

## Program No. 1

### Booting Process of Linux

Press the power button on your system, and after few moments you see the Linux login prompt. From the time you press the power button until the Linux login prompt appears, the following sequence occurs. The following are the 6 high level stages of a typical Linux boot process.

<b>BIOS</b>	Basic input / output system executes MBR
<b>MBR</b>	Master Boot Record executes GRUB
<b>GRUB</b>	Grand Unified Bootloader executes Kernel
<b>Kernel</b>	Kernel executes the /sbin/init program
<b>Init</b>	Init executes Runlevel programs
<b>Runlevel</b>	Runlevel programs are executes from /etc/rc.d/rc*.d/

#### Step-1. BIOS

- BIOS stands for Basic Input/Output System.
- Performs some system integrity checks.
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, CD-ROMs, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

#### Step-2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda.
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

#### Step-3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.

- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz hiddenmenu
title CentOS (2.6.18-194.el5PAE)
root(hd0,0)
kernel/boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

#### Step-4. Kernel

- Mounts the root file system as specified in the “root=” in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

#### Step-5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
  - 0 – halt
  - 1 – Single user mode
  - 2 – Multiuser, without NFS
  - 3 – Full multiuser mode
  - 4 – unused
  - 5 – X11
  - 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate programs.

- Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

#### Step-6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail .... OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
  - Run level 0 – /etc/rc.d/rc0.d/
  - Run level 1 – /etc/rc.d/rc1.d/
  - Run level 2 – /etc/rc.d/rc2.d/
  - Run level 3 – /etc/rc.d/rc3.d/
  - Run level 4 – /etc/rc.d/rc4.d/
  - Run level 5 – /etc/rc.d/rc5.d/
  - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc\*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

#### Login Process

- i. Users enter their username and password
- ii. The operating system confirms your name and password.
- iii. A "shell" is created for you based on your entry in the "/etc/passwd" file
- iv. You are "placed" in your "home"directory.
- v. Start-up information is read from the file named "/etc/profile". This file is known as the system login file. When every user logs in, they read the information in this file.
- vi. Additional information is read from the file named ".profile" that is located in your "home" directory. This file is known as your personal login file.

**Result-** The given program has been performed successfully.

## **Program No. 2**

### **Basic Linux Commands**

#### **1. Basics**

- i. echo SRM: to display the string SRM
- ii. clear: to clear the screen
- iii. date: to display the current date and time
- iv. cal 2022: to display the calendar for the year 2022
- v. cal 6 2022: to display the calendar for the June-2022
- vi. passwd: to change password

#### **2. Working with Files**

- i. ls: list files in the present working directory
- ii. ls -l: list files with detailed information (long list)
- iii. ls -a: list all files including the hidden files
- iv. cat > f1: to create a file (Press ^d to finish typing)
- v. cat f1: display the content of the file f1
- vi. wc f1: list no. of characters, words & lines of a file
- vii. f1 wc -c f1: list only no. of characters of file
- viii. f1 wc -w f1: list only no. of words of file f1
- ix. wc -l f1: list only no. of lines of file f1
- x. cp f1 f2: copy file f1 into f2
- xi. mv f1 f2: rename file f1 as f2
- xii. rm f1: remove the file f1
- xiii. head -5 f1: list first 5 lines of the file f1
- xiv. tail -5 f1: list last 5 lines of the file f1

#### **3. Working with Directories**

- i. mkdir elias: to create the directory elias
- ii. cd elias: to change the directory as elias
- iii. rmdir elias: to remove the directory elias
- iv. pwd: to display the path of the present working directory
- v. cd: to go to the home directory
- vi. cd ..: to go to the parent directory
- vii. cd -: to go to the previous working directory
- viii. cd /: to go to the root directory

#### **4. File name substitution**

- i. ls f?: list files start with 'f' and followed by any one character
- ii. ls \*.c: list files with extension 'c'
- iii. ls [gpy]et: list files whose first letter is any one of the character g, p or y and followed by the word et

- iv. ls [a-d,l-m]ring: list files whose first letter is any one of the character from a to d and l to m and followed by the word ring.

## 5. I/O Redirection

### *Input redirection*

wc -l < ex1:

To find the number of lines of the file ‘ex1’

### *Output redirection*

who > f2:

the output of ‘who’ will be redirected to file f2

cat >> f1:

to append more into the file f1

## 6. Piping

Syntax: Command1 | command2

Output of the command1 is transferred to the command2 as input. Finally output of the command2 will be displayed on the monitor.

ex. cat f1 | more: list the contents of file f1 screen by screen

head -6 f1 |tail -2: prints the 5<sup>th</sup>& 6<sup>th</sup> lines of the file f1.

## 7. Environment variables

echo \$HOME display the path of the home directory

echo \$PS1 display the prompt string \$

echo \$PS2 display the second prompt string ( > symbol by default )

echo \$LOGNAME login name

echo \$PATH list of pathname where the OS searches for an executable file

## 8. File Permission

-- chmod command is used to change the access permission of a file.

Method-1

Syntax : chmod [ugo] [+/-] [ rwx ]  
filename

u : user, g : group, o : others

+ : Add permission - : Remove the permission r : read, w : write, x : execute, a : all permissions

ex. chmodug+rw f1

adding ‘read & write’ permissions of file f1 to both user and group members.

## Method-2

Syntax : chmod octnum file1

The 3 digit octal number is represented as follows

- first digit -- file permissions for the user
- second digit -- file permissions for the group
- third digit -- file permissions for others

Each digit is specified as the sum of following

4 – read permission, 2 – write permission, 1 – execute permission

ex. chmod 754 f1

it change the file permission for the file as follows

- read, write & execute permissions for the user ie;  $4+2+1 = 7$
- read, & execute permissions for the group members ie;  $4+0+1 = 5$
- only read permission for others ie;  $4+0+0 = 4$

## Output-

```
$ echo $RMIST
$RMIST
$ date
Sat May  6 04:21:46 IST 2023
$ echo $HOME
echo $HOME

/home/webmaster
export "PS1=$ "

$ echo $PATH
echo $PATH

/opt/swift/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
:/sbin:/bin:./:/usr/local/mozart/bin:/usr/local/Pawn/bin:/usr
/local/smlnj/bin:/usr/local/icon-v950/bin:
```

**Result-** The given program has been executed successfully.

## **Program No. 3**

### **Linux File System**

#### **Linux File System**

Linux File System or any file system generally is a layer which is under the operating system that handles the positioning of your data on the storage, without it; the system cannot know which file starts from where and ends where.

Linux offers many file systems types like:

- Ext: an old one and no longer used due to limitations.
- Ext2: first Linux file system that allows 2 terabytes of data allowed.
- Ext3: came from Ext2, but with upgrades and backward compatibility.
- Ext4: faster and allows large files with significant speed.(Best Linux File System) It is a very good option for SSD disks and you notice when you try to install any Linux distro that this one is the default file system that Linux suggests.
- JFS: old file system made by IBM. It works very well with small and big files, but it failed and files corrupted after long time use, reports say.
- XFS: old file system and works slowly with small files.
- Btrfs: made by Oracle. It is not stable as Ext in some distros, but you can say that it is a replacement for it if you have to. It has a good performance.

#### **File System Structure**

The following table provides a short overview of the most important higher-level directories you find on a Linux system

Directory	Contents
/	Root directory—the starting point of the directory tree
/bin	Essential binary files. Binary Executable files
/boot	Static files of the boot loader.
/dev	Files needed to access host-specific devices.
/etc	Host-specific system configuration files.
/lib	Essential shared libraries and kernel modules.
/media	Mount points for removable media.
/mnt	Mount point for temporarily mounting a file system.
/opt	Add-on application software packages.
/root	Home directory for the superuser root.
/sbin	Essential system binaries.

/srv	Data for services provided by the system.
/tmp	Temporary files.
/usr	Secondary hierarchy with read-only data.
/var	Variable data such as log files

**Result-** The given program has been performed successfully.

## **Program No. 4**

### **Editors and Filters**

#### **VI EDITOR**

vi fname: to open the file fname

There are two types of mode in vi editor

Escape mode – used to give commands – to switch to escape mode, press <Esc> key

Command mode – used to edit the text – to switch to command mode, press any one the following inserting text command

#### **1. Inserting Text**

- i insert text before the cursor
- a append text after the cursor
- I insert text at the beginning of the line
- A append text to the end of the line
- r replace character under the cursor with the next character typed
- R Overwrite characters until the end of the line
- (small o) open new line after the current line to type text
- (capital O) open new line before the current line to type text

#### **2. Cursor Movements**

h left

j down

k up

l right

(The arrow keys usually work also)

^F forward one screen

^B back one screen

^D down half screen

^U up half screen

(^ indicates control key; case does not matter)

0 (zero) beginning of line

\$ end of line

#### **3. Deleting Text**

Note: (n) indicates a number, and is optional

dd: deletes current line

(n) dd: deletes (n) line(s) ex. 5dd deletes 5 lines

(n)dw: deletes (n) word(s)

D: deletes from cursor to end of line

X: deletes current character

(n)x: deletes (n) character(s)

X: deletes previous character

#### 4. Saving Files

:w- to save & resume editing (write & resume)

:wq- to save & exit (write & quit)

:q!- quit without save

#### 5. Cut, Copy and Paste

Yy: copies current line

(n) yy: copies (n) lines from the current line. ex. 4yy copies 4 lines.

p: paste deleted or yanked (copied) lines after the cursor

### FILTERS

#### 1. cut

Used to cut characters or fields from a file/input

Syntax: cut -cchars filename

-ffieldnos filename

By default, tab is the field separator (delimiter). If the fields of the files are separated by any other character, we need to specify explicitly by -d option

cut -ddelimitchar

-ffiledsfilename

#### 2. paste

Paste files vertically. That is n<sup>th</sup> line of first file and n<sup>th</sup> line of second file are pasted as the n<sup>th</sup> line of result

Syntax: paste file1 file2

-ddcharoption is used to paste the lines using the delimiting character dchar

-s option is used paste the lines of the file in a single line

#### 3. tr

Used to translate characters from standard input

Syntax: tr char1 char2 < filename

It translates char1 into char2 in file filename

Octal representation characters can also be used

Octal value	Character
-------------	-----------

'\7'	Bell
------	------

'\10'	Backspace
-------	-----------

'\11'	Tab
-------	-----

‘\12’ Newline  
‘\33’ Escape

Ex. `tr :'\11' < f1` translates all : into tab of file f1

**-s** Option translate multiple occurrences of a character by single character.

**-d** Option is to delete a character

## 4. grep

Used to search one or more files for a particular pattern.

Syntax: grep pattern filename(s)

Lines that contain the pattern in the file(s) get displayed pattern can be any regular expressions

More than one file can be searched for a pattern

**-v** option displays the lines that do not contain the pattern

**-l** list only name of the files that contain the pattern

**-n** displays also the line number along with the lines that matches the pattern

## 5. sort

Used to sort the file in order

Syntax      sort  
:            filename

Sorts the data as text by default

Sorts by the first field by default

-r	option sorts the file in descending order
-u	eliminates duplicate lines
-o filename	writes sorted data into the file fname
-tdchar	sorts the file in which fields are separated by dchar
-n	sorts the data as number
+1n	skip first filed and sort the file by second filed numerically

## 6. Unique

Displays unique lines of a sorted file

Syntax: uniq filename

**-d** option displays only the duplicate lines

**-c** displays unique lines with no. of occurrences.

## 7. cmp

Used to compare two files

Syntax: `cmp f1 f2` compare two files f1 & f2 and prints the line of first difference

8. diff

Used to differentiate two files

Syntax: diff f1 f2 compare two files f1 & f2 and prints all the lines that are differed between f1 & f2

## 9. comm

Used to compare two sorted files

Syntax: comm file1 file2

Three columns of output will be displayed.

- First column displays the lines that are unique to file1
- Second column displays the lines that are unique to file2
- Third column displays the lines that appears in both the files

-1	option suppress first column
-2	option suppress second column
-3	option suppress third column
-12	option display only third column
-13	option display only second column
-23	option display only first column

**Result-** The given program has been performed successfully.

## **Program No. 5**

### **Compilation of C Program**

#### **Compilation of C Program**

Step-1: Open the terminal and edit your program using vi editor/gedit editor and save with extension “.c”

Ex. vi test.c

(or) gedittext.c

Step-2: Compile your program using gcc compiler

Ex. gcctest.c            Output file will be “a.out”

(or) gcc -o test text.c    Output file will be “test”

Step-3: Correct the errors if any and run the program

Ex. ./a.out

or ./test

Optional Step: In order to avoid ./ prefix each time a program is to be executed, insert the following as the last line in the file .profile export PATH=.:\${PATH}

This Step needs only to be done once.

#### **Debug C Programs using gdb debugger**

Step-1: Compile C program with debugging option -g

Ex. gcc -g test.c

Step-2: Launch gdb. You will get gdb prompt

Ex. gdba.out

Step-3: Step break points inside C program

Ex. (gdb) b 10

Break points set up at line number 10. We can have any number of break points

Step-4: Run the program inside gdb

Ex. (gdb) r

Step-5: Print variable to get the intermediate values of the variables at break point

Ex. (gdb) p i Prints the value of the variable ‘i’

Step-6: Continue or stepping over the program using the following gdb commands

c continue till the next break

n Execute the next line. Treats function as single statement

s Similar to ‘n’ but executes function statements line by line

l List the program statements

Step-7: Quit the debugger

(gdb) q

**Result-** The given program has been performed successfully.

## **Program No. 6**

### **Process Creation**

#### **Syntax for process creation**

```
int fork();
```

Returns 0 in child process and child process ID in parent process.

#### **Other Related Functions**

int getpid()	returns the current process ID
int getppid()	returns the parent process ID
wait()	makes a process wait for other process to complete

#### **Virtual fork**

vfork() function is similar to fork but both processes shares the same address space.

Q1. Find the output of the following program

```
#include <stdio.h>
#include<unistd.h>

int main()
{
int a=5,b=10,pid;
printf("Before fork a=%d b=%d \n",a,b); pid=fork();

if(pid==0)
{
a=a+1; b=b+1;
printf("In child a=%d b=%d \n",a,b);
}
else
{
sleep(1); a=a-1; b=b-1;
printf("In Parent a=%d b=%d \n",a,b);
}
return 0;
}
```

## Output:

```
Activities Terminal Jun 16 02:52
aditya@aditya-VirtualBox:~$ gcc fork.c -o test
aditya@aditya-VirtualBox:~$ ./test
Before fork a=5 b=10
In child a=6 b=11
In Parent a=4 b=9
aditya@aditya-VirtualBox:~$
```

Q2. Calculate the number of times the text “SRMIST” is printed.

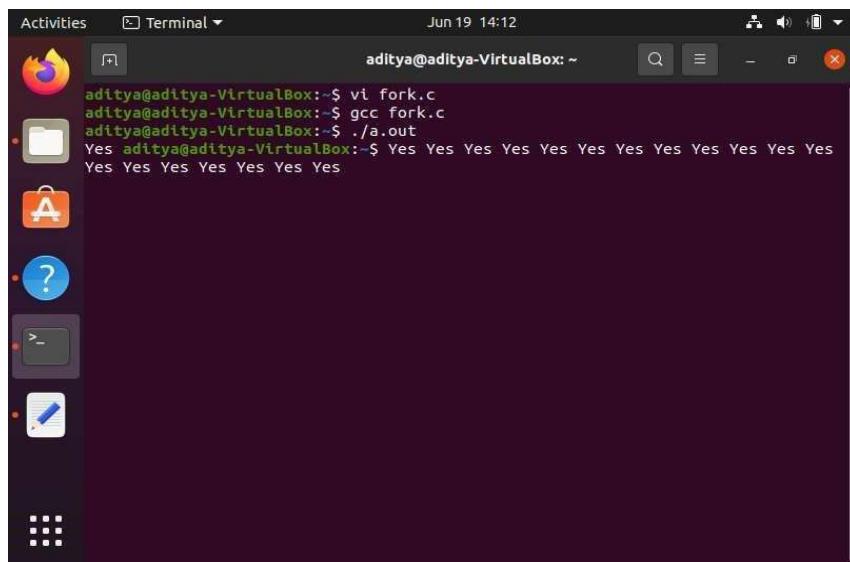
```
#include <stdio.h> #include <unistd.h>
int main()
{
    fork();        fork();
    fork();
    printf("SRMIST\n");
    return 0;
}
```

## Output:

Q3. How many child processes are created for the following code?

```
#include <stdio.h>
#include<unistd.h>
intmain()
{ fork();
fork()&&fork()||fork();
fork();
printf("Yes "); return 0;
}
```

**Output:**



The screenshot shows a terminal window in a Linux desktop environment. The terminal title is "aditya@aditya-VirtualBox: ~". The terminal window displays the following command-line session:

```
aditya@aditya-VirtualBox:~$ vi fork.c
aditya@aditya-VirtualBox:~$ gcc fork.c
aditya@aditya-VirtualBox:~$ ./a.out
Yes aditya@aditya-VirtualBox: ~$ Yes Yes
Yes Yes Yes Yes Yes Yes
```

The terminal window is part of a desktop interface with a dark theme. The desktop environment includes a dock with icons for a file manager, a terminal, a help center, a terminal, and a text editor.

**Result-** The given programs have been executed successfully.

## Program No. 7

### System Admin Commands

#### INSTALLING SOFTWARE

To Update the package repositories

```
sudo apt-get update
```

To update installed software

```
sudo apt-get upgrade
```

To install a package/software

```
sudo apt-get install <package-name>
```

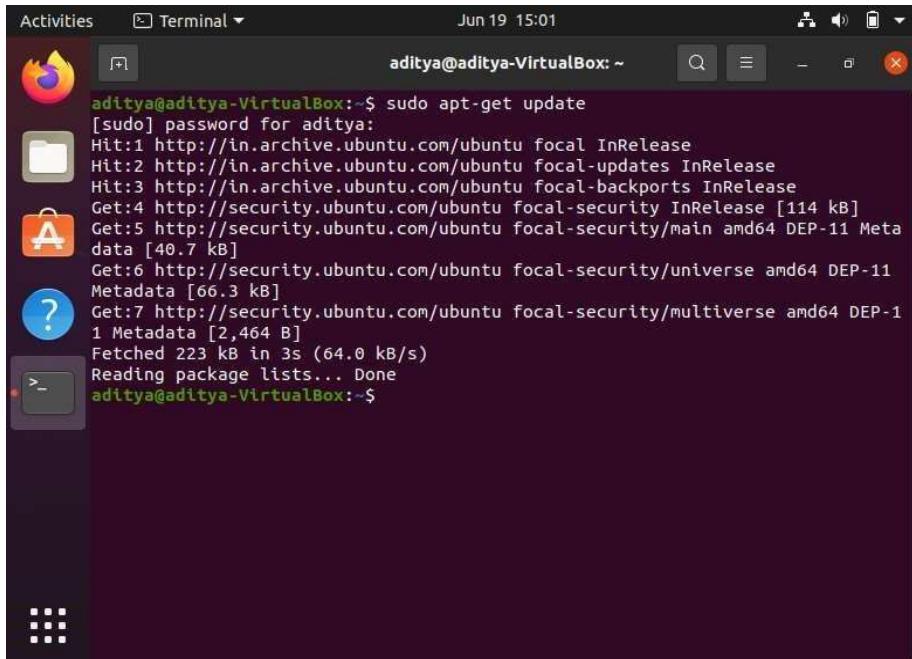
To remove a package from the system

```
sudo apt-get remove <package-name>
```

To reinstall a package

```
sudo apt-get install <package-name> --reinstall
```

Q1. Update the package repositories



The screenshot shows a terminal window on an Ubuntu desktop environment. The terminal title is 'Terminal' and the date and time are 'Jun 19 15:01'. The command entered is 'sudo apt-get update'. The output shows the system connecting to the Ubuntu repositories to check for updates. The process involves hitting multiple URLs to download package metadata. The output ends with 'Reading package lists... Done'.

```
aditya@aditya-VirtualBox:~$ sudo apt-get update
[sudo] password for aditya:
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [40.7 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [66.3 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Fetched 223 kB in 3s (64.0 kB/s)
Reading package lists... Done
aditya@aditya-VirtualBox:~$
```

## Q2. Install the package “simplescreenrecorder”

```
aditya@aditya-VirtualBox:~$ sudo apt-get install simplescreenrecorder
Reading package lists... Done
Building dependency tree...
Reading state information... Done
The following additional packages will be installed:
  i965-v4-driver intel-media-va-driver libaacs0 libao0 libavcodec58
  libavformat58 libavutil56 libbdplus libbluray2 libchromaprint1
  libcodec2-0.9 libdouble-conversion3 libgme0 libgsm1 libgpm1
  libopenmp0 libpcre2-16-0 libqtcore5a libqtdbus5 libqtgifs
  libqtNetworks libqtSvg5 libqtWidgets5 libqtX11Extras5 libshine3
  libsnappy5v libssh-gcrypt-4 libswresample3 libswscale5 libva-drm2
  libva-x11-2 libvdpau libvdpau1 libv2a-155 libv265-179 libxcb-xinerama0
  libxcb-xinput0 libxcbcore4 libzbv1-common libzbv1b mesa-va-drivers
  mesa-vdpau-drivers ocl-icd-libopencl1 qt5-gtk-platformtheme
  qttranslations5-l10n va-driver-all vdpau-driver-all
Suggested packages:
  i965-v4-driver-shaders libbluray-bd1 qt5-image-formats-plugins qtwayland
  opencl-icd libvdpau-va-glx nvidia-vdpau-driver
  nvidia-legacy-340xx-vdpau-driver nvidia-legacy-384xx-vdpau-driver
The following NEW packages will be installed:
  i965-v4-driver intel-media-va-driver libaacs0 libao0 libavcodec58
  libavformat58 libavutil56 libbdplus libbluray2 libchromaprint1
  libcodec2-0.9 libdouble-conversion3 libgme0 libgsm1 libgpm1
  libopenmp0 libpcre2-16-0 libqtcore5a libqtdbus5 libqtgifs
  libqtNetworks libqtSvg5 libqtWidgets5 libqtX11Extras5 libshine3
  libsnappy5v libssh-gcrypt-4 libswresample3 libswscale5 libva-drm2
  libva-x11-2 libvdpau libvdpau1 libv2a-155 libv265-179 libxcb-xinerama0
  libxcb-xinput0 libxcbcore4 libzbv1-common libzbv1b mesa-va-drivers
  mesa-vdpau-drivers ocl-icd-libopencl1 qt5-gtk-platformtheme
  qttranslations5-l10n simplescreenrecorder va-driver-all vdpau-driver-all
0 upgraded, 48 newly installed, 0 to remove and 160 not upgraded.
Need to get 39.1 MB of archives.
After this operation, 214 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 libdouble-conversion3 amd64 3.1.5-4ubuntu1 [37.9 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libpcre2-16-0 amd64 10.34-7 [181 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqtcore5a amd64 5.12.8+dfsg-2ubuntu1.1 [2,086 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/universe amd64 libavutil56 amd64 7.42.7-7ubuntu1.1 [241 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqtdbus5 amd64 5.12.8+dfsg-2ubuntu1.1 [288 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqtNetwork5 amd64 5.12.8+dfsg-2ubuntu1.1 [673 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libxcb-xinerama0 amd64 1.14-2 [5,268 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu focal/main amd64 libxcb-xinput0 amd64 1.14-2 [29.3 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqtgifs5 amd64 5.12.8+dfsg-2ubuntu1.2 [2,971 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqtSvg5 amd64 5.12.8+dfsg-2ubuntu1.2 [2,295 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 libzbv1-common [131 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 libzbv1b [53.5 kB]
```

## Q3. Remove the package “simplescreenrecorder”

```
aditya@aditya-VirtualBox:~$ sudo apt-get remove simplescreenrecorder
Reading package lists... Done
Building dependency tree...
Reading state information... Done
The following packages were automatically installed and are no longer required:
  i965-v4-driver intel-media-va-driver libaacs0 libao0 libavcodec58 libavformat58 libavutil56
  libbdplus0 libbluray2 libchromaprint1 libcodec2-0.9 libdouble-conversion3 libgme0 libgsm1 libgpm1
  libopenmp0 libpcre2-16-0 libqtcore5a libqtdbus5 libqtgifs5 libqtNetworks libqtSvg5 libqtWidgets5 libqtX11Extras5 libshine3 libsnappy5v libssh-gcrypt-4 libswresample3 libswscale5 libva-drm2
  libva-x11-2 libvdpau libvdpau1 libv2a-155 libv265-179 libxcb-xinerama0 libxcb-xinput0 libxcbvidcore4 libzbv1-common libzbv1b mesa-va-drivers mesa-vdpau-drivers ocl-icd-libopencl1 qt5-gtk-platformtheme
  qttranslations5-l10n va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  simplescreenrecorder
0 upgraded, 0 newly installed, 1 to remove and 160 not upgraded.
After this operation, 3,663 kB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 161859 files and directories currently installed.)
Removing simplescreenrecorder (0.3.11-1build2) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
```

## MANAGING USERS

- Managing users is a critical aspect of server management.
- In Ubuntu, the root user is disabled for safety.
- Root access can be completed by using the sudo command by a user who is in the “admin” group.
- When you create a user during installation, that user is added automatically to the admin group.

To add a user:

```
sudoadduser username
```

To disable a user:

```
sudopasswd -l username
```

To enable a user:

```
sudopasswd -u username
```

To delete a user:

```
sudouserdel -r username
```

To create a group:

```
sudoaddgroup groupname
```

To delete a group:

```
sudodelgroup groupname
```

To create a user with group:

```
sudoadduser username groupname
```

To see the password expiry value for a user,

```
sudochage -l username
```

To make changes:

```
sudochage username
```

## GUI Tool for User Management

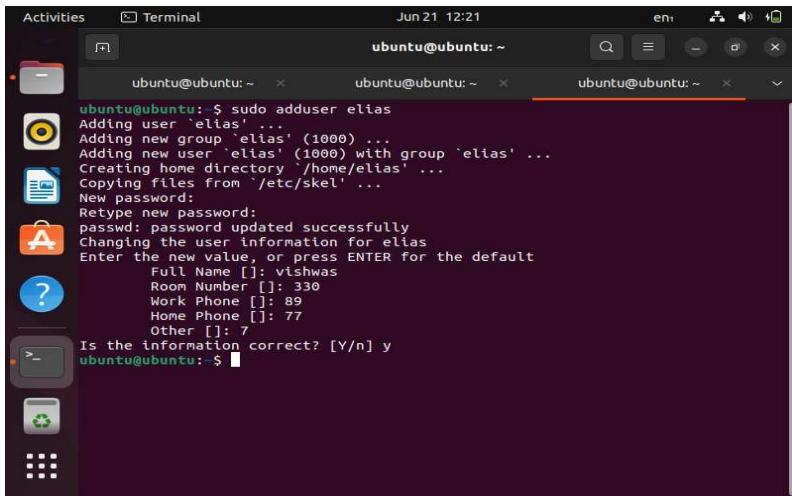
If you do not want to run the commands in terminal to manage users and groups, then you can install a GUI add-on.

```
sudo apt install gnome-system-tools
```

Once done, type

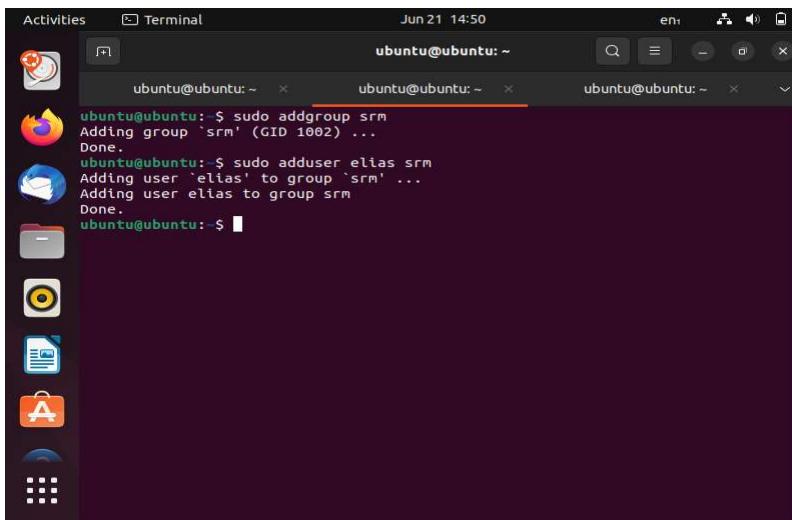
```
users-admin
```

Q4. Create a user 'elias'. Login to the newly created user and exit.



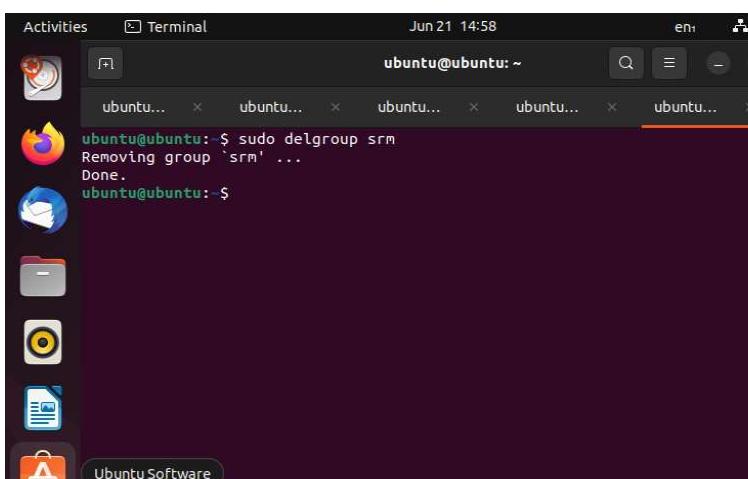
```
Activities Terminal Jun 21 12:21
ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x
ubuntu@ubuntu:~$ sudo adduser elias
Adding user 'elias'...
Adding new group 'elias' (1000) ...
Adding new user 'elias' (1000) with group 'elias' ...
Creating home directory '/home/elias' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for elias
Enter the new value, or press ENTER for the default
  Full Name []: vtshwas
  Room Number []: 330
  Work Phone []: 89
  Home Phone []: 77
  Other []: 7
Is the information correct? [Y/n] y
ubuntu@ubuntu:~$
```

Q5. Create a group 'cse' and add the user 'elias' in that group .



```
Activities Terminal Jun 21 14:50
ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x
ubuntu@ubuntu:~$ sudo addgroup srm
Adding group 'srm' (GID 1002) ...
Done.
ubuntu@ubuntu:~$ sudo adduser elias srm
Adding user 'elias' to group 'srm' ...
Adding user elias to group srm
Done.
ubuntu@ubuntu:~$
```

Q6. Delete the user 'elias' and then delete the group 'cse'.



```
Activities Terminal Jun 21 14:58
ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x ubuntu@ubuntu: ~ x
ubuntu@ubuntu:~$ sudo delgroup srm
Removing group 'srm' ...
Done.
ubuntu@ubuntu:~$
```

## FILE SYSTEM

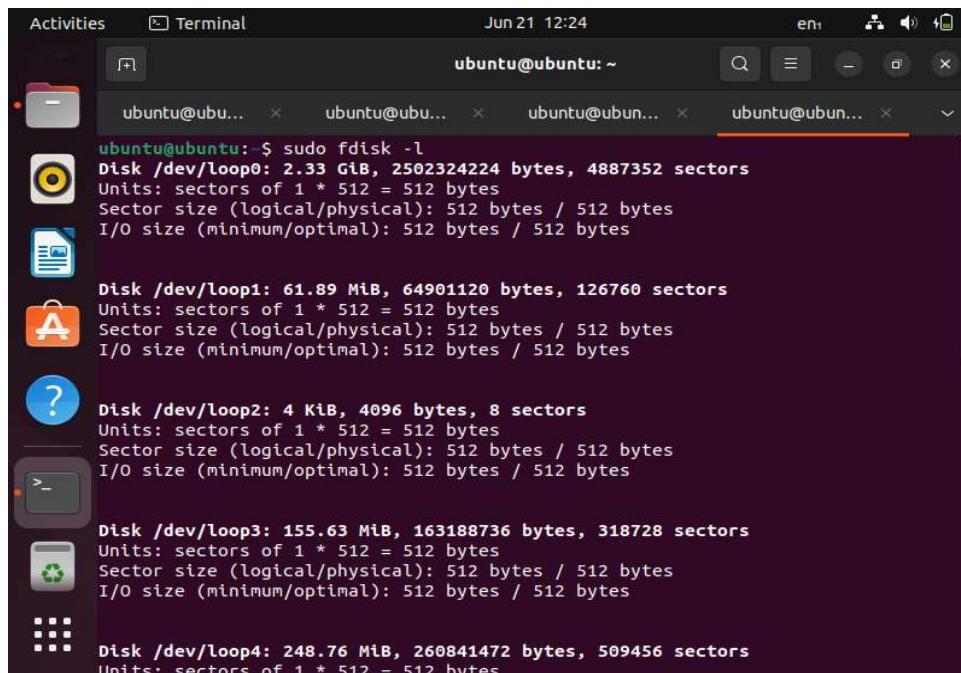
A file system is a permanent storage for containing data. Any non-volatile storage device like hard disk, usbtetc has a file system in place, on top of which data is stored. While installing Linux, you may opt for either EXT4 or EXT3 file system.

EXT3: A journaling filesystem: logs changes in a journal to increase reliability in case of power failure or system crash.

EXT4: It is an advanced file syste. This file system supports 64-bit storage limits, columns up to 1 exabytes and you may store files up to 16 terabytes

Disk Partitions can be viewed by the command `sudo fdisk -l` File system information are available in the file `/etc/fstab`

Q.7. List the partitions available in your system



```
Activities Terminal Jun 21 12:24
ubuntu@ubuntu:~$ sudo fdisk -l
Disk /dev/loop0: 2.33 GiB, 2502324224 bytes, 4887352 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop1: 61.89 MiB, 64901120 bytes, 126760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop2: 4 KiB, 4096 bytes, 8 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop3: 155.63 MiB, 163188736 bytes, 318728 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop4: 248.76 MiB, 260841472 bytes, 509456 sectors
Units: sectors of 1 * 512 = 512 bytes
```

## NETWORKING

Most networking is configured by editing two files:

- `/etc/network/interfaces`  
Ethernet, TCP/IP, bridging
- `/etc/resolv.conf` DNS  
Other networking files:
  - `/etc/hosts`
  - `/etc/dhcp3/dhcpd.conf`

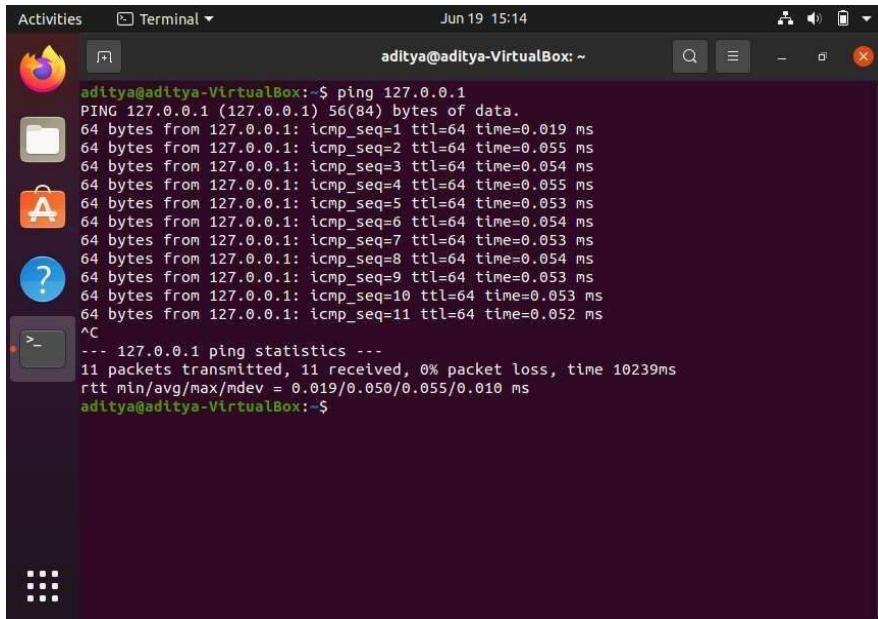
To test any host's connectivity  
`ping <ip-address>`

To start/stop/restart/reload networking services

sudo /etc/init.d/mnetworking<function>

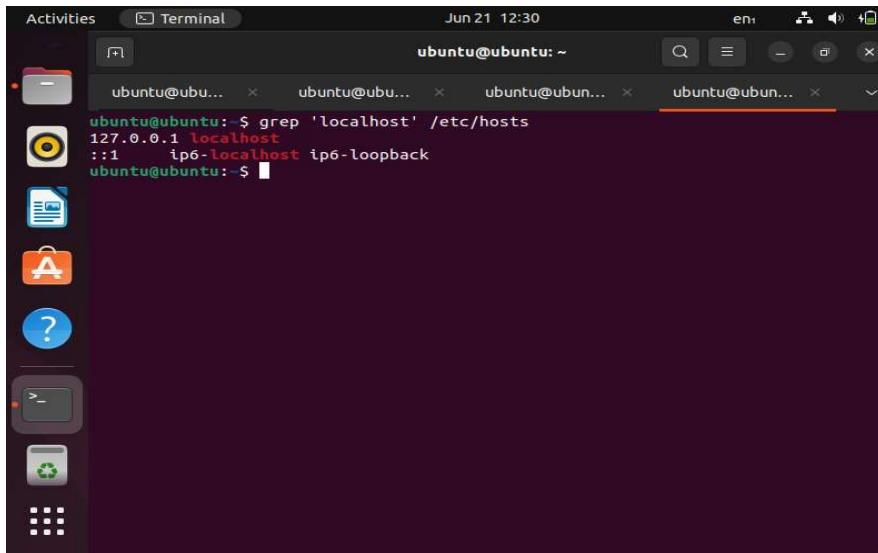
Note : <function> can be any one of stop or start or reload or restart

Q8. Check the connectivity of the host with IP address 127.0.0.1.



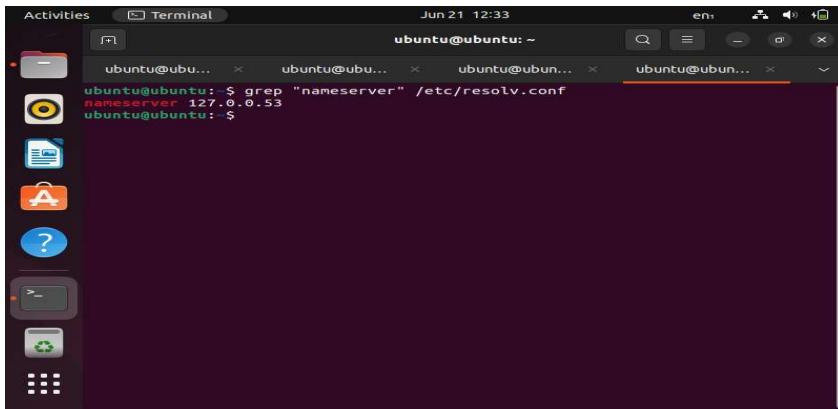
```
aditya@aditya-VirtualBox:~$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.019 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.052 ms
^C
--- 127.0.0.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10239ms
rtt min/avg/max/mdev = 0.019/0.050/0.055/0.010 ms
aditya@aditya-VirtualBox:~$
```

Q9. Find the IP address of the localhost.



```
ubuntu@ubuntu:~$ grep 'localhost' /etc/hosts
127.0.0.1 localhost
::1 ip6-localhost ip6-loopback
ubuntu@ubuntu:~$
```

Q10. Find the IP address of the DNS Server (name server)



```
Activities Terminal Jun 21 12:33 en1
ubuntu@ubuntu: ~
ubuntu@ubuntu: $ grep "nameserver" /etc/resolv.conf
nameserver 127.0.0.53
ubuntu@ubuntu: $
```

## INSTALLING INTERNET SERVICES

Installing Apache server

```
sudo apt-get install apache2
```

Configuration file for Apache server

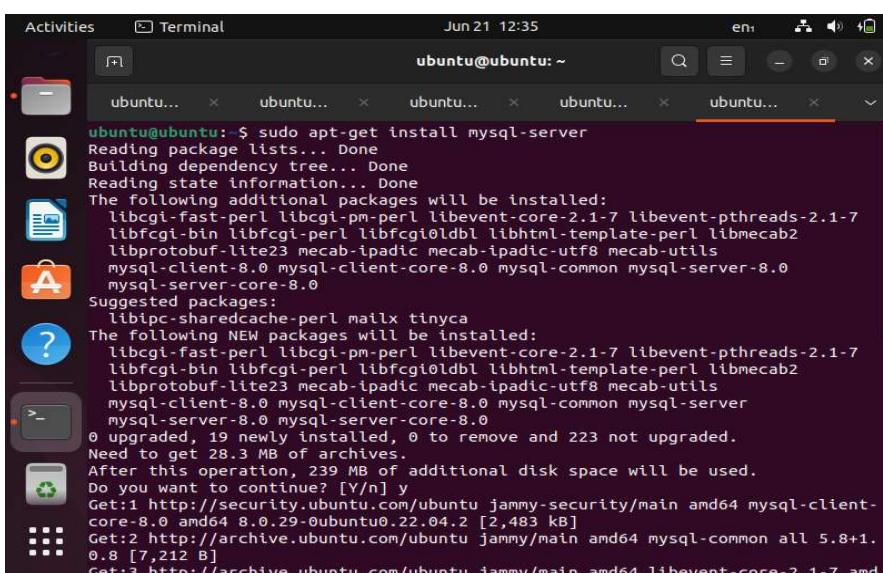
```
apache2.conf
```

Restart apache services after any configuration changes made

```
sudo /etc/init.d/mnetworking restart
```

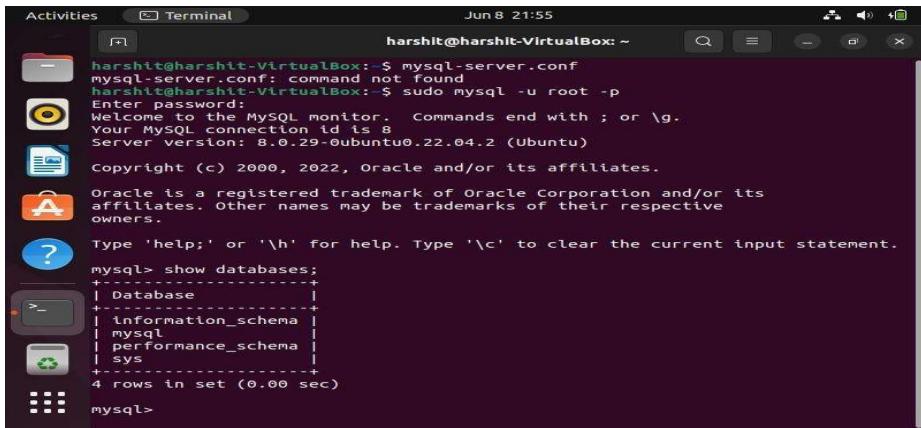
Similarly all services can be installed, configured and restarted

Q11. Install mysql server



```
Activities Terminal Jun 21 12:35 en1
ubuntu@ubuntu: ~
ubuntu@ubuntu: $ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7 libevent-pthreads-2.1-7
  libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-template-perl libmecab2
  libprotobuf-lite23 necab-ipadic necab-ipadic-utf8 necab-utils
  mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0
  mysql-server-core-8.0
Suggested packages:
  libipc-sharedcache-perl mailx tinyca
The following NEW packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7 libevent-pthreads-2.1-7
  libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-template-perl libmecab2
  libprotobuf-lite23 necab-ipadic necab-ipadic-utf8 necab-utils
  mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server
  mysql-server-8.0 mysql-server-core-8.0
0 upgraded, 19 newly installed, 0 to remove and 223 not upgraded.
Need to get 28.3 MB of archives.
After this operation, 239 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.ubuntu.com/ubuntu jammy-security/main amd64 mysql-client-core-8.0 amd64 8.0.29-0ubuntu0.22.04.2 [2,483 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 mysql-common all 5.8+1.0.8 [7,212 B]
Get:3 http://archive.ubuntu.com/ubuntu iammv/main amd64 libevent-core-2.1-7 amd
```

Q12. Log on as root into mysql server



```
Activities Terminal Jun 8 21:55
harshit@harshit-VirtualBox:~$ mysql-server.conf
mysql-server.conf: command not found
harshit@harshit-VirtualBox:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.29-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

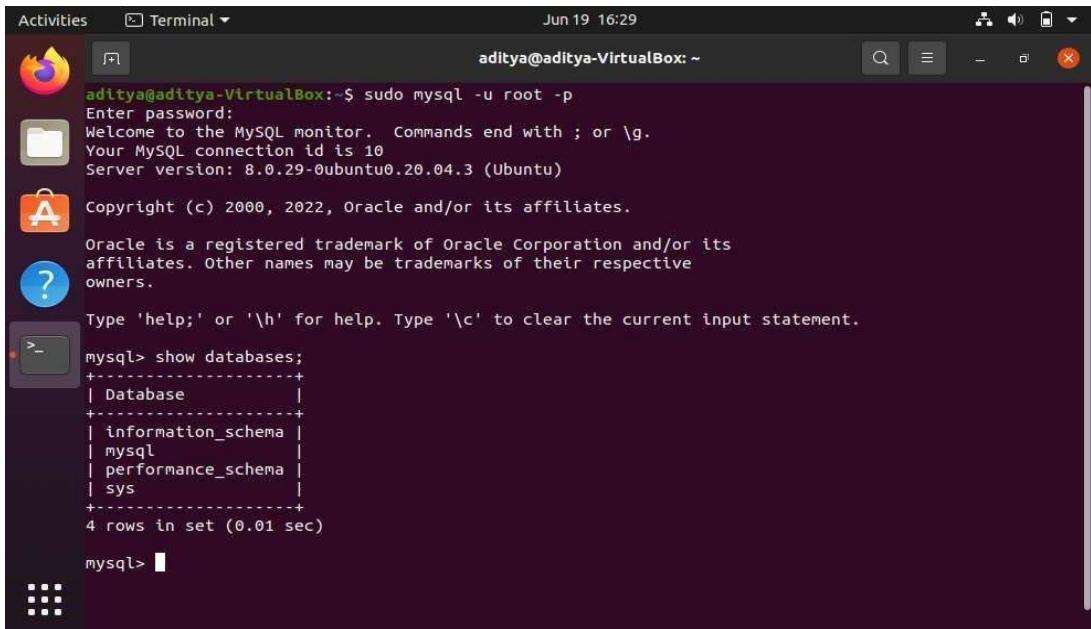
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)

mysql>
```

Q13. Create a new database for mysql server



```
Activities Terminal Jun 19 16:29
aditya@aditya-VirtualBox:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.29-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.01 sec)

mysql> ■
```

**Result-** The given program has been performed successfully.

## Program No. 8

### Simple Task Automation

Linux Cron utility is an effective way to schedule a routine background job at a specific time and/or day on an on-going basis. You can use this to schedule activities, either as one- time events or as recurring tasks.

#### Crontab Syntax m h dom mon dow command

- m – The minute when the cron job will run (0-59)
- h - a numeric value determining the hour when the tasks will run (0-23)
- dom – Day of the Month when the cron job will run (1-31)
- mon - The month when the cron job will run (1-12)
- dow – Day Of the Week from 0-6 with Sunday being 0
- command- The linux command you wish to execute

#### Scheduling of Tasks (For Ubuntu)

Step-1: Open terminal and type the command

```
crontab -e
```

Step-2: Choose the editor. Better to select nano editor

Step-3: Edit the file based on the syntax given above

Step-4: Save and Exit the file

Step-5: Start cron daemon using the following command

```
systemctl start cron
```

#### Example of crontab entry

```
0 8 * * 1 echo Have a Good Week >>tmpfile
```

Every Monday 8:00 am the message “Have a Good Week” transferred to the file ‘tmpfile’

#### Special Crontab Characters

\* represents all possible value

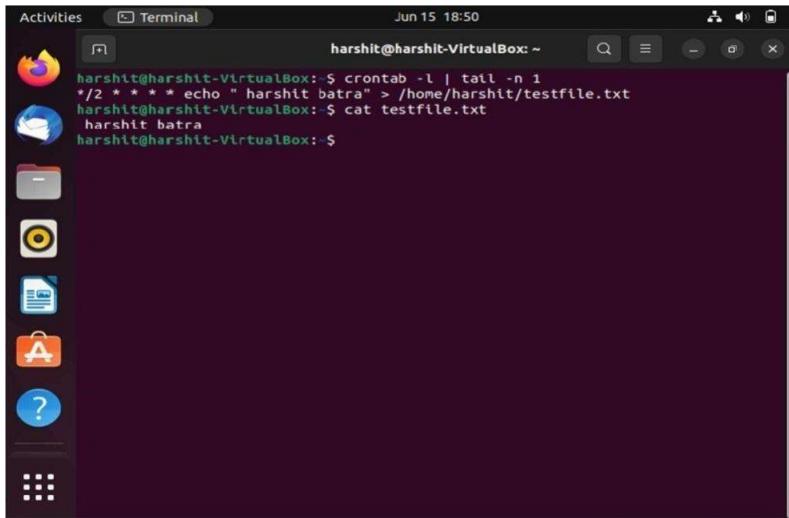
/ represents partial value. Ex. \*/10 in minute column specifies every 10 minutes

- represent range of values. Ex. 6-9 in hour column specifies 6am to 9 am

, (Comma) represent different set of values. Ex. 1,4 in month specifies Jan and Apr month

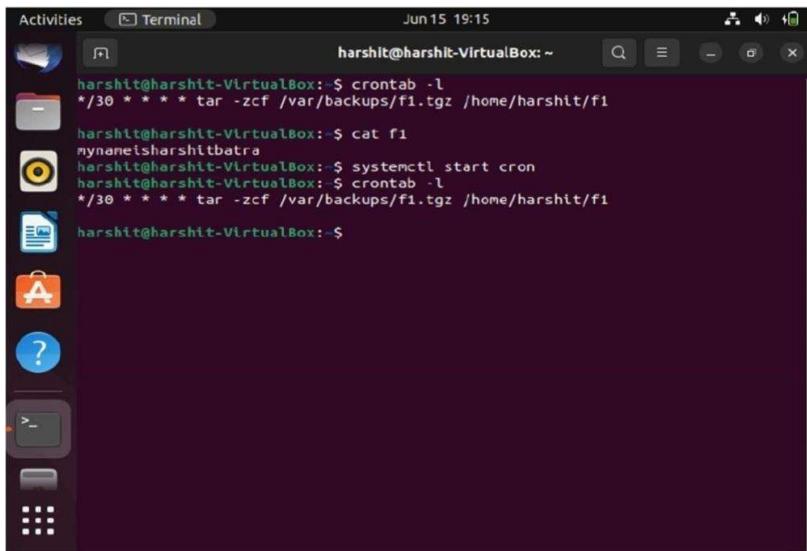
Q1. Schedule a task to display the following message on the monitor for every 2 minutes.





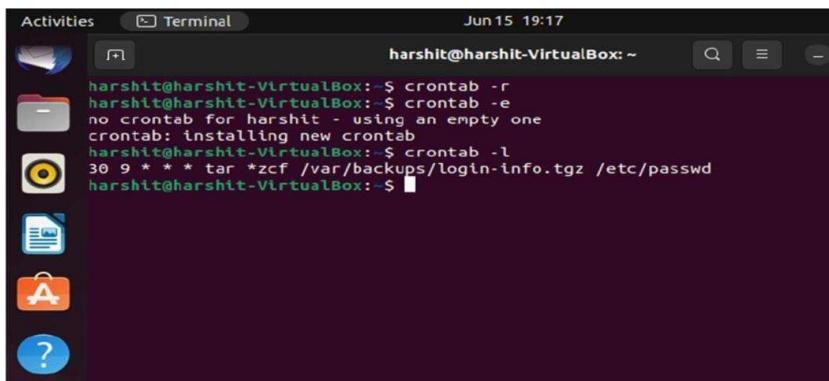
Activities Terminal Jun 15 18:50  
harshit@harshit-VirtualBox:~\$ crontab -l | tail -n 1  
\*/2 \* \* \* \* echo "harshit batra" > /home/harshit/testfile.txt  
harshit@harshit-VirtualBox:~\$ cat testfile.txt  
harshit batra  
harshit@harshit-VirtualBox:~\$

Q2. Schedule a task to take backup of your important file (say file f1) for every 30 minutes



Activities Terminal Jun 15 19:15  
harshit@harshit-VirtualBox:~\$ crontab -l  
\*/30 \* \* \* \* tar -zcf /var/backups/f1.tgz /home/harshit/f1  
harshit@harshit-VirtualBox:~\$ cat f1  
mynameisharshitbatra  
harshit@harshit-VirtualBox:~\$ systemctl start cron  
harshit@harshit-VirtualBox:~\$ crontab -l  
\*/30 \* \* \* \* tar -zcf /var/backups/f1.tgz /home/harshit/f1  
harshit@harshit-VirtualBox:~\$

Q3. Schedule a task to take backup of login information everyday 9:30am.



Activities Terminal Jun 15 19:17  
harshit@harshit-VirtualBox:~\$ crontab -r  
harshit@harshit-VirtualBox:~\$ crontab -e  
no crontab for harshit - using an empty one  
crontab: installing new crontab  
harshit@harshit-VirtualBox:~\$ crontab -l  
30 9 \* \* \* tar -zcf /var/backups/login-info.tgz /etc/passwd  
harshit@harshit-VirtualBox:~\$

**Result-** The given program has been performed successfully.

## **Program No. 9**

### **Shell Programs**

#### **How to run a Shell Script**

- Edit and save your program using editor
- Add execute permission by chmod command
- Run your program using the name of your program
- ./program-name

#### **Important Hints**

- No space before and after the assignment operator    Ex. sum=0
- Single quote ignores all special characters. Dollar sign, Back quote and Back slash are not ignored inside Double quote. Back quote is used as command substitution. Back slash is used to remove the special meaning of a character.
- Arithmetic expression can be written as follows : i=\$((i+1) or i=\$(expr \$i + 1)
- Command line arguments are referred inside the programme as \$1, \$2, ..and so on
- \$\* represents all arguments, \$# specifies the number of arguments
- read statement is used to get input from input device. Ex. read a b

#### **Syntax for if statement**

```
if [ condition ] then
...
elif [ condition ] then
    ...
else
...
fi
```

#### **Syntax for case structure**

```
case value in pat1) ...statement;;
pat2) ... Statement;;
*)
    ...
Statement;;
Esac
```

#### **Syntax for for-loop**

```
for var in list-of-values do
...
...
done
```

## Syntax for While loop

while command<sub>t</sub>do

• • •

• • •

done

## Syntax for printf statement

printf “string and format” arg1 arg2 ... ...

- Break and continue statements functions similar to C programming
  - Relational operators are -lt, -le, -gt, -ge, -eq, -ne
  - Ex.  $(i \geq 10)$  is written as [ \$i -ge 10 ]
  - Logical operators (and, or, not) are -o, -a, !
  - Ex.  $(a > b) \&\& (a > c)$  is written as [ \$a -gt \$b -a \$a -gt \$c ]
  - Two strings can be compared using = operator  $\square$

Q1. Write a program to perform sum of two numbers using shell.

**Result-** The given program has been performed successfully.

## Program No. 10

### Pipes

Pipe is a communication medium between two or more processes. The system call for creating pipe is

```
int pipe(int p[2]);
```

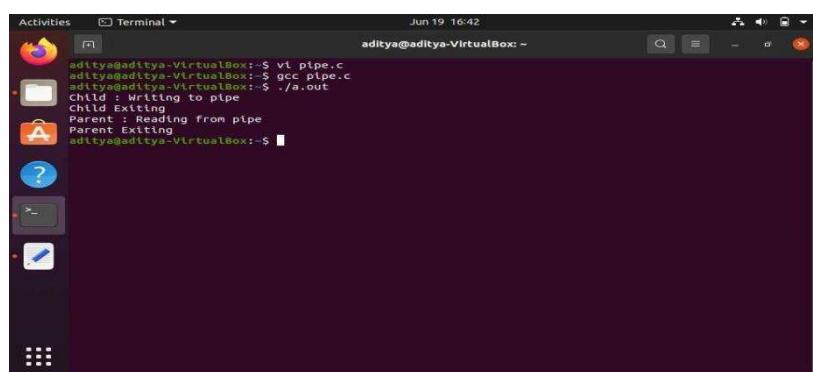
This system call would create a pipe for one-way communication i.e., it creates two descriptors, first one is connected to read from the pipe and other one is connected to write into the pipe.

Descriptor p[0] is for reading and p[1] is for writing. Whatever is written into p[1] can be read from p[0].

Q1. Write the output of the following program

```
#include <stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
    int p[2];
    char buff[25];
    if(fork()==0)
    {
        printf("Child : Writing to pipe \n");
        write(p[1],"Welcome",8); printf("Child Exiting\n");
    }
    else
    {
        wait(NULL);
        printf("Parent : Reading from pipe \n");
        read(p[1],buff,8); printf("Parent Exiting\n");
    }
    return 0;
}
```

#### Output:



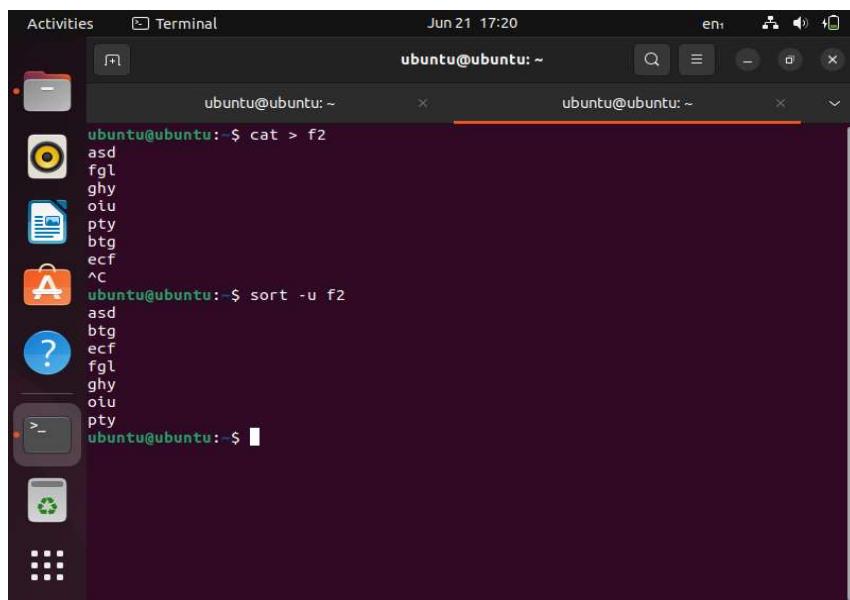
```
aditya@aditya-VirtualBox:~$ vi pipe.c
aditya@aditya-VirtualBox:~$ gcc pipe.c
aditya@aditya-VirtualBox:~$ ./a.out
Child : Writing to pipe
Child Exiting
Parent : Reading from pipe
Parent Exiting
aditya@aditya-VirtualBox:~$
```

## Implementing command line pipe using exec() family of functions00

Follow the steps to transfer the output of a process to pipe:

- i. Close the standard output descriptor
  - ii. Use the following system calls, to take duplicate of output file descriptor of the pipe  
`int dup(intfd);`  
`int dup2(intoldfd, intnewfd);`
  - iii. Close the input file descriptor of the pipe
  - iv. Now execute the process

Q.2. Write a program to sort a file using pipes.



**Result-**The given programs have been performed successfully.

## Program No. 11

### First Come First Serve Scheduling

Q. Write a program to implement First Come First Serve scheduling in C language

```
#include <stdio.h>
int main()
{ int pid[15], bt[15], n;
  printf("Enter the number of processes: ");
  scanf("%d",&n);

  printf("Enter process id of all the processes: ");
  for(int i=0;i<n;i++)
  {   scanf("%d",&pid[i]);   }

  printf("Enter burst time of all the processes: ");
  for(int i=0;i<n;i++)
  {   scanf("%d",&bt[i]);   }

  int i, wt[n]; wt[0]=0;
  for(i=1; i<n; i++)
  {   wt[i]= bt[i-1]+ wt[i-1];   }

  printf("Process ID    Burst Time    Waiting Time    TurnAround Time\n");
  float twt=0.0;
  float tat= 0.0;
  for(i=0; i<n; i++)
  {   printf("%d\t\t", pid[i]);
      printf("%d\t\t", bt[i]);
      printf("%d\t\t", wt[i]);
      printf("%d\t\t", bt[i]+wt[i]);
      printf("\n");

      twt += wt[i];
      tat += (wt[i]+bt[i]);   }

  float att,awt;
  awt = twt/n;
  att = tat/n;
  printf("Avg. waiting time= %f\n",awt);
  printf("Avg. turnaround time= %f",att);
}
```

**Output-**

```
Enter the number of processes: 3
Enter process id of all the processes: 1 2 3
Enter burst time of all the processes: 5 11 11
Process ID      Burst Time      Waiting Time      TurnAround Time
1              5                  0                  5
2              11                 5                  16
3              11                 16                 27
Avg. waiting time= 7.000000
Avg. turnaround time= 16.000000
```

**Result-**The given program have been performed successfully.

## Program No. 12

### Shortest Job First Scheduling

Q. Write a program to implement Shortest Job First scheduling in C language

```
#include<stdio.h>
int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,totalT=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }

    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }
        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;
    for(i=1;i<n;i++)
    {
        wt[i]=0;
    }
}
```

```

        for(j=0;j<i;j++)
            wt[i]+=bt[j];
        total+=wt[i];
    }

    avg_wt=(float)total/n;

    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        totalT+=tat[i];
        printf("\n p%d\t %d\t %d\t %d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)totalT/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f",avg_tat);
}

```

### Output-

```

Enter number of process:4

Enter Burst Time:
p1:5
p2:4
p3:12
p4:7

Process      Burst Time      Waiting Time      Turnaround Time
p2            4                0                4
p1            5                4                9
p4            7                9                16
p3            12               16               28

Average Waiting Time=7.250000
Average Turnaround Time=14.250000

```

**Result-**The given programs have been performed successfully.