20s|\n" ${pid[$i]} ${burst\_time[$i]} ${arrival\_time[$i]} ${waiting\_time[$i]} ${tat[$i]} $completion\_time

#echo "${burst\_time[$i]} ${arrival\_time[$i]} ${waiting\_time[$i]} ${tat[$i]} $completion\_time"

done

border

#avgwt=`echo "scale=3; $total\_wt / $n" | bc`

echo -n "Average waiting time ="

printf %.3f\\n "$(($total\_wt / $n))"

#avgtat=`echo "scale=3; $total\_tat / $n" | bc`

echo -n "Average turn around time ="

printf %.3f\\n "$(($total\_tat / $n))"

for ((i=0; i<8\*n+n+1; i++))

do

echo -n "-"

done

echo ""

for ((i=0; i<$n; i++))

do

echo -n "| "

echo -n "P${pid[$i]}"

echo -n " "

done

echo "|"

for ((i=0; i<8\*n+n+1; i++))

do

echo -n "-"

done

echo ""

echo -n "0 "

for ((i=0; i<$n; i++))

do

echo -n "`expr ${arrival\_time[$i]} + ${tat[$i]}`"

echo -n " "

done

echo ""

}

echo -n "Enter the number of processes: "

read n

for ((i=0; i<$n; i++))

do

echo -n "Enter Process Id: "

read pid[$i]

echo -n "Enter arrival time: "

read arrival\_time[$i]

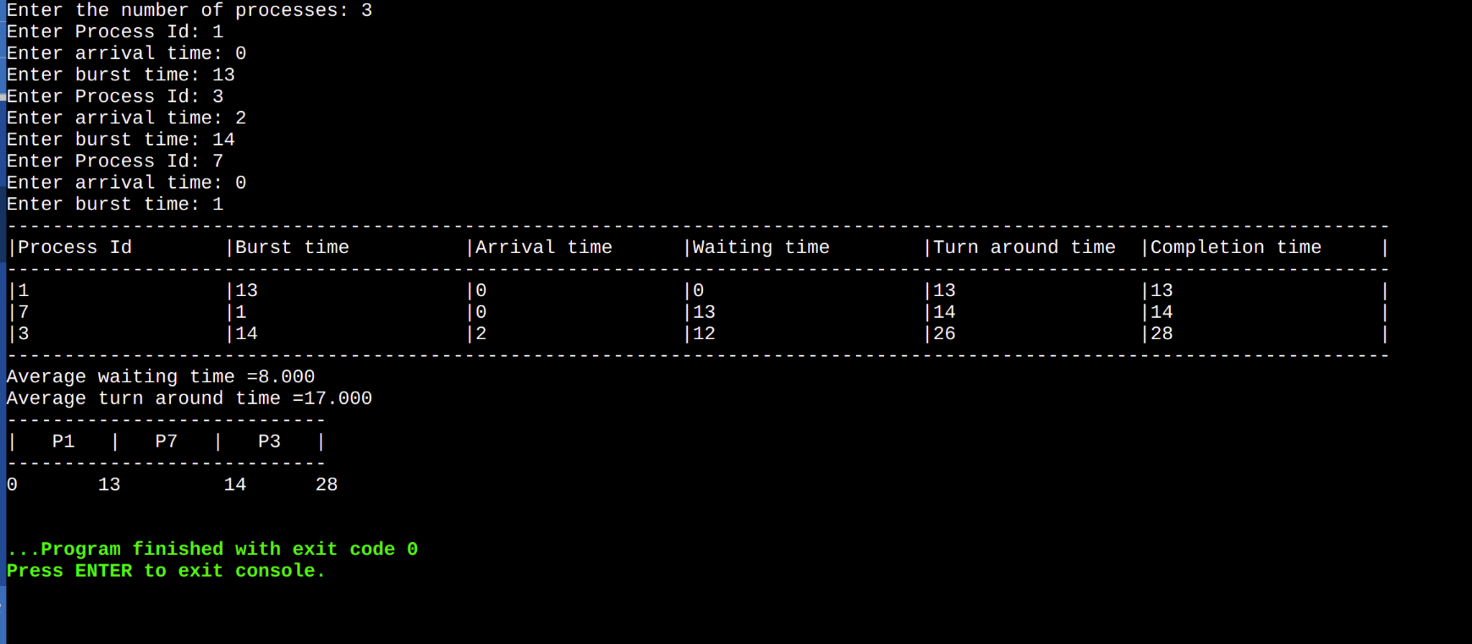
echo -n "Enter burst time: "

read burst\_time[$i]

done

findAverageTime

## Output



## OPERATING SYSTEMS LAB PRACTICAL RECORD

### PAPER CODE: ETCS - 352

Name of the student: Muskan

University Roll No: 02614802719

Group: 6C2

Branch: CSE

### PRACTICAL DETAILS

1. List of Experiments **beyond the syllabus** prescribed by GGSIPU

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp**  **No** | **Exp**  **Part** | **Experiment Name** | **Date of**  **Performance** | **Date of**  **Checking** | **Marks** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 | (i) |  |  |  |  |
| (ii) |  |  |  |  |
| (iii) |  |  |  |  |
| (iv) |  |  |  |  |
| (v) |  |  |  |  |
| 4 | (i) |  |  |  |  |
| (ii) |  |  |  |  |
| (iii) |  |  |  |  |
| (iv) |  |  |  |  |
| (v) |  |  |  |  |

# Experiment – 1

**Aim:** Installation of Linux Operating System.

**Theory:**

[Linux is an](https://www.educba.com/what-is-linux/) open source and free operating system to install which allows anyone with programming knowledge to modify and create its own operating system as per their requirements. Over many years, it has become more user-friendly and supports a lot of features such as

* + Reliable when used with servers
  + No need of antivirus
  + A Linux server can run nonstop with the boot for many years.

It has many distributions such as Ubuntu, [Fedora,](https://www.educba.com/install-fedora/) Redhat, Debian but all run on top of Linux server itself. Installation of every distribution is similar, thus we are explaining Ubuntu here. So let’s get started using this wonderful operating system by any of the following methods.

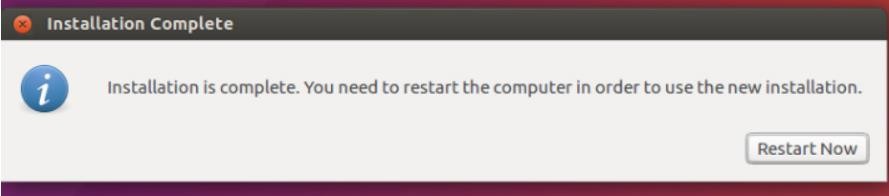
**Installing Linux Using Virtual Box VMWARE**

*What Are Requirements?*

* + Good internet connection
  + At least 4GB RAM
  + At least 12GB of free space

*Steps:*

* + Download the VIRTUAL BOX from original ORACLE VIRTUAL BOX site. You can refer below link <https://www.virtualbox.org/>
  + Use the .iso file or ISO file that can be downloaded from the internet and start the virtual box.
  + Choose an option under Create a virtual disk.
  + Choose a type of storage on physical hard disk. And choose the disk size (min 12 GB as per requirement)
  + Click on create option and then click on the START button to start the virtual box and browse to the location of the .iso file of the OS.
  + Now Linux OS will start, click on install option.
  + Select the drive for completing the OS installation. Select “Erase Disk and install Ubuntu” in case you want to replace the existing OS otherwise select “Something else” option and click INSTALL NOW.
  + Click on Continue.
  + Choose a username and password.
  + Installation will get completed in a while.



## Source:

[https://www.educba.com/install-](https://www.educba.com/install-linux/#%3A~%3Atext%3DB.%20Install%20Linux%20Using%20Virtual%20Box%20VMWARE) [linux/#:~:text=B.%20Install%20Linux%20Using%20Virtual%20Box%20VMWARE](https://www.educba.com/install-linux/#%3A~%3Atext%3DB.%20Install%20Linux%20Using%20Virtual%20Box%20VMWARE)

# Experiment – 2

**Aim:** Introduction to basic files, directories, system commands of Linux

**Theory:**

#### ls –

In Linux, the *ls* command is used to list out files and directories. Some versions may support color-coding.

$ ls -l filename

#### cd /var/log –

Change the current directory. The forward slash is to be used in Linux.

$ cd /var/log

#### su / sudo command –

*su* command changes the shell to be used as a super user and until you use the exit command you can continue to be the super user

*sudo* – if you just need to run something as a super user, you can use the *sudo* command. Example – shutdown command the shutdown command safely turns off the computer system.

* + *sudo shutdown 2* – shutdown and turns of the computer after 2 minutes
  + *sudo shutdown –r 2* – shuts down and reboots in 2 minutes

$ sudo **shutdown** 2

$ sudo **shutdown** –r 2

#### pwd – Print Working Directory

One way to identify the directory you are working in is the *pwd* command

It displays the current working directory path and is useful when directory changes are often

$ pwd

#### mv – Move a file

To move a file or rename a file you would use the *mv* command. Here the file name gets changed from *first.txt* to *second.txt.*

Type *ls* to view the change

$ mv first.txt second.txt

#### cp – Copy a file

*cp source file destination file*. In case you need a copy of the file *second.txt* in the same directory you have to use the *cp* command

$ cp second.txt third.txt

#### mkdir – to make a directory.

*mkdir [directory name]* if you would like to create a directory in the name ‘myproject’ type

*mkdir myproject*

$ **mkdir** myproject

#### echo –

This command is used to display a text or a string to the standard output or a file.

$ echo –e “This **is** an article **is for** beginners. \nIt **is on** basic linux com mands

Will display the output as

This **is** an article **is for** beginners.

It **is on** basic linux commands

#### clear –

This command lets you clear the terminal screen.

$ clear

#### apt –get

[*apt -get*](https://www.tecmint.com/useful-basic-commands-of-apt-get-and-apt-cache-for-package-management/) is a powerful and free front-end package manager for Debian/Ubuntu systems. It is used to install new software packages, remove available software packages, upgrade existing software packages as well as upgrade the entire operating system. apt – stands for advanced packaging tool.

**$ sudo apt-get update**

# Experiment – 3 (i)

**Aim:** Write a script to find the greatest of three numbers.

**Program:**

echo "Enter Num1" read num1

echo "Enter Num2" read num2

echo "Enter Num3" read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ] then

echo $num1

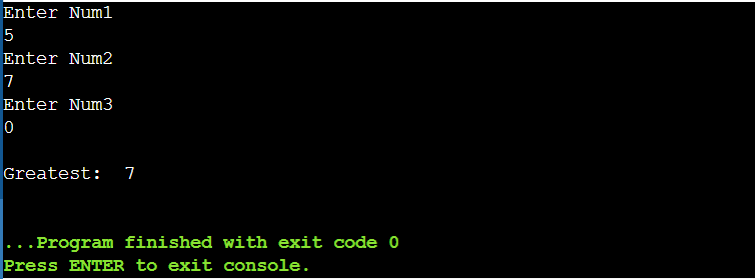
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ] then

echo $num2 else

echo $num3

fi

**Output:**



# Experiment – 3 (ii)

**Aim:** Write a script to check whether the given number is even or odd.

**Program:**

echo "Enter number:" read num

if ((num%2 == 0)); then echo

echo "$num is Even." else

echo

echo "$num is Odd."

fi

**Output:**



# Experiment – 3 (iii)

**Aim:** Write a script to check whether the given number is prime or not.

**Program:**

echo "Enter number" read num

function prime

{

for((i=2; i<=num/2; i++)) do

if [ $((num%i)) -eq 0 ] then

echo

echo "$num is not a prime number." exit

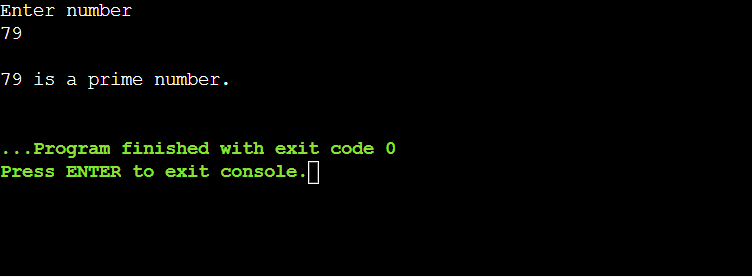
fi done echo

echo "$num is a prime number."

}

r=`prime $number` echo "$r"

**Output:**



# Experiment – 3 (iv)

**Aim:** Write a script to check whether the given input is a number or a string.

**Program:**

read -p "Type a number or a string: " input if [[ $input =~ ^[+-]?[0-9]+$ ]]; then

echo

echo "'$input' is an integer."

elif [[ $input =~ ^[+-]?[0-9]+\.$ ]]; then echo

echo "'$input' is a string."

elif [[ $input =~ ^[+-]?[0-9]+\.?[0-9]\*$ ]]; then echo

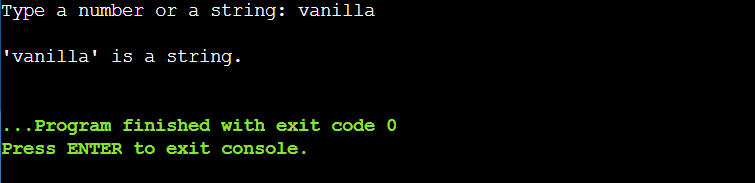
echo "'$input' is a float." else

echo

echo "'$input' is a string."

fi

**Output:**



# Experiment – 3 (v)

**Aim:** Write a script to compute number of characters and words in each line of a given file.

**Program:**

file\_path="C:/Users/kanis/OneDrive/Desktop/Sem6Y3/input.txt"

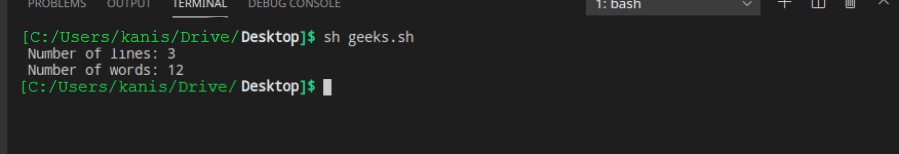
number\_of\_lines=`wc --lines < $file\_path` number\_of\_words=`wc --word < $file\_path`

echo "Number of lines: $number\_of\_lines" echo "Number of words: $number\_of\_words"

## Input File:



**Output:**



# Experiment – 4 (i)

**Aim:** Write a script to calculate the average of n numbers.

**Program:**

echo "Enter Size(N)" read N

i=1 sum=0

echo "Enter Numbers" while [ $i -le $N ] do

read num #get number sum=$((sum + num)) #sum+=num i=$((i + 1))

done

avg=`expr $sum / $N` echo

echo "Average($N): " $avg

**Output:**



# Experiment – 4 (ii)

**Aim:** Write a script to print the Fibonacci series upto n terms.

**Program:**

echo "How many number of terms to be generated ?" read n

function fib

{

x=0 y=1 i=2 echo

echo "Fibonacci Series up to $n terms :" echo "$x"

echo "$y"

while [ $i -lt $n ] do

i=`expr $i + 1 ` z=`expr $x + $y ` echo "$z"

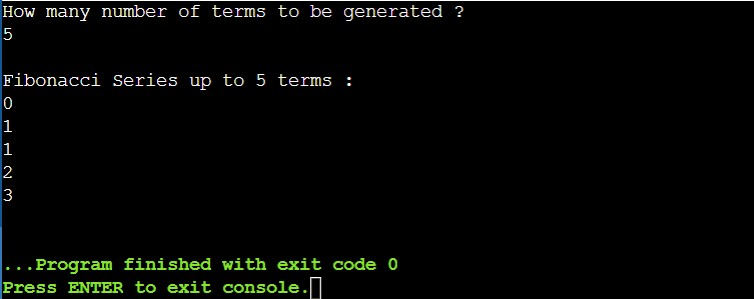
x=$y y=$z

done

}

r=`fib $n` echo "$r"

**Output:**



# Experiment – 4 (iii)

**Aim:** Write a script to calculate the factorial of a given number.

**Program:**

echo "Enter a number" read num

save=$num fact=1

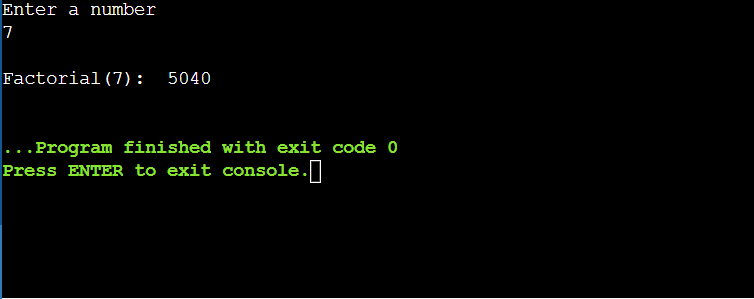
while [ $num -gt 1 ] do

fact=$((fact \* num)) #fact = fact \* num num=$((num - 1)) #num = num - 1

done echo

echo "Factorial($save): " $fact

**Output:**



# Experiment – 4 (iv)

**Aim:** Write a script to calculate the sum of digits of a given number.

**Program:**

echo "Enter a number" read num

save=$num sum=0

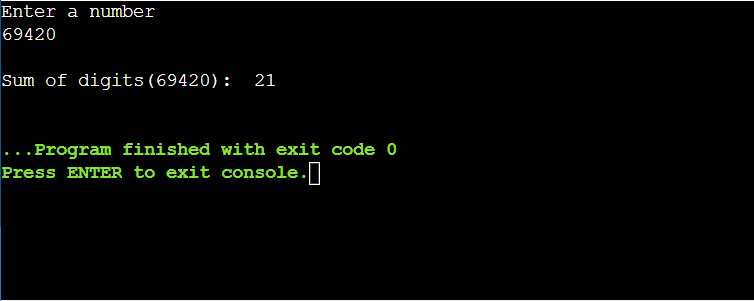
while [ $num -gt 0 ] do

mod=$((num % 10)) #It will split each digits sum=$((sum + mod)) #Add each digit to sum num=$((num / 10)) #divide num by 10.

done echo

echo "Sum of digits($save): " $sum

**Output:**



# Experiment – 4 (v)

**Aim:** Write a script to check whether the string is a palindrome.

**Program:**

echo "Enter a String" read input reverse="" len=${#input}

for (( i=$len-1; i>=0; i-- )) do

reverse="$reverse${input:$i:1}"

done

if [ $input == $reverse ] then

echo

echo "'$input' is a palindrome." else

echo

echo "$input is not palindrome."

fi

## Output:

