| VP Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |
|---|---|

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: L001C | Name of Subject Teacher: Vijay Patil |
| Name of Student: Siddharth P. Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 01 |
| Title of Experiment | Install and configure Linux(or alike) operating system. |

## X. Program Code

**1. Install and configure Linux (or similar) operating system on your computer. Write down the steps for same.**

**Ans:**

**Step 1:** Download the ISO File.

**Step 2:** Boot Your system with Bootable DVD/USB drive.
To start the installation click on “Install Ubuntu”.

**Step 3:** Check Install Prerequisite.

**Step 4:** Select the Installation Type.

**Step 5**: Select Your respective Time Zone.

**Step 6:** Select Your respective Keyboard Layout.

**Step 7:** Set the Hostname of your system and User credentials that will be used after installation.
Installation has started .Once the installation is completed , it will ask to restart the machine. Click on “Restart Now”

**Step 8:** Login Screen after reboot.
Use the same user and its credentials that we have set during the installation.
We will get below screen after entering the credentials.
Ubuntu Installation is Completed Now.
Similarly any open source installation shall be considered

**2. Compare Unix and Windows OS**

**Ans:**

| **Aspect** | **Unix** | **Windows OS** |
|---|---|---|
| **Architecture** | Monolithic Kernel | Hybrid Kernel |
| **File System** | UFS, ext4, ZFS | NTFS |
| **Process Management** | Efficient multitasking; fork/exec | Complex process and inter-process management |
| **Networking** | Robust support; many tools | Extensive support; enterprise-focused tools |
| **Command-Line Interface** | Powerful CLI (Bash, Zsh) | Command Prompt, PowerShell |
| **Graphical Interface** | Various GUIs (GNOME, KDE, Xfce) | User-friendly, consistent GUI |
| **Security** | Strong security model; permissions | Advanced security features; regular updates |

## XII. Practical Related Questions

**1. What are different versions of Linux operating system?**

**Ans:**

a) Ubuntu
b) Fedora
c) Debian
d) CentOS
e) RHEL (Red Hat Enterprise Linux)
f) Arch Linux
g) openSUSE
h) Linux Mint
i) Kali Linux

**2. Enlist the steps for booting the operating system.**

**Ans:**

a) **Power On**: Press the power button to start the computer.
b) **POST (Power-On Self-Test)**: Perform hardware checks and initialize components.
c) **Load Bootloader**: Read the bootloader from the boot device (e.g., GRUB, LILO).
d) **Select OS**: Choose the operating system (if multiple are available).
e) **Load Kernel**: Load the operating system kernel into memory.
f) **Initialize Kernel**: Set up system resources, drivers, and hardware interfaces.
g) **Start Init/Systemd**: Begin the initialization process (start essential system services).
h) **Run Startup Scripts**: Execute scripts to configure user environments and services.
i) **User Login**: Present login screen or command prompt for user access.
j) **Load Desktop/GUI**: Start graphical user interface (if applicable) and user session.

**3. State names of latest multiuser operating system and its advantages.**

**Ans:**

a) **Ubuntu Server 24.04 LTS:** Long-term support and ease of use.
b) **Red Hat Enterprise Linux (RHEL) 9:** Enterprise-grade stability and advanced security.
c) **CentOS Stream 9:** Continuous updates and community-driven insights.
d) **Debian 12 (Bookworm):** Stability and flexibility.
e) **Fedora Server 39:** Cutting-edge features and modularity.

## X111 Exercise:

**1. Differentiate between command line OS and GUI OS by giving example.**

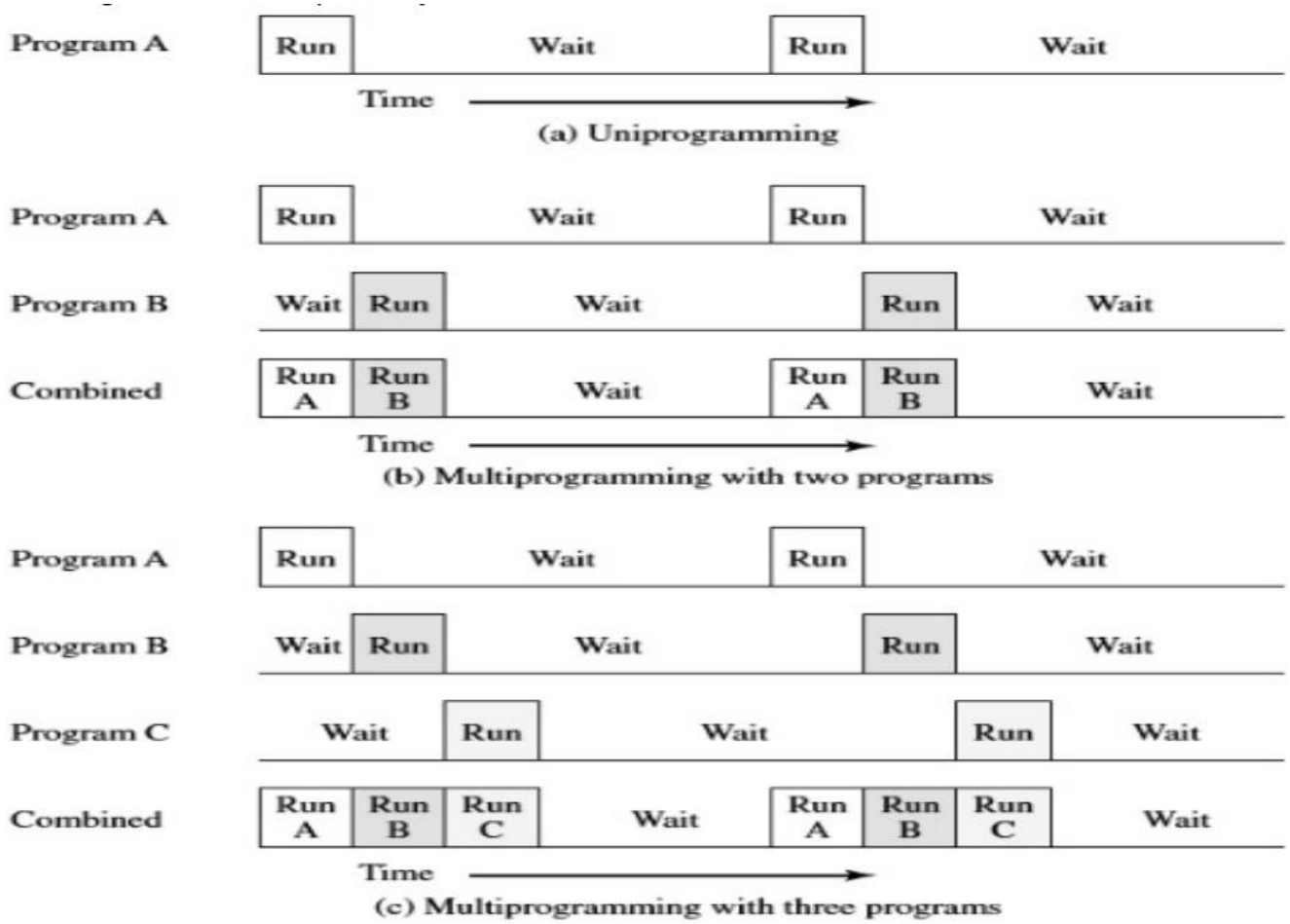
**Ans:**

| **Feature** | **Command Line OS (CLI)** | **GUI OS (Graphical User Interface)** |
|---|---|---|
| User Interaction | Command-based, text input | Graphical, uses icons, windows, and menus |
| Ease of Use | Requires knowledge of commands | More intuitive, user-friendly |
| Learning Curve | Steeper, requires memorization | Shallower, easier for beginners |
| Resource Consumption | Low, uses fewer system resources | Higher, requires more memory and CPU |
| Example | MS-DOS, Linux Terminal | Windows, macOS, Ubuntu with GNOME |
| Flexibility | High for advanced users | Limited to what the GUI provides |
| Efficiency | Fast for tasks that require repetition | Can be slower due to graphical elements |

**2. Draw the diagram of multiprogramming system and state concept of it.**

**Ans:**

i) Diagram:




ii) Multiprogramming increases CPU utilization by organizing jobs such that the CPU always has one to execute. In multiprogrammed systems the operation system keeps several jobs in memory at a time. This set of jobs is a subset of the jobs kept in the job pool.

iii) The operating system picks and begins to execute one of the jobs in the memory. Eventually the job may have to wait for some task, such as a tape is mounted, or an input/output operation to complete.

iv) In a non-multiprogramming system, the CPU would sit idle. In a multiprogramming system, the operating system simply switches to and executes another job. When that job needs to wait, the CPU is switched to another job and so on. Eventually the first job finishes waiting and gets the CPU back. As long as there is always some job to execute, the CPU will never be idle.

**3. Which are the extra facilities provided by Unix other than Windows OS?**

**Ans:**

Unix offers several facilities beyond what is typically found in Windows OS:

a) Multi-user Environment: Unix is designed to handle multiple users efficiently, allowing simultaneous logins and operations without conflicts.
b) Advanced Shell Scripting: Unix provides powerful shell scripting capabilities with shells like Bash and KornShell, enabling complex task automation and scripting.
c) Unified File System Hierarchy: Unix treats everything as a file, including devices and processes, within a single hierarchical file system.
d) Robust Security Model: Unix has a strong security model with detailed permission settings and user controls.

**4. Enlist four features of the following operating system:**

**a. Windows 98** **b. Windows 2000** **c. Windows XP**

**Ans:**

## a. Windows 98

Start Menu and Taskbar
Plug and Play Support
Internet Explorer 4.0 Integration
USB Device Support

## b. Windows 2000

Enhanced Security Features
NTFS File System Support
Active Directory Integration
Improved Hardware Support

## c. Windows XP

Improved User Interface (Luna Theme)
Fast User Switching
Remote Desktop Access
System Restore Feature

| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |
|---|---|

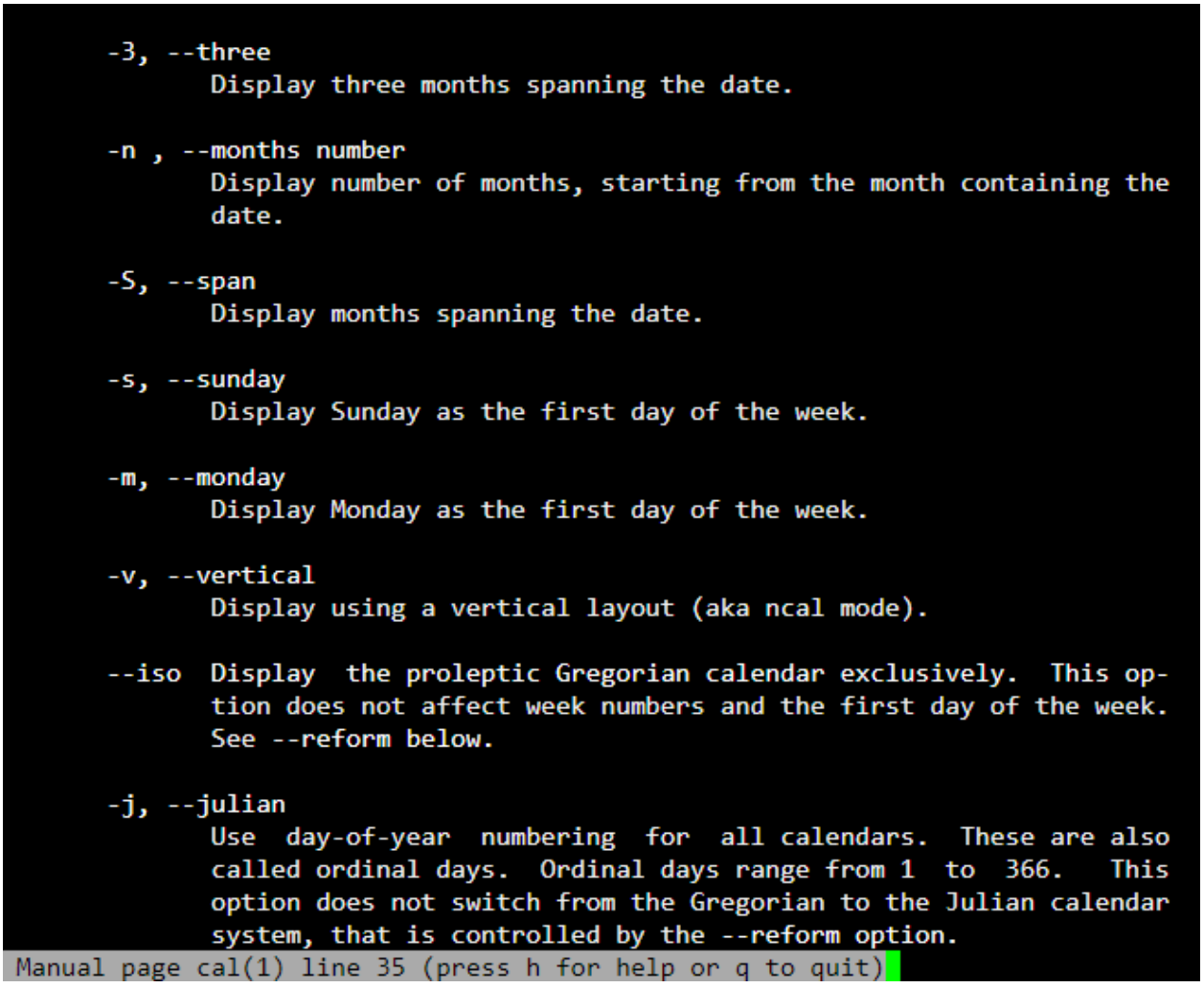
| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: L001C | Name of Subject Teacher: Vijay Patil |
| Name of Student: Sidddharth P. Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 02 |
| Title of Experiment | Execute general purpose commands. |

## X. PROGRAM CODE:

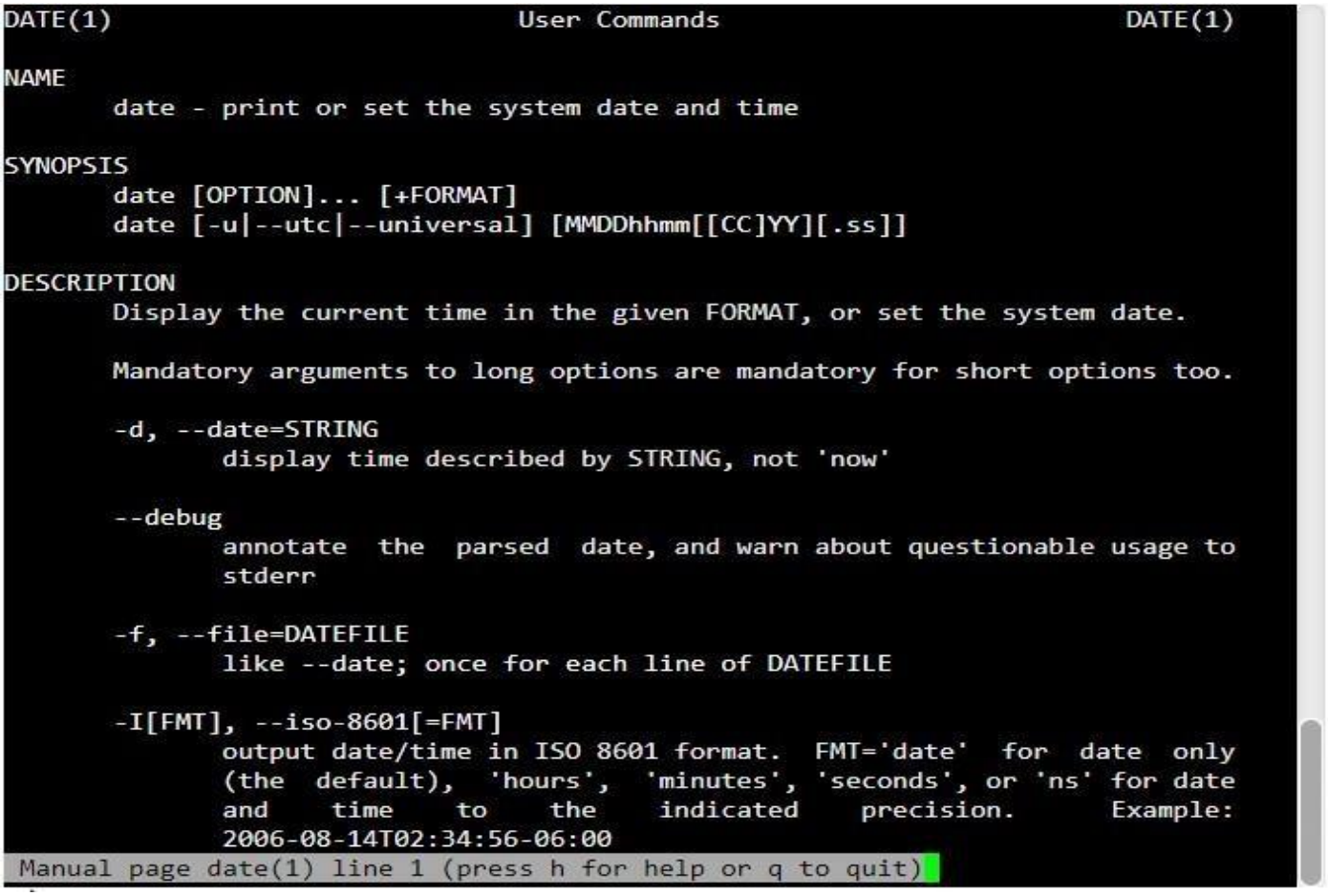
### 1. Write down different options of cal command. (Use $man cal)

**Output:**



## 2. Write options of date command. (Use $man cal)

## Output:



## XII. PRACTICAL RELATED QUESTIONS:

### 1. How you record all the following activities performed by user.

**Ans**: Use `script`, `auditd`, `tcpdump`, keyloggers (with legal/ethical considerations), application logs, and UI recording. Consider data privacy, storage, analysis, and security.

### 2. Give a command to display calendar for month of January.

**Ans**:

```
[root@localhost ~]# cal 01 2024
     January 2024
Su Mo Tu We Th Fr Sa
       1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

### 3. Give single statement command to display calendar of previous, current and next month.

**Ans:**

```
[root@localhost ~]# cal -3
      July 2024              August 2024            September 2024
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6                  1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30
```

### 4. Give the command to display full week day (eg. Sunday) using date command.

**Ans:**

```
[root@localhost ~]# date +%A
Tuesday
```

## XIII. EXERCISE:

**1. What is output of following command?**

**a. $cal 04 2019**

**OUTPUT:**

```
[root@localhost ~]# cal 04 2019
      April 2019
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

**b. $date "+Today's information: %D and %B"**

**OUTPUT:**

```
[root@localhost ~]# date "+Today's information: %D and %B"
Today's information: 08/06/24 and August
```

**c. $date "+My clock is showing %H hours, %M minutes and %S seconds"**

**OUTPUT:**

```
[root@localhost ~]# date "+My clock is showing %H hours, %M mimutes and %S seconds"
My clock is showing 11 hours, 07 mimutes and 59 seconds
```

**d. $cal-3**

**OUTPUT:**

```
[root@localhost ~]# cal -3
      July 2024              August 2024            September 2024
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6                  1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30
```

**e. $cal-5**

**OUTPUT:**

```
[root@localhost ~]# cal 5
                                0005

      January                February                 March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2   1  2  3  4  5  6  7   1  2  3  4  5  6  7
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   8  9 10 11 12 13 14
11 12 13 14 15 16 17  15 16 17 18 19 20 21  15 16 17 18 19 20 21
18 19 20 21 22 23 24  22 23 24 25 26 27 28  22 23 24 25 26 27 28
25 26 27 28 29 30 31                       29 30 31

      April                  May                    June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3  4                  1  2      1  2  3  4  5  6
 5  6  7  8  9 10 11   3  4  5  6  7  8  9   7  8  9 10 11 12 13
12 13 14 15 16 17 18  10 11 12 13 14 15 16  14 15 16 17 18 19 20
19 20 21 22 23 24 25  17 18 19 20 21 22 23  21 22 23 24 25 26 27
26 27 28 29 30        24 25 26 27 28 29 30  28 29 30
                      31
      July                   August                 September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3  4                  1         1  2  3  4  5
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   6  7  8  9 10 11 12
12 13 14 15 16 17 18   9 10 11 12 13 14 15  13 14 15 16 17 18 19
19 20 21 22 23 24 25  16 17 18 19 20 21 22  20 21 22 23 24 25 26
26 27 28 29 30 31     23 24 25 26 27 28 29  27 28 29 30
                      30 31

      October                November                 December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7         1  2  3  4  5
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
25 26 27 28 29 30 31  29 30                 27 28 29 30 31
```

**f. $cal-2000 OUTPUT:**

```
[root@localhost ~]# cal 2000
                                2000

      January               February              March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5            1  2  3  4
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   5  6  7  8  9 10 11
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  12 13 14 15 16 17 18
16 17 18 19 20 21 22  20 21 22 23 24 25 26  19 20 21 22 23 24 25
23 24 25 26 27 28 29  27 28 29              26 27 28 29 30 31
30 31

      April                 May                 June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1      1  2  3  4  5  6               1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30

      July                  August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1            1  2  3  4  5               1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 30
30 31

      October             November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7               1  2  3  4               1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |
|---|---|

| | |
|---|---|
| **Subject:** Operating Systems | **Subject Code**: 22516 |
| **Semester**: 5th Semester | **Course:** Computer Engineering |
| **Laboratory No**: LOO1C | **Name of Subject Teacher**: Vijay T. Patil |
| **Name of Student**: Siddharth P. Shah | **Roll Id**: 22203A0041 |

| | |
|---|---|
| **Experiment No:** | 03 |
| **Title of Experiment** | Work with multiple Linux terminals and basic commands. |

**X. PROGRAM CODE:**

- List down all options for who commands and write its description
  **ANSWER:**

| Commands | Syntax | Description |
|---|---|---|
| who | $who | It is used to display who are the users connected to our computer currently |
| who am i | $who am i | Display the details of the current working directory |
| login | login: $username | Prompt, enter username |
| passwd | $passwd uname | Sets password for users |
| su(sudo) | $su Is | Provides super user privileges |
| pwd(Present Working Dir) | $pwd | To print the complete path of the current working directory |

**XI. RESULT (OUTPUT OF COMMAND):**

1. who
2. who am i
3. login
4. passwd
5. su (sudo)

```
[root@localhost ~]# $su ls
bench.py  hello.c
```

6. pwd

**XII. PRACTICAL RELATED QUESTIONS:**

1. Give command for present working directory

   **ANSWER:** $pwd is the command to print the present working directory in Linux.

2. State currently logged in users by command

   **ANSWER:** $who command will list the currently logged-in users on a Linux system. It provides information such as the username, terminal, login time, and often the remote host (if applicable).

3. Acquire the status of super user

   **ANSWER:** By using the `su` command followed by the root password.

```
[root@localhost ~]# $su ls
bench.py  hello.c
```

**XIII. EXCERSISE:**

1. Acquire the status of super user
   **ANSWER:** By using the `su` command followed by the root password.

```
[root@localhost ~]# $su ls
bench.py  hello.c
```

2. Write output of the following commands:

   i. $who;clear;who am i

```
[root@localhost ~]#
```

ii. $who;tty;date

```
[root@localhost ~]# $who;tty;date
/dev/hvc0
Tue Aug  6 11:05:52 AM UTC 2024
[root@localhost ~]#
```

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| **Subject:** Operating Systems | **Subject Code**:22413 |
| **Semester**: 5th Semester | **Course:** Computer Engineering |
| **Laboratory No**: L001C | **Name of Subject Teacher**: Vijay Patil |
| **Name of Student**: Siddharth P. Shah | **Roll Id**: 22203A0041 |

| | |
|---|---|
| **Experiment No:** | 04 |
| **Title of Experiment** | Work with multiple Linux terminals and basic commands. |

## X. PROGRAM CODE:

- Check all the permissions started on your system. Stop the services which are not required for long time?

**ANSWER:**

**Start:**

Sudo service ssh start

**Stop:**

Sudo service ssh stop

**Restart:**

Sudo service ssh restart

**List all services:**

Service –status-all

```
┌──(mc㉿kali)-[~]
└─$ service --status-all
 [ - ]  apache-htcacheclean
 [ - ]  apache2
 [ - ]  apparmor
 [ - ]  atftpd
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ - ]  cryptdisks
 [ - ]  cryptdisks-early
 [ + ]  dbus
 [ - ]  dns2tcp
 [ - ]  exim4
 [ + ]  gdm3
 [ + ]  haveged
 [ - ]  inetsim
 [ - ]  iodined
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  mariadb
 [ - ]  miredo
 [ - ]  mosquitto
 [ + ]  networking
 [ - ]  nfs-common
 [ - ]  nginx
 [ - ]  nmbd
 [ - ]  openvpn
 [ + ]  pcscd
 [ - ]  plymouth
 [ + ]  plymouth-log
 [ - ]  postgresql
 [ + ]  procps
 [ - ]  ptunnel
 [ - ]  redis-server
 [ - ]  redsocks
 [ - ]  rpcbind
 [ - ]  rsync
 [ - ]  rwhod
 [ - ]  samba-ad-dc
 [ - ]  saned
 [ - ]  screen-cleanup
 [ - ]  smartmontools
 [ - ]  smbd
 [ - ]  snmpd
 [ - ]  speech-dispatcher
 [ - ]  ssh
 [ - ]  sslh
 [ - ]  stunnel4
 [ - ]  sudo
 [ - ]  sysstat
 [ + ]  virtualbox-guest-utils
 [ - ]  x11-common
```

```
└─$ sudo service sysstat start

┌──(mc㉿kali)-[~]
└─$ service --status-all
 [ - ]  apache-htcacheclean
 [ - ]  apache2
 [ - ]  apparmor
 [ - ]  atftpd
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ - ]  cryptdisks
 [ - ]  cryptdisks-early
 [ + ]  dbus
 [ - ]  dns2tcp
 [ - ]  exim4
 [ + ]  gdm3
 [ + ]  haveged
 [ - ]  inetsim
 [ - ]  iodined
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  mariadb
 [ - ]  miredo
 [ - ]  mosquitto
 [ + ]  networking
 [ - ]  nfs-common
 [ - ]  nginx
 [ - ]  nmbd
 [ - ]  openvpn
 [ - ]  pcscd
 [ - ]  plymouth
 [ + ]  plymouth-log
 [ - ]  postgresql
 [ + ]  procps
 [ - ]  ptunnel
 [ - ]  redis-server
 [ - ]  redsocks
 [ - ]  rpcbind
 [ - ]  rsync
 [ - ]  rwhod
 [ - ]  samba-ad-dc
 [ - ]  saned
 [ - ]  screen-cleanup
 [ - ]  smartmontools
 [ - ]  smbd
 [ - ]  snmpd
 [ - ]  speech-dispatcher
 [ - ]  ssh
 [ - ]  sslh
 [ - ]  stunnel4
 [ - ]  sudo
 [ + ]  sysstat
 [ + ]  virtualbox-guest-utils
 [ - ]  x11-common
```

```
┌──(mc㉿kali)-[~]
└─$ sudo service sysstat stop

┌──(mc㉿kali)-[~]
└─$ service --status-all
 [ - ]  apache-htcacheclean
 [ - ]  apache2
 [ - ]  apparmor
 [ - ]  atftpd
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ - ]  cryptdisks
 [ - ]  cryptdisks-early
 [ + ]  dbus
 [ - ]  dns2tcp
 [ - ]  exim4
 [ + ]  gdm3
 [ + ]  haveged
 [ - ]  inetsim
 [ - ]  iodined
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  mariadb
 [ - ]  miredo
 [ - ]  mosquitto
 [ + ]  networking
 [ - ]  nfs-common
 [ - ]  nginx
 [ - ]  nmbd
 [ - ]  openvpn
 [ - ]  pcscd
 [ - ]  plymouth
 [ + ]  plymouth-log
 [ - ]  postgresql
 [ + ]  procps
 [ - ]  ptunnel
 [ - ]  redis-server
 [ - ]  redsocks
 [ - ]  rpcbind
 [ - ]  rsync
 [ - ]  rwhod
 [ - ]  samba-ad-dc
 [ - ]  saned
 [ - ]  screen-cleanup
 [ - ]  smartmontools
 [ - ]  smbd
 [ - ]  snmpd
 [ - ]  speech-dispatcher
 [ - ]  ssh
 [ - ]  sslh
 [ - ]  stunnel4
 [ - ]  sudo
 [ - ]  sysstat
 [ + ]  virtualbox-guest-utils
 [ - ]  x11-common
```

```
┌──(mc㉿kali)-[~]
└─$ sudo service sysstat restart

┌──(mc㉿kali)-[~]
└─$ service --status-all
 [ - ]  apache-htcacheclean
 [ - ]  apache2
 [ - ]  apparmor
 [ - ]  atftpd
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ - ]  cryptdisks
 [ - ]  cryptdisks-early
 [ + ]  dbus
 [ - ]  dns2tcp
 [ - ]  exim4
 [ + ]  gdm3
 [ + ]  haveged
 [ - ]  inetsim
 [ - ]  iodined
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  mariadb
 [ - ]  miredo
 [ - ]  mosquitto
 [ + ]  networking
 [ - ]  nfs-common
 [ - ]  nginx
 [ - ]  nmbd
 [ - ]  openvpn
 [ - ]  pcscd
 [ - ]  plymouth
 [ + ]  plymouth-log
 [ - ]  postgresql
 [ + ]  procps
 [ - ]  ptunnel
 [ - ]  redis-server
 [ - ]  redsocks
 [ - ]  rpcbind
 [ - ]  rsync
 [ - ]  rwhod
 [ - ]  samba-ad-dc
 [ - ]  saned
 [ - ]  screen-cleanup
 [ - ]  smartmontools
 [ - ]  smbd
 [ - ]  snmpd
 [ - ]  speech-dispatcher
 [ - ]  ssh
 [ - ]  sslh
 [ - ]  stunnel4
 [ - ]  sudo
 [ + ]  sysstat
 [ + ]  virtualbox-guest-utils
 [ - ]  x11-common
```

## XI. Practical Related Questions

**1. List various menus you observed on your system?**

**ANS:**

File manager
Firefox web
Libre office writer
Libre office calc
Libre office impress
Ubantu software
Amazon

System settings
Backup
Floppy disk
Trash

**2. STUDY THE GUI OF YOUR LINUX SYSTEM?**

**ANS:**

Ubuntu's GUI is clean and user-friendly, primarily using the GNOME desktop environment. At the top of the screen, you'll find a bar with system menus and status icons, while the Activities Overview lets you manage windows and launch apps. The left side features an app launcher and dock for quick access. Nautilus serves as the file manager for browsing and managing files. Overall, the interface is designed to be intuitive and straightforward, making navigation easy for both new and experienced users.

**3. Differentiate between CLI and GUI?**

**ANS:**

| S.NO | CLI | GUI |
|---|---|---|
| **1.** | CLI is difficult to use. | Whereas it is easy to use. |
| **2.** | It consumes low memory. | While consuming more memory. |
| **3.** | In CLI we can obtain high precision. | While in it, low precision is obtained. |
| **4.** | CLI is faster than GUI. | The speed of GUI is slower than CLI. |
| **5.** | CLI operating system needs only a keyboard. | While GUI operating system needs both a mouse and keyboard. |
| **6.** | CLI's appearance can not be modified or changed. | While its appearance can be modified or changed. |
| **7.** | In CLI, input is entered only at a command prompt. | While in GUI, the input can be entered anywhere on the screen. |

| S.NO | CLI | GUI |
|---|---|---|
| 8. | In CLI, the information is shown or presented to the user in plain text and files. | While in GUI, the information is shown or presented to the user in any form such as: plain text, videos, images, etc. |
| 9. | In CLI, there are no menus provided. | While in GUI, menus are provided. |
| 10. | There are no graphics in CLI. | While in GUI, graphics are used. |
| 11. | CLI do not use any pointing devices. | While it uses pointing devices for selecting and choosing items. |
| 12. | In CLI, spelling mistakes and typing errors are not avoided. | Whereas in GUI, spelling mistakes and typing errors are avoided. |
| 13. | Some command-line environments provide multitasking but it is complicated to see several things on one screen. | GUI enables a user to easily observe and operate various things at once. |
| 14. | CLI enables a user to simply script a series of instructions to carry out a task or execute a program. | GUI does not provide the facility to script a sequence of commands. |

## XIII. EXERCISE

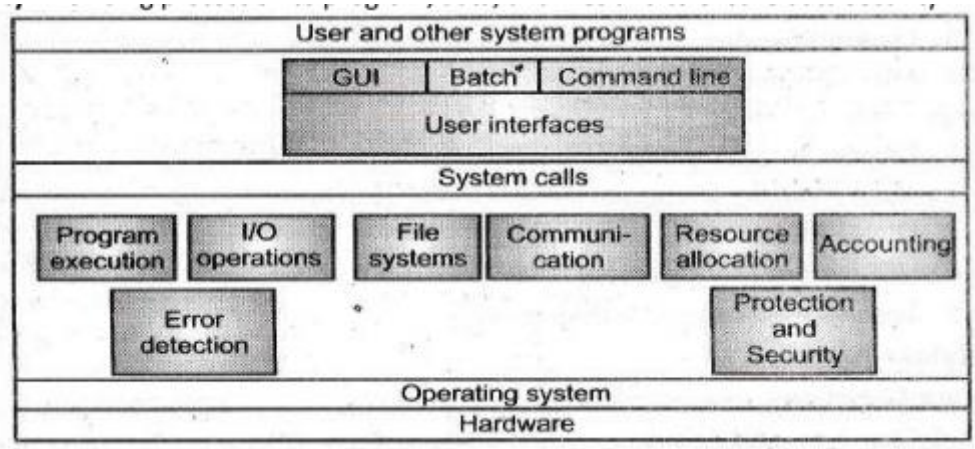
**1.What are the system calls provided by file management?**
**ANS:**

- Create file, Delete file
- Open a file, Close a file
- Create directory
- Read, Write, Reposition
- Get file attributes, Set file attributes
- Create a link
- Change working directory

**2.Draw and explain services provided by operating system?**

**ANS:**

· Program Execution: system capability to load a program into memory and to run it.
· I/O operations: since user programs cannot execute I/O operations directly, the operating system
must provide some means to perform I/O.
· File-system manipulation: program capability to read, write, create, and delete files. Maintain details
of files or directories with their respective details.
· Communications: exchange of information between processes executing either on the same
computer or on different systems tied together by a network. Implemented via shared memory or
message passing.
· Error Detection: ensure correct computing by detecting errors in the CPU and memory hardware, in
I/O devices, or in user programs.
· Resource Allocation: allocating resources to multiple users or multiple jobs running at the same time.
Coordinating among peripherals.
· Accounting: keep track of and record which users use how much and what kinds of computer
resources for account billing or for accumulating usage statistics.
· Protection: ensuring that all access to system resources is controlled.
· Security: Providing protection to program, data, and files and to ensure data security.




**3. What are the system component of operating system?**

**ANS:**

- Process management
- Files management
- Command Interpreter
- System calls
- Signals
- Network management
- Security management
- I/O device management
- Secondary storage management
- Main memory management

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: L001 | Name of Subject Teacher: Prof. Vijay Patil |
| Name of Student: Siddharth P. Shah | Roll Id: 22203A0041 |

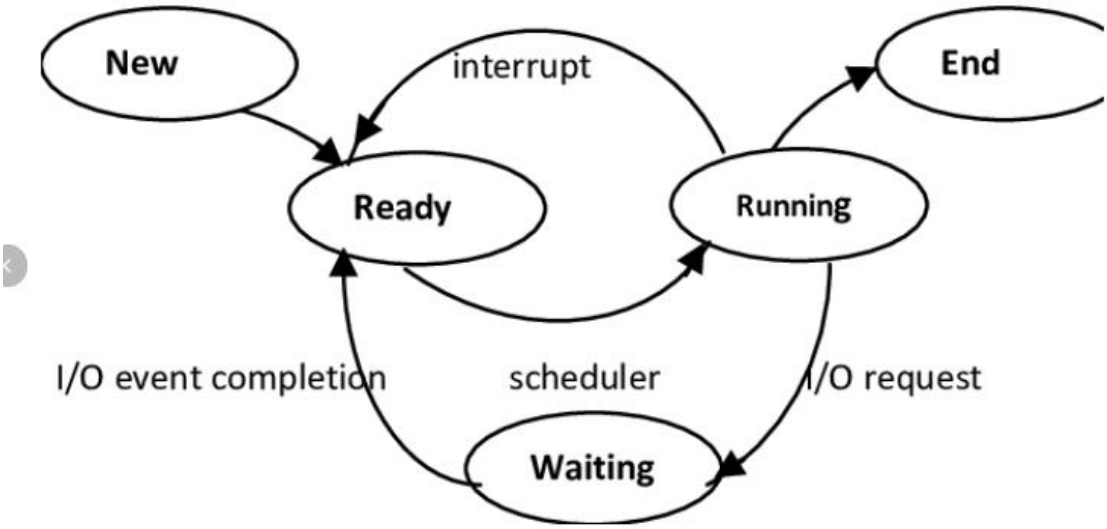
| | |
|---|---|
| Experiment No: | 5 |
| Title of Experiment | Execute process commands |

**Practical Set**

**1. Execute process commands: ps, wait, sleep, exit, kill.**

```
┌──(mc㉿kali)-[~]
└─$ ps
    PID TTY          TIME CMD
   2978 pts/0    00:00:00 zsh
   3341 pts/0    00:00:00 ps

┌──(mc㉿kali)-[~]
└─$ wait

┌──(mc㉿kali)-[~]
└─$ sleep 10; echo Mohit
Mohit

┌──(mc㉿kali)-[~]
└─$ kill
kill: not enough arguments
```

**2. Explain Process state diagram.**




• New State: A process that has just been created but has not yet been admitted to the pool of execution processes by the operating system. Every new operation which is requested to the system is known as the new born process.

• Ready State: When the process is ready to execute but it is waiting for the CPU to execute then this is called as the ready state. After the completion of the input and outputs the process will be on ready state means the process will wait for the processor to execute.

• Running State: The process that is currently being executed. When the process is running under the CPU, or when the program is executed by the CPU, then this is called as the running state process and when a process is running then this will also provide us some outputs on the screen.

• Waiting or Blocked: A process that cannot execute until some event occurs or an I/O completion. When a process is waiting for some input and output operations then this is called as the waiting state. And in this state process is not under the execution instead the process is stored out of memory and when the user will provide the input then this will again be on ready state.

• Terminated State: After the completion of the process, the process will be automatically terminated by the CPU, so this is also called as the terminated state of the process. After executing the whole process, the processor will also de-allocate the memory which is allocated to the process. So this is called as the terminated process.

**3. Difference between Process and Thread.**

| Process | Thread |
|---|---|
| Process means any program is in execution. | Thread means a segment of a process. |
| The process takes more time to terminate. | The thread takes less time to terminate. |
| It takes more time for creation. | It takes less time for creation. |
| It also takes more time for context switching. | It takes less time for context switching. |
| The process is less efficient in terms of communication. | Thread is more efficient in terms of communication. |
| Multiprogramming holds the concepts of multi-process. | We don't need multi programs in action for multiple threads because a single process consists of multiple threads. |
| The process is isolated. | Threads share memory. |
| The process is called the heavyweight process. | A Thread is lightweight as each thread in a process shares code, data, and resources. |

**4. List System Calls for Process Management**

- **wait()** - Waits for a child process to terminate and retrieves its exit status.
- **exit()** - Terminates the calling process.
- **getpid()** - Retrieves the process ID of the calling process.
- **kill()** - Sends a signal to a process, often used to terminate it.
- **getppid()** - Retrieves the parent process ID of the calling process.
- **nice()** - Sets the scheduling priority of a process.

**5. What is process ID of your login shell?**

You can find the process ID (PID) of your login shell by running `echo $$` in your terminal.

**6. Give PID of all processes, how will you terminate the processes running on your terminal?**

To terminate processes running in your terminal, you can use `kill` followed by the PID of each process.
To get a list of PIDs, you can use `ps` or `pgrep`.
For example, `kill $(pgrep -o -u $USER)` will terminate the oldest process you own.

**7. What is difference between wait and sleep?**

| Wait | Sleep |
|---|---|
| Waits for processes to complete before continuing. | Delays execution for a defined time period. |
| Can wait for specific process IDs or all background jobs. | Always waits for the exact time specified. |
| Useful for synchronizing tasks in scripts. | Used to introduce time delays, regardless of process states. |

**Practical Related Questions:**

1. **What is name of your login shell?**
   Echo $shell
2. **What are various options of kill command?**
   - **kill 0**: Kills all the processes on the terminal except the login shell by using the special argument 0.
   - **kill 120 230 234**: Kills three processes with PIDs 120, 230, and 234.
   - **kill -9 0**: Kills all processes including the login shell.
   - **kill -9 $$**: Kills the login shell.
3. **What are various options of ps command?**
   - **ps -f**: Full listing showing PPID of each process.
   - **ps -u username**: Displays processes of user username.
   - **ps -a**: Processes of all users.
   - **ps -e**: Processes including user and system processes.
4. **Explain about exit command.**

   Used to quit the shell.

5. **List the system calls for process management.**
   - **wait()** - Waits for a child process to terminate and retrieves its exit status.
   - **exit()** - Terminates the calling process.
   - **getpid()** - Retrieves the process ID of the calling process.
   - **kill()** - Sends a signal to a process, often used to terminate it.
   - **getppid()** - Retrieves the parent process ID of the calling process.
   - **nice()** - Sets the scheduling priority of a process.

## XIII. Exercise

1. **Observe the output of the following commands:**
   **$sleep 30; date**
   **$$echo $$**

```
┌──(mc㉿kali)-[~]
└─$ sleep 30; date
Tue Aug 20 11:41:10 IST 2024
```

```
┌──(mc㉿kali)-[~]
└─$ echo $$
4680
```

2. **Display full listing of all the processes running on your terminal.**

```
┌──(mc㉿kali)-[~]
└─$ sleep 30; date ; echo $$
Tue Aug 20 21:24:03 IST 2024
2978

┌──(mc㉿kali)-[~]
└─$ sleep 60; figlet GOOD
  ____ ___   ___  ____
 / ___|/ _ \ / _ \|  _ \
| |  _| | | | | | | | | |
| |_| | |_| | |_| | |_| |
 \____|\___/ \___/|____/


┌──(mc㉿kali)-[~]
└─$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.6  0.1  22600 13852 ?        Ss   21:21   0:01 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    21:21   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    21:21   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/R-netns]
root           8  0.0  0.0      0     0 ?        I    21:21   0:00 [kworker/0:0-events]
root           9  0.0  0.0      0     0 ?        I    21:21   0:00 [kworker/0:1-cgroup_destroy]
root          10  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/0:0H-events_highpri]
root          11  0.0  0.0      0     0 ?        I    21:21   0:00 [kworker/u6:0-ext4-rsv-conversi
root          12  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/R-mm_pe]
root          13  0.0  0.0      0     0 ?        I    21:21   0:00 [rcu_tasks_kthread]
root          14  0.0  0.0      0     0 ?        I    21:21   0:00 [rcu_tasks_rude_kthread]
root          15  0.0  0.0      0     0 ?        I    21:21   0:00 [rcu_tasks_trace_kthread]
root          16  0.0  0.0      0     0 ?        S    21:21   0:00 [ksoftirqd/0]
root          17  0.1  0.0      0     0 ?        I    21:21   0:00 [rcu_preempt]
root          18  0.0  0.0      0     0 ?        S    21:21   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    21:21   0:00 [idle_inject/0]
root          20  0.0  0.0      0     0 ?        S    21:21   0:00 [cpuhp/0]
root          21  0.0  0.0      0     0 ?        S    21:21   0:00 [cpuhp/1]
root          22  0.0  0.0      0     0 ?        S    21:21   0:00 [idle_inject/1]
root          23  0.2  0.0      0     0 ?        S    21:21   0:00 [migration/1]
root          24  0.0  0.0      0     0 ?        S    21:21   0:00 [ksoftirqd/1]
root          25  0.0  0.0      0     0 ?        I    21:21   0:00 [kworker/1:0-ata_sff]
root          26  0.0  0.0      0     0 ?        I<   21:21   0:00 [kworker/1:0H-events_highpri]
root          27  0.0  0.0      0     0 ?        S    21:21   0:00 [cpuhp/2]
root          28  0.0  0.0      0     0 ?        S    21:21   0:00 [idle_inject/2]
root          29  0.2  0.0      0     0 ?        S    21:21   0:00 [migration/2]
root          30  0.0  0.0      0     0 ?        S    21:21   0:00 [ksoftirqd/2]
```

3. **Write output of following commands.**
   **$sleep 60; banner GOOD**

   (mc@kali)-[~] $ sleep 60; figlet GOOD

   GOOD

4. **Write all the process commands.**
   - **ps**: Displays current processes.
   - **top**: Shows a dynamic view of system processes.
   - **htop**: An enhanced, interactive process viewer.
   - **kill**: Sends signals to processes, typically to terminate them.
   - **pkill**: Sends signals to processes based on name.
   - **killall**: Terminates processes by name.
   - **nice**: Starts a process with a modified priority.
   - **renice**: Changes the priority of an already running process.

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| **Subject:** OSY | **Subject Code**:22516 |
| **Semester:** 5th Semester | **Course:** Computer Engineering |
| **Laboratory No:** V118 | **Name of Subject teacher:** Prof. Natasha Brahmne |
| **Name of Student:** Siddharth Shah | **Roll Id:** 22203A0041 |

| | |
|---|---|
| **Experiment No:** | 6 |
| **Title of Experiment:** | Execute file and directory commands. |

## • Practical Related Questions

1. **What are the different options of ls command? Write down the command along with options and note down the output. (Use $man command to check options)**

→

2. **What are two different options of mv command?**

→

1. -i (interactive)
2. -f (Force)
3. -n (no-clobber)
4. -b(backup)
5. –version

3. **what is use of split commands**

→ The split command in Linux is a highly useful tool for dividing large files into smaller, more manageable pieces. This command is often used in scenarios where you need to break down a large file for easier data processing or distribution. In this article, we'll dissect the split command, its syntax, options, and some practical examples of its usage.

Syntax of split Command in Linux:
The basic syntax of the split command in Linux is as follows:
**Split [option][file prefix]**
*Where:*
• OPTIONS: These are optional parameters that modify the behavior of the split command.
• FILE: This is the input file that you want to split.
• PREFIX: This parameter specifies the prefix for the output files. The default prefix is 'x'.

4. **How to use join command?**

→ The Linux join command is a powerful tool that is used to merge two different files based on a common field. command reads contents of two files and merges them based on specified field, which can be a string or a numeric value. In this article, we will discuss various aspects of join command and its usage.
The syntax for join command is as follows −
**join [options] file1 file2**
t − This option is used to specify delimiter character used in files. By default, delimiter is a
blank space.
• -1 − This option is used to specify field number in first file.
• -2 − This option is used to specify field number in second file.
• -a − This option is used to print all lines from both files, including those that do not match.
• -e − This option is used to replace missing fields with a specified value.

**Example 1**

**File 1 −**
**1 Alpha 2 Bravo 3 Charlie 4 Delta 5 Echo**

**File 2 −**
**2 20 3 30 4 40 5 50 6 60**

We can join these two files based on first field in each file using following command − join file1 file2

The output will be as follows −
**2 Bravo 20 3 Charlie 30 4 Delta 40 5 Echo 50**

• **EXERCISE**

**1.Write output of following commands**

a) **Display all file names whose name starts with ‘a’ and ends with ‘y’.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ man ls

┌──(mc㉿kali)-[~/OSY]
└─$ ls a*t
a1.txt  a1cp.txt  a2.txt  a3.txt
```

b) **Enlist all files beginning with ‘m’ and ending with any range 1 to 5.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ ls a*[1-5]
a1  a2  a3
```

c) **Show the contents of the files whose file names contains exactly two characters.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat ??
Hello
Mohit
Atharva
Shravan
Neel
Hitesh
Shelar
Parth
Abhishek
Vedant
Rane
Pranit
Hello Kali Linux
```

**d) Create a file ABCD.txt, create a copy with XXXX.txt .Rename the original file with AACD.txt. Delete the file XXXX.txt.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat > ABCD.txt
Mohit Chaudhari
^C

┌──(mc㉿kali)-[~/OSY]
└─$ cp ABCD.txt XXXX.txt

┌──(mc㉿kali)-[~/OSY]
└─$ mv ABCD.txt AACD.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
AACD.txt  a1cp.txt  a3      chapter1     combined1.txt  names    xz
XXXX.txt  a2        a3.txt  chapter2     dte            surname
a1        a2.txt    abc     combine.txt  name           xaa

┌──(mc㉿kali)-[~/OSY]
└─$ rm AACD.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
XXXX.txt  a1cp.txt  a2.txt  a3.txt  chapter1  combine.txt    dte   names    xaa
a1        a2        a3      abc     chapter2  combined1.txt  name  surname  xz
```

**e) Display the inodes of any 2 files at the same time.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat a1

┌──(mc㉿kali)-[~/OSY]
└─$ cat > a1
Mohit
Atharva
^C

┌──(mc㉿kali)-[~/OSY]
└─$ ls -i a1 a2
1966216 a1  1969438 a2
```

**2. List all file processing commands.**

→

- cat
- less
- more
- head
- tail
- file
- diff
- wc
- nano

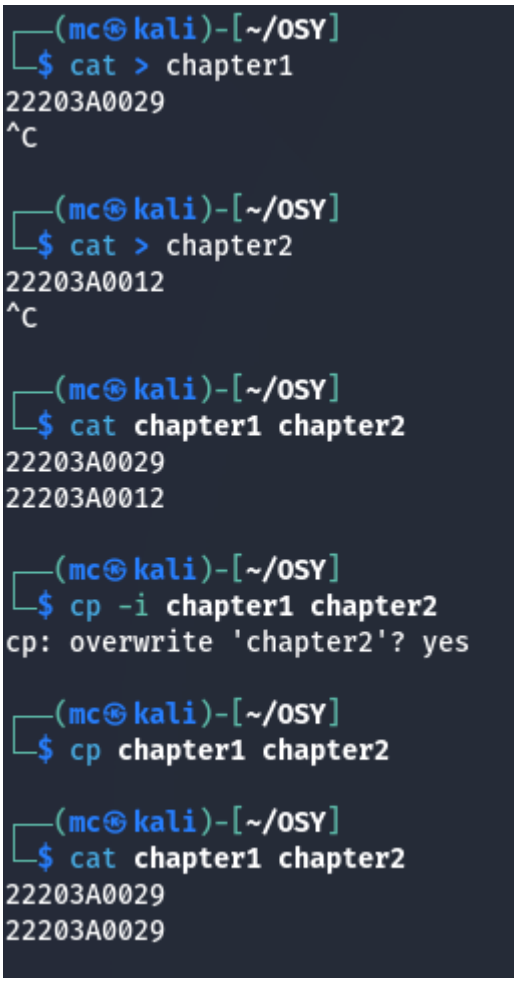
- vim
- sed
- awk

**3. How many lines will be displayed with head command if number is not specified?**

→ If the number of lines is not specified with the head command, it will display the first 10 lines of the file by default.

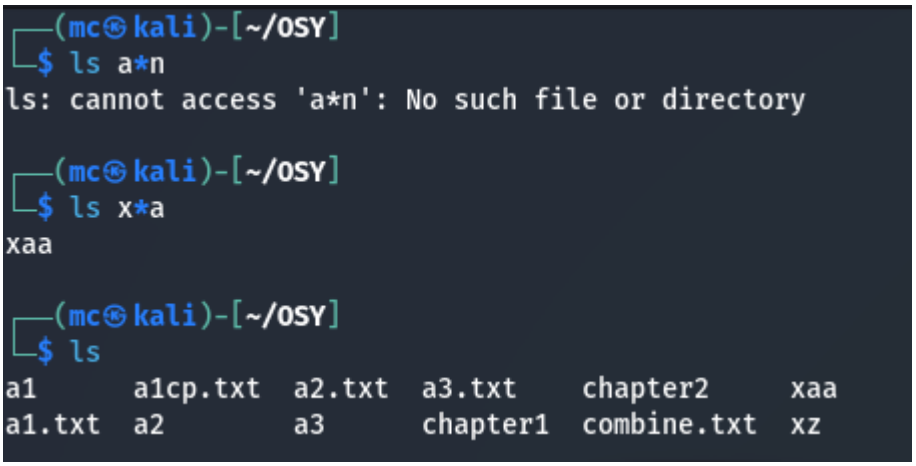
**4. Create 2 files chapter1 and chapter2 and perform the following operations**

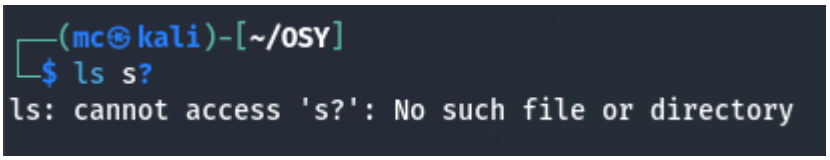
1) Copy content of chapter1 to chapter2 by asking the user before overwrite.
2) Display the inodes of 2 files
3)Rename the file ‘chapter1’ to ‘lesson1’.

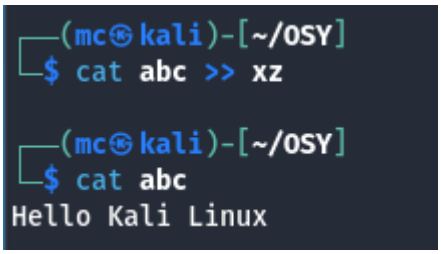


**5. Execute the following command.**

1) $ls a*n
2) $ls s?
3) $cat abc>> xz



- **Program code**

1. **Create four files a1,a2,a3**

→

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat a1 a2 a3

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1        a2  abc       chapter2     xaa
a1cp.txt  a3  chapter1  combine.txt  xz
```

2. **Apply different commands like ls, mv, cp, rm, join, split, and check the list of files at the end**

   1. **ls**

```
┌──(mc㉿kali)-[~/OSY]
└─$ touch a1.txt a2.txt a3.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1      a1cp.txt  a2.txt  a3.txt  chapter1  combine.txt  xz
a1.txt  a2        a3      abc     chapter2  xaa
```

   2. **mv**

```
┌──(mc㉿kali)-[~/OSY]
└─$ mv a1.txt /home/mc/Desktop

┌──(mc㉿kali)-[~/OSY]
└─$ la
a2.txt  a3.txt
```

```
┌──(mc㉿kali)-[~/OSY]
└─$ cd ..

┌──(mc㉿kali)-[~]
└─$ cd Desktop

┌──(mc㉿kali)-[~/Desktop]
└─$ ls
a1.txt
```

3. **rm**

```
┌──(mc㉿kali)-[~/OSY]
└─$ rm a1.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1        a2      a3      abc       chapter2     xaa
a1cp.txt  a2.txt  a3.txt  chapter1  combine.txt  xz
```

4. **join**

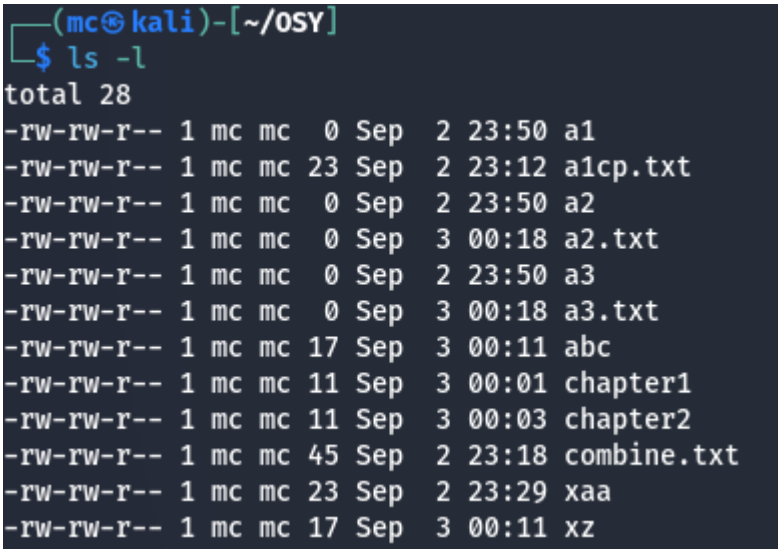
```
┌──(mc㉿kali)-[~/OSY]
└─$ cat  a1.txt
Hello there! I'm Mohit

┌──(mc㉿kali)-[~/OSY]
└─$ cat  a2.txt
 Chaudhari from CO5IA
```

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat combine.txt
Hello there! I'm Mohit
 Chaudhari from CO5IA

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1.txt  a1cp.txt  a2.txt  a3.txt  combine.txt
```

5. **split**

```
┌──(mc㉿kali)-[~/OSY]
└─$ split -12 a1.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1.txt  a1cp.txt  a2.txt  a3.txt  combine.txt  xaa

┌──(mc㉿kali)-[~/OSY]
└─$ cat xaa
Hello there! I'm Mohit
```

6. **ls -l**


(mc㉿kali)-[~/OSY]
$ ls -l
total 28
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a1
-rw-rw-r-- 1 mc mc 23 Sep  2 23:12 a1cp.txt
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a2
-rw-rw-r-- 1 mc mc  0 Sep  3 00:18 a2.txt
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc  0 Sep  3 00:18 a3.txt
-rw-rw-r-- 1 mc mc 17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc 11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc 11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc 45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc 23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc 17 Sep  3 00:11 xz

- **Set 5 questions**

1. **Execute file manipulation commands: ls, rm, mv, cp, join, split, ls), head, tail, touch.**

→

**1. ls**

```
┌──(mc㉿kali)-[~/OSY]
└─$ touch a1.txt a2.txt a3.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1      a1cp.txt  a2.txt  a3.txt  chapter1  combine.txt  xz
a1.txt  a2        a3      abc     chapter2  xaa
```

**2. mv**

```
┌──(mc㉿kali)-[~/OSY]
└─$ mv a1.txt /home/mc/Desktop

┌──(mc㉿kali)-[~/OSY]
└─$ la
a2.txt  a3.txt
```

```
┌──(mc㉿kali)-[~/OSY]
└─$ cd ..

┌──(mc㉿kali)-[~]
└─$ cd Desktop

┌──(mc㉿kali)-[~/Desktop]
└─$ ls
a1.txt
```

**3. rm**

```
┌──(mc㉿kali)-[~/OSY]
└─$ rm a1.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1        a2      a3      abc       chapter2     xaa
a1cp.txt  a2.txt  a3.txt  chapter1  combine.txt  xz
```

**4. join**

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat  a1.txt
Hello there! I'm Mohit

┌──(mc㉿kali)-[~/OSY]
└─$ cat  a2.txt
 Chaudhari from CO5IA
```

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat combine.txt
Hello there! I'm Mohit
 Chaudhari from CO5IA

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1.txt  a1cp.txt  a2.txt  a3.txt  combine.txt
```

**5. split**

```
┌──(mc㉿kali)-[~/OSY]
└─$ split -12 a1.txt

┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1.txt  a1cp.txt  a2.txt  a3.txt  combine.txt  xaa

┌──(mc㉿kali)-[~/OSY]
└─$ cat xaa
Hello there! I'm Mohit
```

**6. ls -l**

```
┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 28
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a1
-rw-rw-r-- 1 mc mc 23 Sep  2 23:12 a1cp.txt
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a2
-rw-rw-r-- 1 mc mc  0 Sep  3 00:18 a2.txt
-rw-rw-r-- 1 mc mc  0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc  0 Sep  3 00:18 a3.txt
-rw-rw-r-- 1 mc mc 17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc 11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc 11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc 45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc 23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc 17 Sep  3 00:11 xz
```

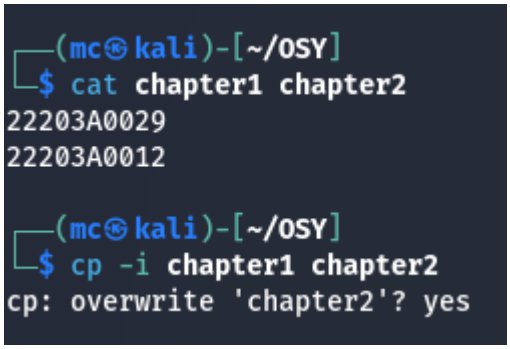
## 7. head

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat > a2
Hello
Mohit
Atharva
Shravan
Neel
Hitesh
Shelar
Parth
Abhishek
Vedant
Rane
Pranit
^C

┌──(mc㉿kali)-[~/OSY]
└─$ head a2
Hello
Mohit
Atharva
Shravan
Neel
Hitesh
Shelar
Parth
Abhishek
Vedant
```

## 8. Touch

```
┌──(mc㉿kali)-[~/OSY]
└─$ ls
a1  a1cp.txt  a2  a2.txt  a3  a3.txt  abc  chapter1  chapter2  combine.txt  dte  xaa  xz
```

**2. Write command to display prompt before copy the content of one file to another →**



**3. Explain the different use of cat command with example**

→

The **cat** command is one of the most useful commands in Linux – it is used for displaying, combining, and manipulating text files.

1. for creating new file:

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat > names
Mohit
Atharva
Shravan
```

2. For displaying content

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat  names
Mohit
Atharva
Shravan
```

3. For combining files

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat  names surname > combined1.txt

┌──(mc㉿kali)-[~/OSY]
└─$ cat combined1.txt
Mohit
Atharva
Shravan
Chaudhari
Jadhav
Salgaonkar
```

4. Append content to an existing file :

```
┌──(mc㉿kali)-[~/OSY]
└─$ cat surname >> name

┌──(mc㉿kali)-[~/OSY]
└─$ cat name
Chaudhari
Jadhav
Salgaonkar
```

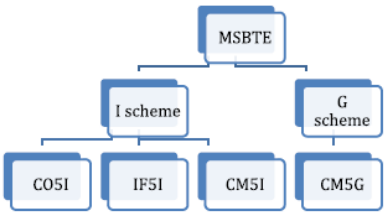
| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| **Subject:** OSY | **Subject Code**:22516 |
| **Semester:** 5th Semester | **Course:** Computer Engineering |
| **Laboratory No:** V118 | **Name of Subject teacher:** Prof. Natasha Brahmne |
| **Name of Student:** Siddharth Shah | **Roll Id:** 22203A0041 |

| | |
|---|---|
| **Experiment No:** | 7 |
| **Title of Experiment:** | Execute file and directory manipulation commands. |

- **Program Code:**

1. Create the following structure.




```
┌──(mc㉿kali)-[~]
└─$ mkdir MSBTE

┌──(mc㉿kali)-[~]
└─$ cd MSBTE

┌──(mc㉿kali)-[~/MSBTE]
└─$ mkdir Ischeme

┌──(mc㉿kali)-[~/MSBTE]
└─$ mkdir Gscheme

┌──(mc㉿kali)-[~/MSBTE]
└─$ cd Ischeme

┌──(mc㉿kali)-[~/MSBTE/Ischeme]
└─$ mkdir CO5I

┌──(mc㉿kali)-[~/MSBTE/Ischeme]
└─$ mkdir IF5I

┌──(mc㉿kali)-[~/MSBTE/Ischeme]
└─$ mkdir CM5I

┌──(mc㉿kali)-[~/MSBTE/Ischeme]
└─$ ls
CM5I  CO5I  IF5I

┌──(mc㉿kali)-[~/MSBTE/Ischeme]
└─$ cd ..

┌──(mc㉿kali)-[~/MSBTE]
└─$ cd Gscheme

┌──(mc㉿kali)-[~/MSBTE/Gscheme]
└─$ mkdir CM5G

┌──(mc㉿kali)-[~/MSBTE/Gscheme]
└─$ cd ..

┌──(mc㉿kali)-[~/MSBTE]
└─$ cd ..

┌──(mc㉿kali)-[~]
```


┌──(mc㉿kali)-[~]
└─$ ls *
Desktop:
Documents:
AUTORUN.INF VBoxDarwinAdditions.pkg VBoxWindowsAdditions-amd64.exe cert
NT3x VBoxDarwinAdditionsUninstall.tool VBoxWindowsAdditions-x86.exe runasroot.sh
OS2 VBoxLinuxAdditions.run VBoxWindowsAdditions.exe windows11-bypass.reg
TRANS.TBL VBoxSolarisAdditions.pkg autorun.sh
Downloads:
google-chrome-stable_current_amd64.deb
MSBTE:
Ischeme Ischeme
Music:
OSY:
XXXX.txt a1cp.txt a2.txt a3.txt chapter1 combine.txt dte names xaa
a1 a2 a3 abc chapter2 combined1.txt name surname xz
Pictures:
Public:
Shark:
1.pcapng
Templates:
Videos:

- ## Practical Related Questions

### 1) How to shift from Root Directory to User(Home) directory

Ans:

To shift from the root directory (/) to the home directory (/home/username) in a Unix-like operating system (like Linux or macOS), you can use the following commands:

**Using cd command:**

*cd /home/your_username*

### 2) How to see the directories

Ans: To see the directories, you can use the following commands:

1. **Using the ls command:**
   ls
   This command lists all files and directories in the current directory.
2. **Using ls with the -l option for detailed information:**
   ls -l
   This option provides a detailed list, including permissions, ownership, size, and modification date.
3. **Using ls with the -a option to show hidden directories:**
   ls -a
   This command lists all files and directories, including hidden ones (those starting with a dot .).
4. **Using ls with both -l and -a options together:**
   ls -la
   This command gives a detailed list of all files and directories, including hidden ones.

### 3) What is the default set of permissions given by the system to a directory?

Ans:

The default set of permissions given by the system to a newly created directory is typically 755 or *rwxr-xr-x*. This means:

- **Owner:** Has read (r), write (w), and execute (x) permissions.
- **Group:** Has read (r) and execute (x) permissions.
- **Others:** Has read (r) and execute (x) permissions.

These permissions ensure that the owner can fully manage the directory, while others can view and access its contents, but cannot modify them.

**4) How do you assign all permissions to your directory for all users using both symbolic and octal methods?**

Ans:

To assign all permissions (read, write, and execute) to all users (owner, group, and others) for a directory, you can use the following methods:

1. **Symbolic Method:**

*chmod u+rwx,g+rwx,o+rwx directory_name*

OR

*chmod a+rwx directory_name*

2. **Octal Method:**

*chmod 777 directory_name*

**5) What is the difference between the comm and cmp commands?**

Ans:

| Feature | comm Command | cmp Command |
|---|---|---|
| **Purpose** | Compares two sorted files line by line. | Compares two files byte by byte. |
| **Output** | Displays lines that are unique to each file or common between them. | Displays the first differing byte and its position (by default). |
| **Input Requirement** | Requires the files to be sorted. | Does not require the files to be sorted. |
| **Use Case** | Best for comparing text files with sorted lines. | Best for identifying binary differences between files. |
| **Options** | Can suppress specific columns (unique or common lines). | Can provide detailed reports of differences using options. |

- **Exercise:**

1. **Write the command for performing the following tasks sequentially.**

a. **Display your current directory.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ pwd
/home/mc/OSY
```

b. **Create a directory 'subject' in the current directory.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ mkdir subject
```

c. **Create a file 'sample' in the directory 'subject'.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ cd subject

┌──(mc㉿kali)-[~/OSY/subject]
└─$ touch sample
touch: cannot touch 'sample': Permission denied

┌──(mc㉿kali)-[~/OSY/subject]
└─$ sudo touch sample
[sudo] password for mc:

┌──(mc㉿kali)-[~/OSY/subject]
└─$ ls
sample
```

d. **Remove the write permission for the owner for 'sample' using symbolic method.**

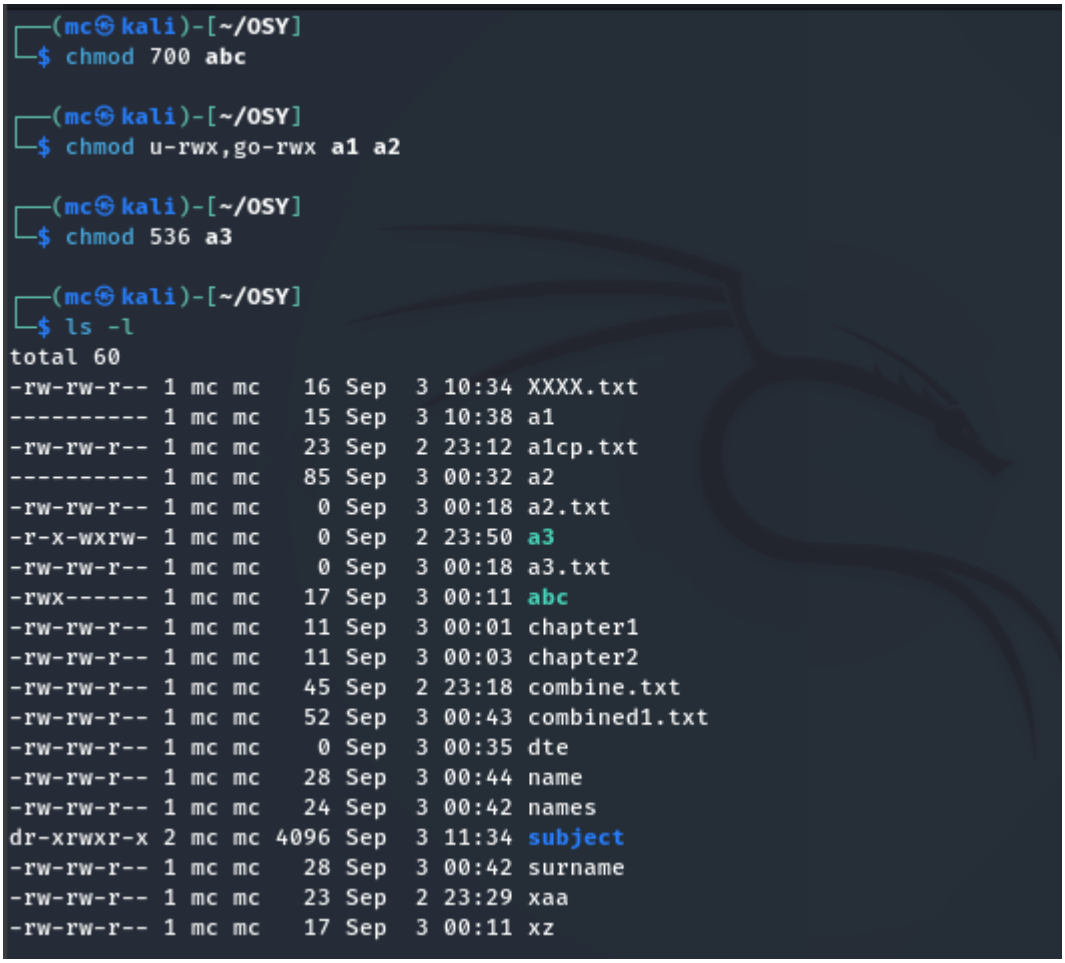
```
┌──(mc㉿kali)-[~/OSY/subject]
└─$ sudo chmod u-w sample

┌──(mc㉿kali)-[~/OSY/subject]
└─$ ls -l
total 0
-r--r--r-- 1 root root 0 Sep  3 11:32 sample
```

e. **Delete the file 'sample'. What is an error message displayed?**

```
┌──(mc㉿kali)-[~/OSY/subject]
└─$ rm sample
rm: remove write-protected regular empty file 'sample'? yes
rm: cannot remove 'sample': Permission denied
```

2. **What are the permissions assigned to the file after the execution of the following commands?**

a. **$chmod 700 abc**
b. **$chmod u+rwx,go-rwx file1 file2**
c. **$chmod 536 xyz**



3. **Create new files pqr and pqr1, perform the commands:**
   a. **$chmod ugo=r pqr**
   b. **$chmod ugo+r pqr1**

```
┌──(mc㉿kali)-[~/OSY]
└─$ chmod ugo=r dte

┌──(mc㉿kali)-[~/OSY]
└─$ chmod ugo+r dte

┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 60
-rw-rw-r-- 1 mc mc   16 Sep  3 10:34 XXXX.txt
---------- 1 mc mc   15 Sep  3 10:38 a1
-rw-rw-r-- 1 mc mc   23 Sep  2 23:12 a1cp.txt
---------- 1 mc mc   85 Sep  3 00:32 a2
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a2.txt
-r-x-wxrw- 1 mc mc    0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a3.txt
-rwx------ 1 mc mc   17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc   11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc   11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc   45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc   52 Sep  3 00:43 combined1.txt
-r--r--r-- 1 mc mc    0 Sep  3 00:35 dte
-rw-rw-r-- 1 mc mc   28 Sep  3 00:44 name
-rw-rw-r-- 1 mc mc   24 Sep  3 00:42 names
dr-xrwxr-x 2 mc mc 4096 Sep  3 11:34 subject
-rw-rw-r-- 1 mc mc   28 Sep  3 00:42 surname
-rw-rw-r-- 1 mc mc   23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc   17 Sep  3 00:11 xz
```

4. **Assign read and write permission for the owner, write permission for the group and execute permission for others using octal method for file ‘mfile’.**

```
┌──(mc㉿kali)-[~/OSY]
└─$ chmod 621 mfile

┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 60
-rw-rw-r-- 1 mc mc   16 Sep  3 10:34 XXXX.txt
---------- 1 mc mc   15 Sep  3 10:38 a1
-rw-rw-r-- 1 mc mc   23 Sep  2 23:12 a1cp.txt
---------- 1 mc mc   85 Sep  3 00:32 a2
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a2.txt
-r-x-wxrw- 1 mc mc    0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a3.txt
-rwx------ 1 mc mc   17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc   11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc   11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc   45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc   52 Sep  3 00:43 combined1.txt
-r--r--r-- 1 mc mc    0 Sep  3 00:35 dte
--w--w---x 1 mc mc    0 Sep  3 11:42 mfile
-rw-rw-r-- 1 mc mc   28 Sep  3 00:44 name
-rw-rw-r-- 1 mc mc   24 Sep  3 00:42 names
dr-xrwxr-x 2 mc mc 4096 Sep  3 11:34 subject
-rw-rw-r-- 1 mc mc   28 Sep  3 00:42 surname
-rw-rw-r-- 1 mc mc   23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc   17 Sep  3 00:11 xz
```

5. **Write commands to assign following permissions to the file OSY using octal method.**

a. **----------**

b. **–rw-r-xr--**

c. **-r-xr-xr-x**

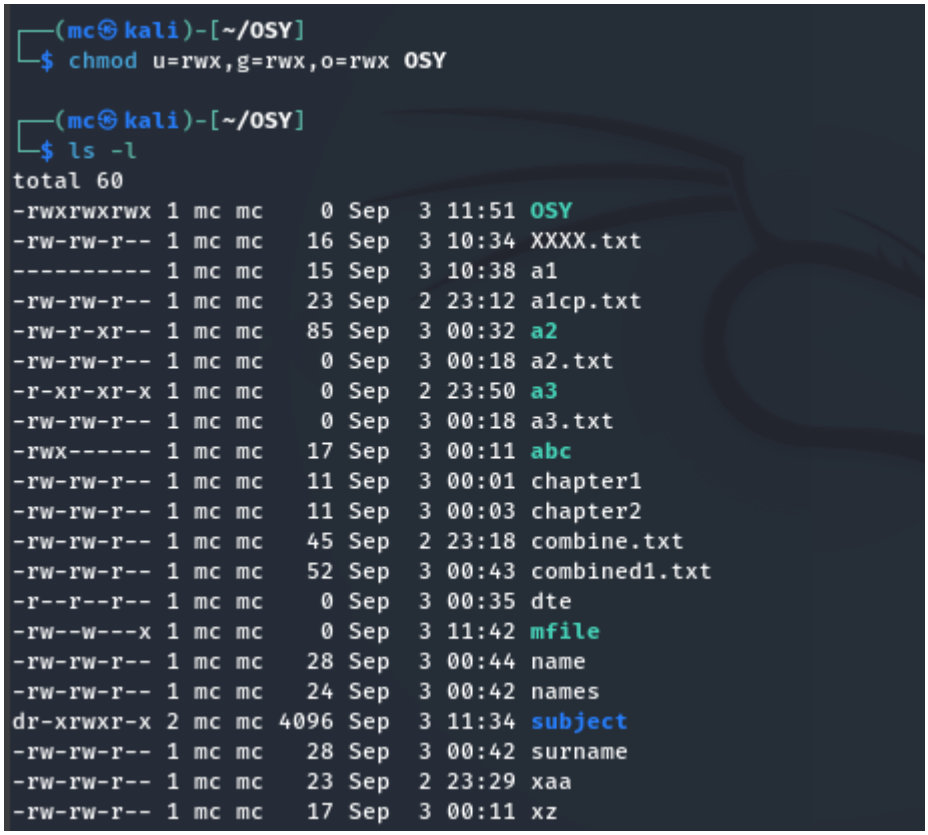
```
┌──(mc㉿kali)-[~/OSY]
└─$ chmod 000 a1

┌──(mc㉿kali)-[~/OSY]
└─$ chmod 654 a2

┌──(mc㉿kali)-[~/OSY]
└─$ chmod 555 a3

┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 60
-rw-rw-r-- 1 mc mc    16 Sep  3 10:34 XXXX.txt
---------- 1 mc mc    15 Sep  3 10:38 a1
-rw-rw-r-- 1 mc mc    23 Sep  2 23:12 a1cp.txt
-rw-r-xr-- 1 mc mc    85 Sep  3 00:32 a2
-rw-rw-r-- 1 mc mc     0 Sep  3 00:18 a2.txt
-r-xr-xr-x 1 mc mc     0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc     0 Sep  3 00:18 a3.txt
-rwx------ 1 mc mc    17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc    11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc    11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc    45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc    52 Sep  3 00:43 combined1.txt
-r--r--r-- 1 mc mc     0 Sep  3 00:35 dte
-rw--w---x 1 mc mc     0 Sep  3 11:42 mfile
-rw-rw-r-- 1 mc mc    28 Sep  3 00:44 name
-rw-rw-r-- 1 mc mc    24 Sep  3 00:42 names
dr-xrwxr-x 2 mc mc 4096 Sep  3 11:34 subject
-rw-rw-r-- 1 mc mc    28 Sep  3 00:42 surname
-rw-rw-r-- 1 mc mc    23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc    17 Sep  3 00:11 xz
```

6. **Write commands to assign following permissions to the file OSY using symbolic method.**

a. **-rwxr-xr--**

b. **-rwxrwxrwx**

```
┌──(mc㉿kali)-[~/OSY]
└─$ chmod u=rwx,g=rx,o=r OSY

┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 60
-rwxr-xr-- 1 mc mc     0 Sep  3 11:51 OSY
-rw-rw-r-- 1 mc mc    16 Sep  3 10:34 XXXX.txt
---------- 1 mc mc    15 Sep  3 10:38 a1
-rw-rw-r-- 1 mc mc    23 Sep  2 23:12 a1cp.txt
-rw-r-xr-- 1 mc mc    85 Sep  3 00:32 a2
-rw-rw-r-- 1 mc mc     0 Sep  3 00:18 a2.txt
-r-xr-xr-x 1 mc mc     0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc     0 Sep  3 00:18 a3.txt
-rwx------ 1 mc mc    17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc    11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc    11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc    45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc    52 Sep  3 00:43 combined1.txt
-r--r--r-- 1 mc mc     0 Sep  3 00:35 dte
-rw--w---x 1 mc mc     0 Sep  3 11:42 mfile
-rw-rw-r-- 1 mc mc    28 Sep  3 00:44 name
-rw-rw-r-- 1 mc mc    24 Sep  3 00:42 names
dr-xrwxr-x 2 mc mc 4096 Sep  3 11:34 subject
-rw-rw-r-- 1 mc mc    28 Sep  3 00:42 surname
-rw-rw-r-- 1 mc mc    23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc    17 Sep  3 00:11 xz
```


┌──(mc㉿kali)-[~/OSY]
└─$ chmod u=rwx,g=rwx,o=rwx OSY
┌──(mc㉿kali)-[~/OSY]
└─$ ls -l
total 60
-rwxrwxrwx 1 mc mc    0 Sep  3 11:51 OSY
-rw-rw-r-- 1 mc mc   16 Sep  3 10:34 XXXX.txt
---------- 1 mc mc   15 Sep  3 10:38 a1
-rw-rw-r-- 1 mc mc   23 Sep  2 23:12 a1cp.txt
-rw-r-xr-- 1 mc mc   85 Sep  3 00:32 a2
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a2.txt
-r-xr-xr-x 1 mc mc    0 Sep  2 23:50 a3
-rw-rw-r-- 1 mc mc    0 Sep  3 00:18 a3.txt
-rwx------ 1 mc mc   17 Sep  3 00:11 abc
-rw-rw-r-- 1 mc mc   11 Sep  3 00:01 chapter1
-rw-rw-r-- 1 mc mc   11 Sep  3 00:03 chapter2
-rw-rw-r-- 1 mc mc   45 Sep  2 23:18 combine.txt
-rw-rw-r-- 1 mc mc   52 Sep  3 00:43 combined1.txt
-r--r--r-- 1 mc mc    0 Sep  3 00:35 dte
-rw--w---x 1 mc mc    0 Sep  3 11:42 mfile
-rw-rw-r-- 1 mc mc   28 Sep  3 00:44 name
-rw-rw-r-- 1 mc mc   24 Sep  3 00:42 names
dr-xrwxr-x 2 mc mc 4096 Sep  3 11:34 subject
-rw-rw-r-- 1 mc mc   28 Sep  3 00:42 surname
-rw-rw-r-- 1 mc mc   23 Sep  2 23:29 xaa
-rw-rw-r-- 1 mc mc   17 Sep  3 00:11 xz

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Prof. Natasha Bhrahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 8 |
| Title of Experiment | Execute text processing commands. |

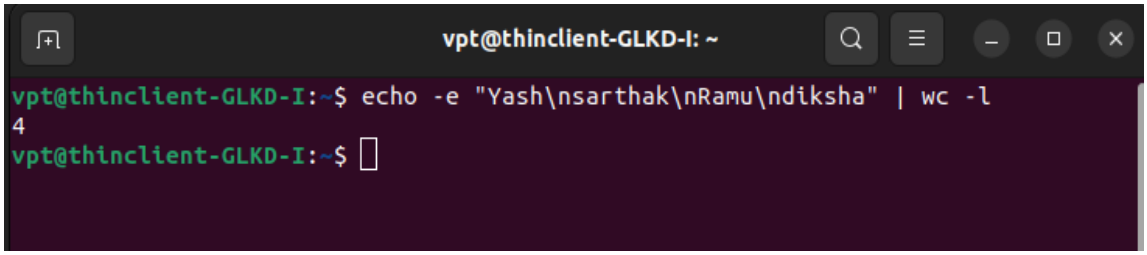
**Set Question:**

1. **Execute text processing commands: tr, wc, Explain tr command with examples**

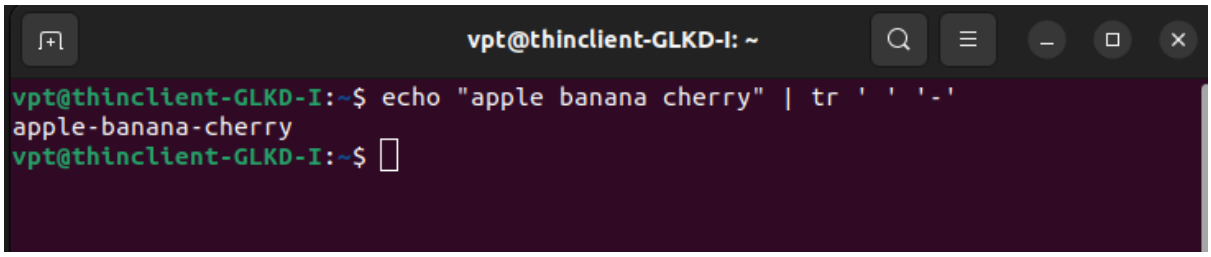
   **Ans: -**

   - **tr command**

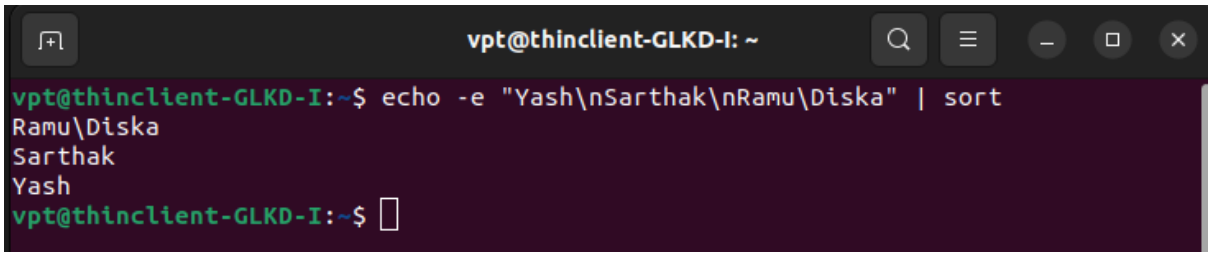

   - **wc command**



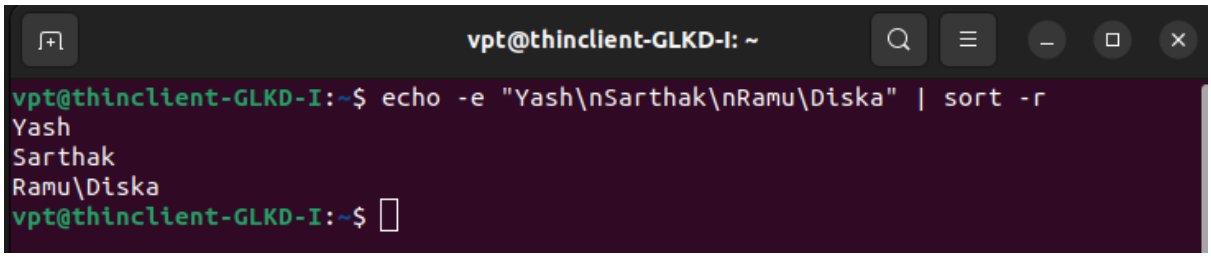
The tr (translate) command in Unix/Linux is a utility for translating or deleting characters from standard input and output. It's useful for text processing tasks such as converting characters, deleting specific characters, or squeezing repeated characters.
Eg:




2. **Explain sort command with different options and output.**
   **Ans:** -The sort command in Unix/Linux is used to sort lines of text files or standard input. It offers a variety of options to customize the sorting behavior, including sorting by numeric values, reversing the order, and sorting based on specific columns.
   i. **ascending order**



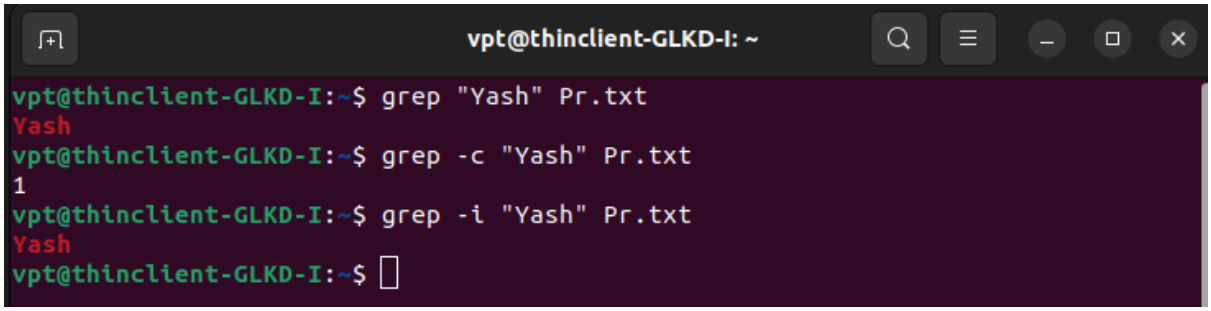

   ii. **reverse sorting**



### iii. numeric sorting


3. **Explain grep command with options**

   **Ans: -** The grep command is a powerful tool used in Unix/Linux systems for searching text using patterns. It prints lines from the input that match a specified pattern, making it invaluable for text processing and data extraction tasks. The `grep` command supports various options to refine and customize the search.




4. **Explain cut command with options.**

   **Ans: -** The cut command in Unix/Linux is used to extract specific sections from lines of a file or standard input. It is particularly useful for processing structured text data, such as columns in a CSV file or fields in a log file. The `cut` command allows you to specify delimiters and fields to extract.

## XII. Practical Related Questions

1) **Give the applications of paste commands**

**Ans: -** Paste command is one of the useful commands in Unix or Linux operating system. It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output.

2) **How to move cursor at the end of line**.

**Ans: -** Ctrl+E or End command can be used moves the cursor to the end of the line.

3) **What are options of wc commands.**

**Ans: -**

**i)** -l: This option prints the number of lines present in a file. With this option wc command displays two-columnar output, 1st column shows number of lines present in a file and 2nd itself represent the file name.

**ii)** -w: This option prints the number of words present in a file. With this option wc command displays two-columnar output, 1st column shows number of words present in a file and 2nd is the file name.
**iii)** -c: This option displays count of bytes present in a file. With this option it display two-columnar output, 1st column shows number of bytes present in a file and 2nd is the
file name.
**iv)** -m: Using -m option 7wc8 command displays count of characters from a file
**v)** -L: The 7wc8 command allow an argument -L, it can be used to print out the length of longest (number of characters) line in a file. So, we have the longest character line Arunachal Pradesh in a file state.txt and Hyderabad in the file capital.txt. But with this option if more than one file name is specified then the last row i.e. the extra row, doesn't display total but it display the maximum of all values displaying in the first column of individual files
**vi)** –version: This option is used to display the version of wc which is currently running on your system.

4) **What are different types of filters used in Linux**

**Ans.**

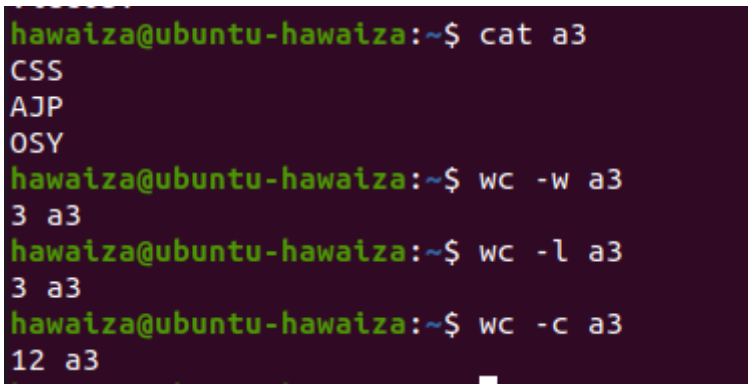
- cat : Displays the text of the file line by line.
- head : Displays the first n lines of the specified text files. If the number of lines is not specified then by default prints first 10 lines
- tail : It works the same way as head, just in reverse order. The only difference in
  tail is, it returns the lines from bottom to up.
- sort : Sorts the lines alphabetically by default but there are many options available to modify the sorting mechanism. Be sure to check out the man page to see everything it can do
- uniq : Removes duplicate lines. uniq has a limitation that it can only remove continuous duplicate lines(although this can be fixed by the use of piping). Assuming we have the following data.
- wc: wc command gives the number of lines, words and characters in the data

5) **What is difference between $cat abc and $cat abc more.**

**Ans: -** Cat displays file contents. If the file is large the contents scroll off the screen before we view it. So, command 'more' is like a pager which displays the contents page by page.

**Program Code:**

**1. Write the commands for:**
**Counting number of words in the 'data.txt'**
**Counting number of line in the 'data.txt'**
**Counting all characters in the 'data.txt'**
**Ans: -**



# Exercise:

1. **Write the significance of the following.**
   i. **Only one character is specified**

```
hawaiza@ubuntu-hawaiza:~$ paste -d "|" number state capital
1|Maharashtra|Mumbai
2|Andhra Pradesh|Hyderabad
3|Assam|Dispur
```

   ii. **More than one character is specified**

   **Ans: -**

```
hawaiza@ubuntu-hawaiza:~$ paste -d "|," number state capital
1|Maharashtra,Mumbai
2|Andhra Pradesh,Hyderabad
3|Assam,Dispur
```

   iii. **–s (serial), combination of –d and –s, --version (write its syntax and example)**

   **Ans: -**

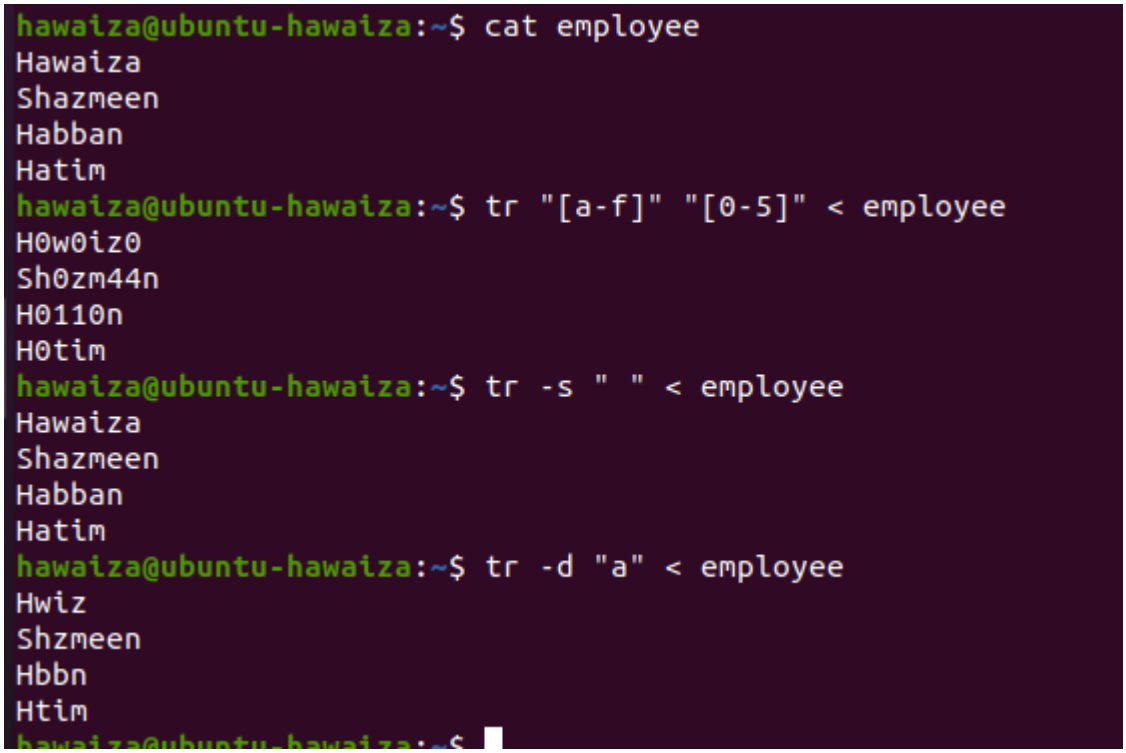
```
hawaiza@ubuntu-hawaiza:~$ paste -s number state capital
1       2       3
Maharashtra     Andhra Pradesh  Assam
Mumbai  Hyderabad       Dispur
```

```
hawaiza@ubuntu-hawaiza:~$ paste -s -d ", " number state capital
1,2 3
Maharashtra,Andhra Pradesh Assam
Mumbai,Hyderabad Dispur
```

```
hawaiza@ubuntu-hawaiza:~$ paste --version
paste (GNU coreutils) 8.30
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by David M. Ihnat and David MacKenzie.
```

2. **Try the commands and write the output with its meaning**
   i. **$tr “[a-f]” “[0-5]” < employee**
   ii. **$tr -s “ ” < employee**
   iii. **$tr -d “f” < employee**

   **Ans: -**



| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |
|---|---|

| | |
|---|---|
| Subject: OSY | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Natasha Bramhne |
| Name of Student: Siddharth P. Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 9 |
| Title of Experiment | Use vi editor and perform all editor commands |

# X1. EXERCISE

1. write significance of following

i. $vi temp.txt,

ii. insert multiple lines,

iii. deleting contents using commands

Ans:

**i. $vi temp.txt:**

This command opens the file "temp.txt" in the vi editor, where you can view, edit, and manipulate its contents in different modes (insert, command, visual, etc.).

**ii. Insert multiple lines:**

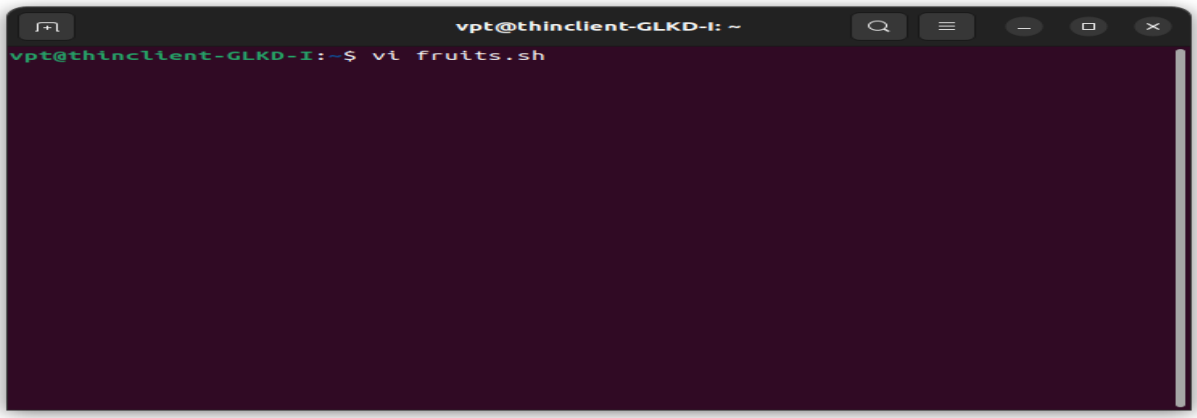
In vi, pressing i switches to insert mode, allowing you to add multiple lines of text. Press Esc to return to command mode when done.

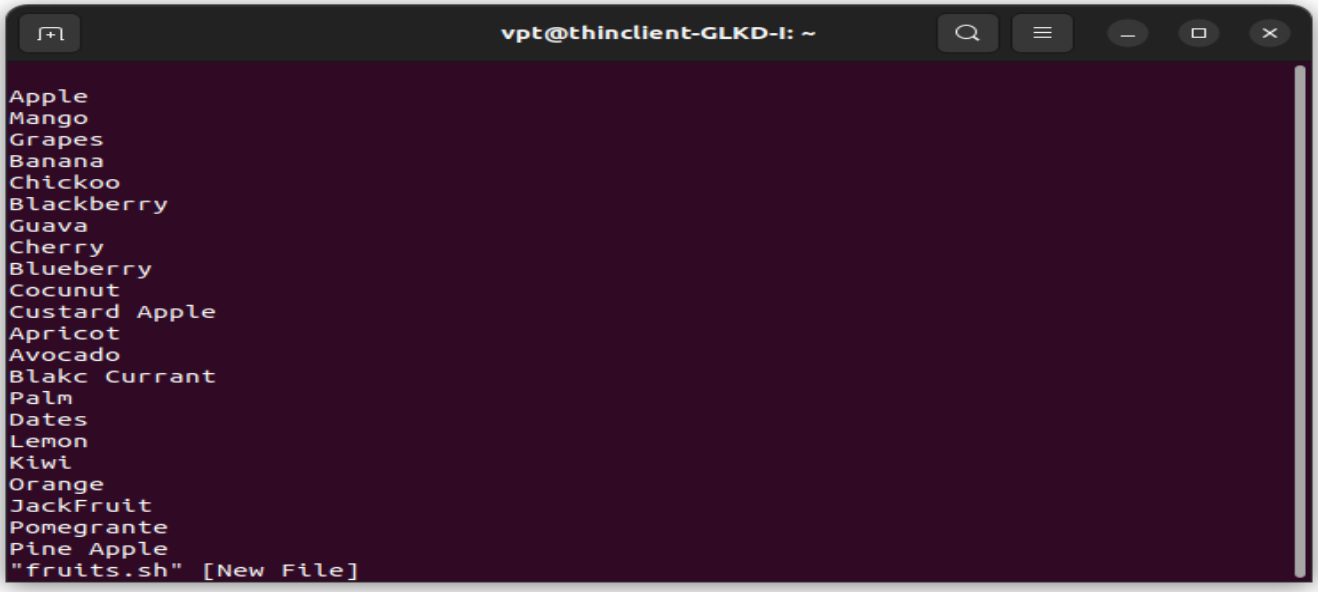
**iii. Deleting contents using commands:**

Use dd in command mode to delete a single line, or d<number>d to delete multiple lines. The :x or :wq commands will save and exit after deletion.

2. Create a new file and practice and executing shell commands from within the editor. Capture the results of some shell commands into the file

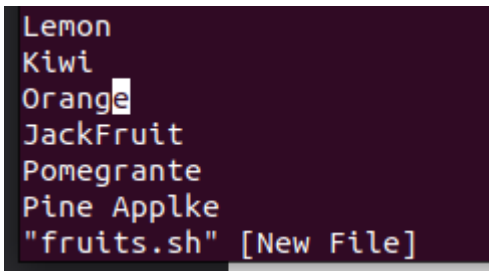
Ans:

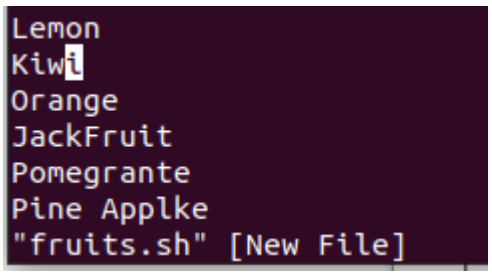


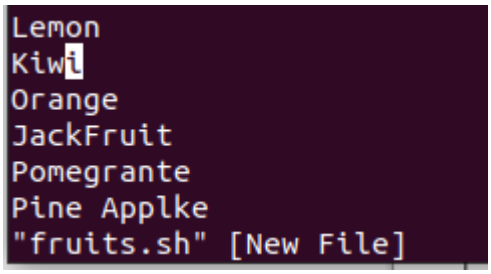
| Sr. No. | Command | Description |
|---|---|---|
| 1 | **k** | Moves the cursor up one line |
| 2 | **j** | Moves the cursor down one line |
| 3 | **h** | Moves the cursor to the left one character position |
| 4 | **l** | Moves the cursor to the right one character position |

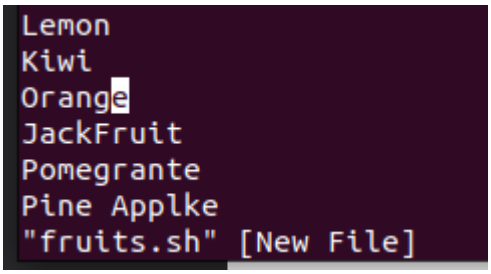
**1. k**



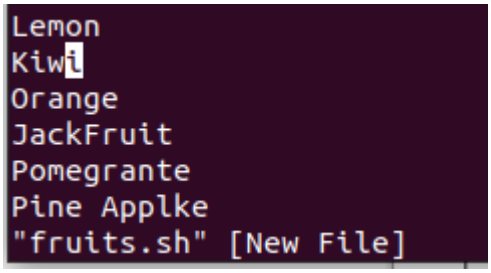



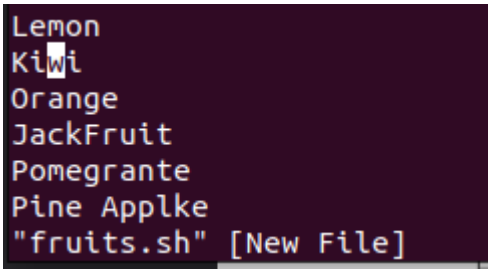

**2. j**



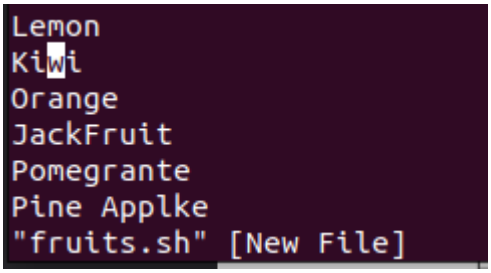



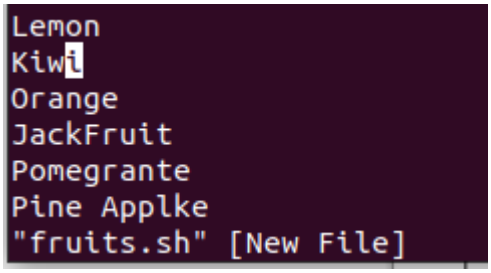

**3. h**



**4. l**

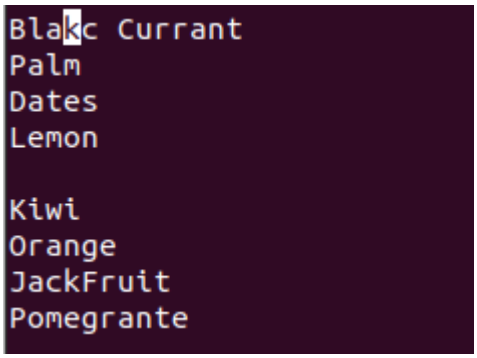


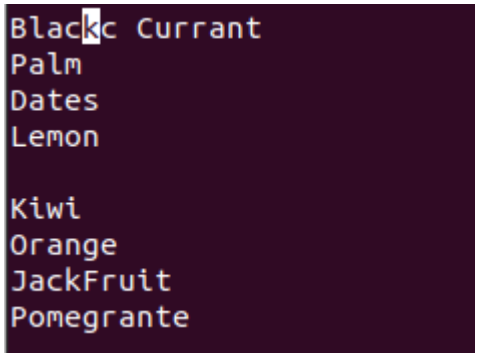




**Editing Files**
To edit the file, you need to be in the insert mode. There are many ways to enter the insert mode from the command mode –

| Sr. No. | Command | Description |
|---|---|---|
| 1 | **i** | Inserts text before the current cursor location |
| 2 | **I** | Inserts text at the beginning of the current line |
| 3 | **a** | Inserts text after the current cursor location |
| 4 | **A** | Inserts text at the end of the current line |
| 5 | **o** | Creates a new line for text entry below the cursor location |
| 6 | **O** | Creates a new line for text entry above the cursor location |

**1. i**



**2. I**

```
Blackc Currant
Palm
Dates
Lemon

Kiwi
Orange
JackFruit
Pomegrante
```

```
FruitBlackc Currant
Palm
Dates
Lemon

Kiwi
Orange
JackFruit
Pomegrante
```

**3. a**

```
Blueberry
Cocunut
Custard Apple
Apricot
Avocado
FruitBlackc Currant
```

```
BlueberryFruit
Cocunut
Custard Apple
Apricot
Avocado
FruitBlackc Currant
```

**4. A**

```
BlueberryFruit
Cocunut
Custard Apple
Apricot
Avocado
FruitBlackc Currant
```

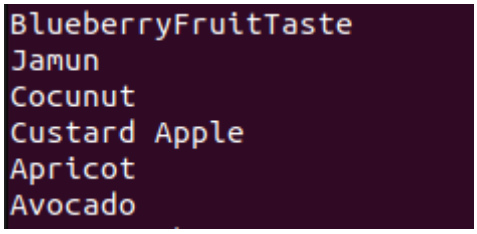
```
BlueberryFruitTaste
Cocunut
Custard Apple
Apricot
Avocado
FruitBlackc Currant
```

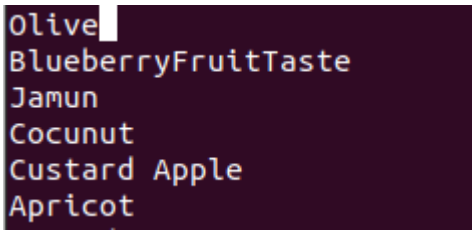
**5. o**

```
BlueberryFruitTaste
Cocunut
Custard Apple
Apricot
Avocado
FruitBlackc Currant
```

```
BlueberryFruitTaste
Jamun
Cocunut
Custard Apple
Apricot
Avocado
```

**6. O**

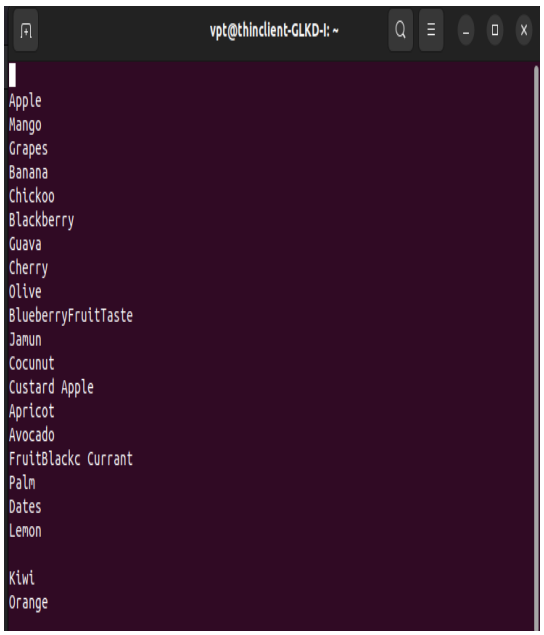


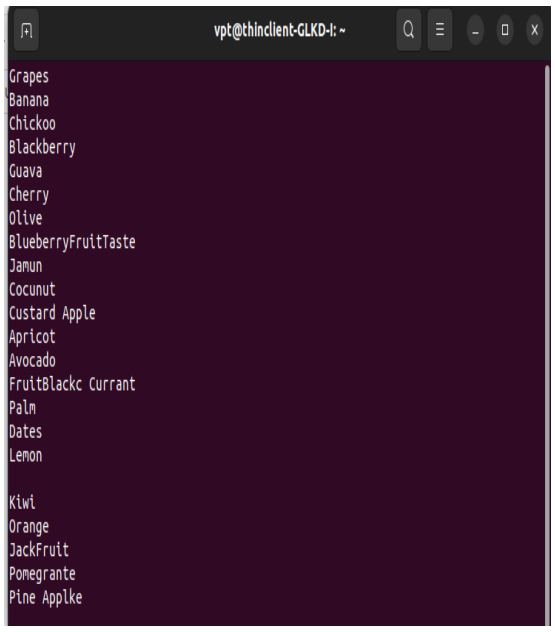




**Page movement Description:-**

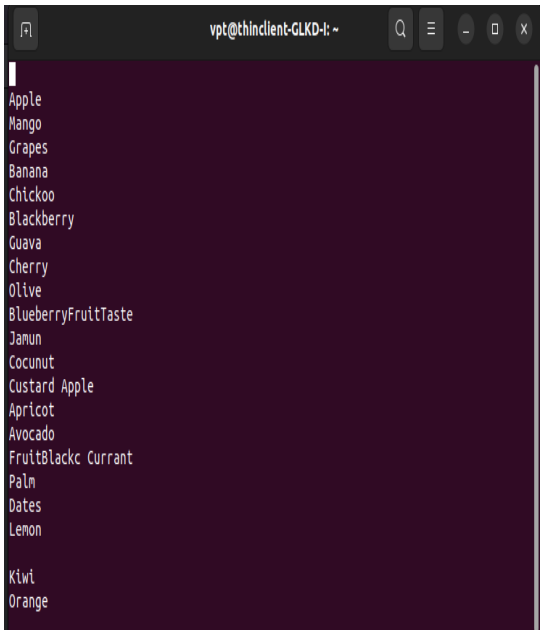
| Sr. No. | Command | Description |
|---|---|---|
| 1 | CTRL+d - | Move forward 1/2 screen |
| 2 | CTRL+f- | Move forward one full screen |
| 3 | CTRL+u - | Move backward 1/2 screen |
| 4 | CTRL+b | Move backward one full screen |
| 5 | CTRL+e | Moves screen up one line |
| 6 | CTRL+y- | Moves screen down one line |
| 7 | CTRL+I | Redraws screen |

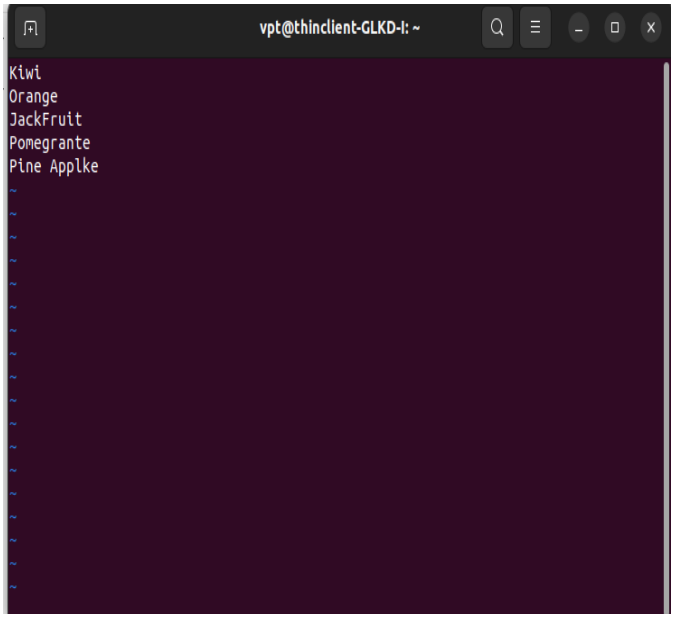
**1. CTRL+d**



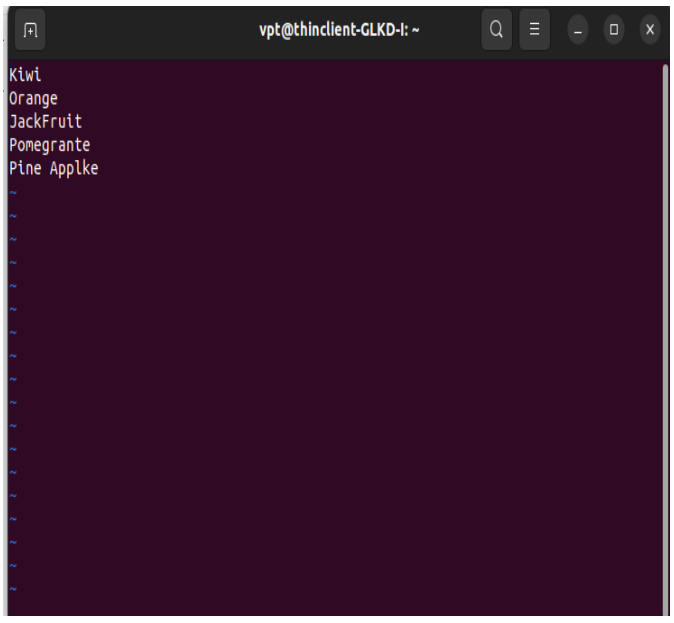



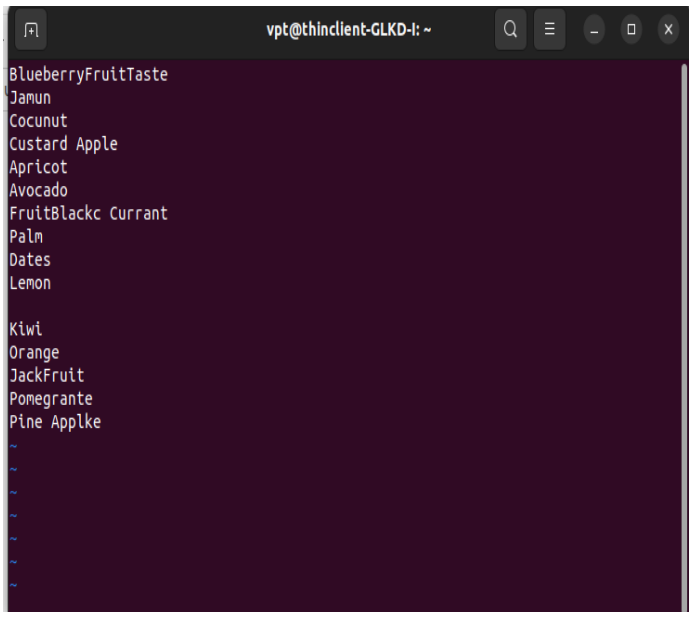
## 2. CTRL+f



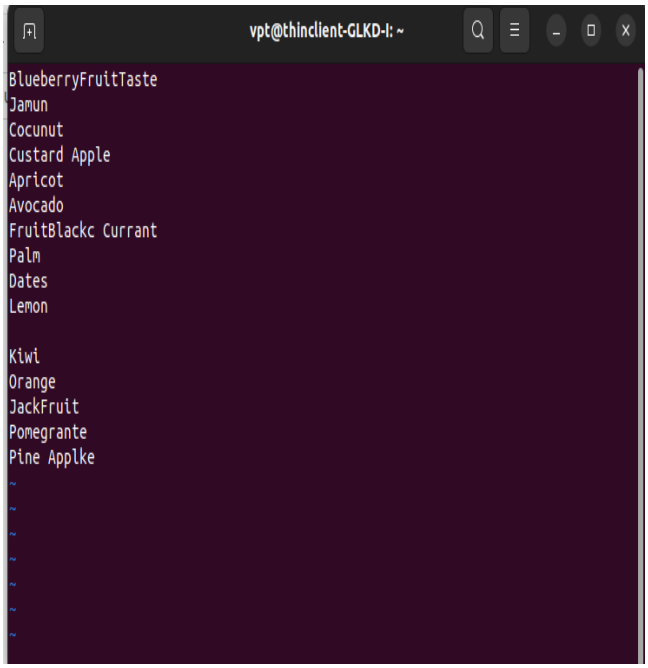



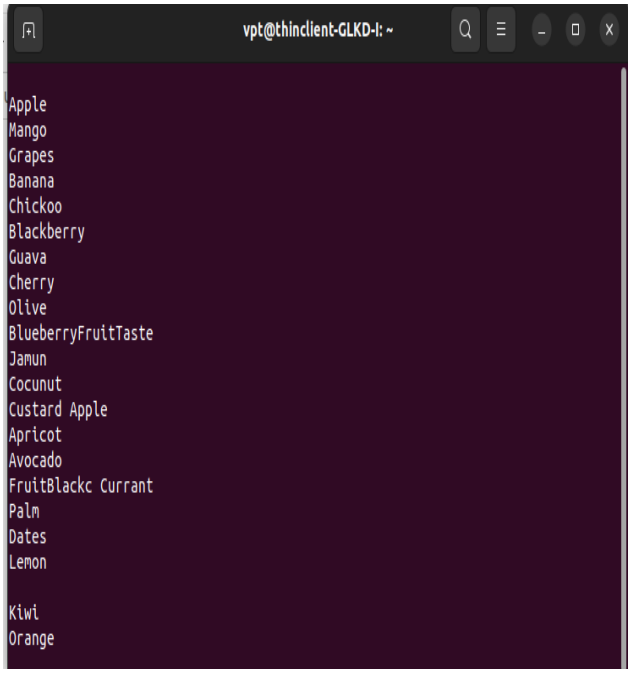

## 3. CTRL+u



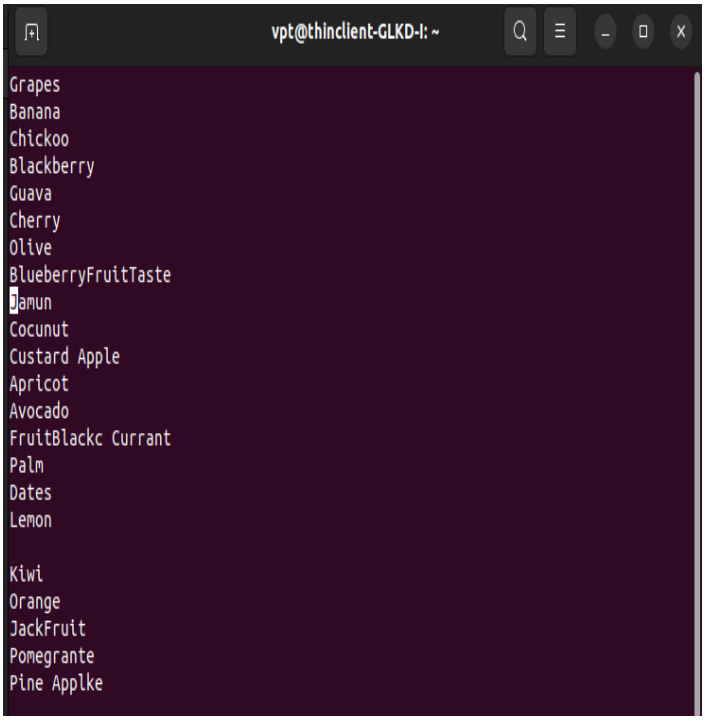



## 4. CTRL+b

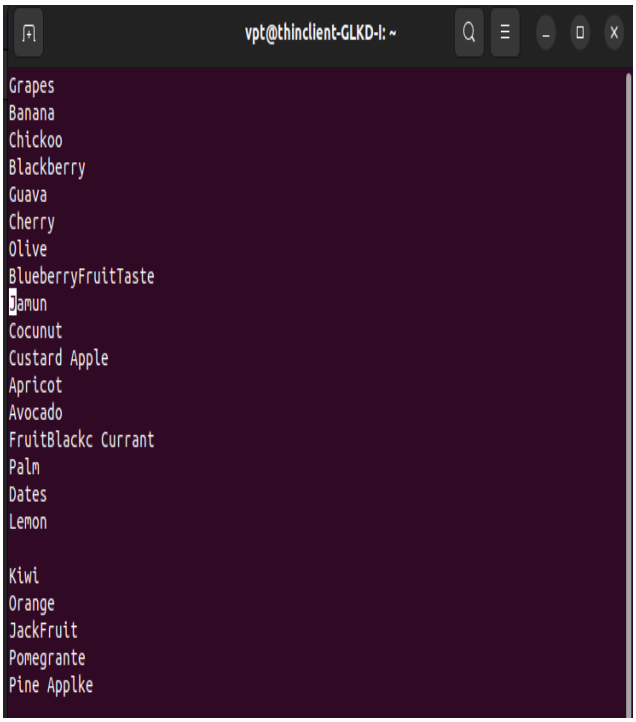


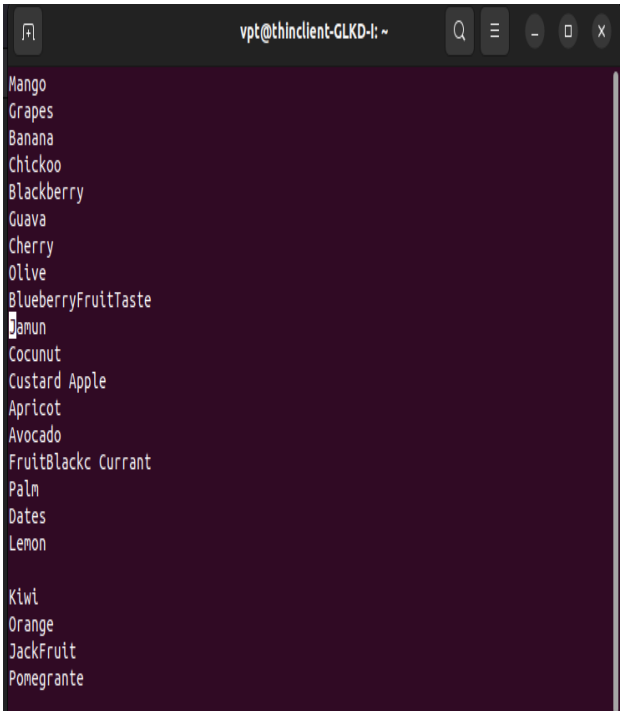




## 5. CTRL+e

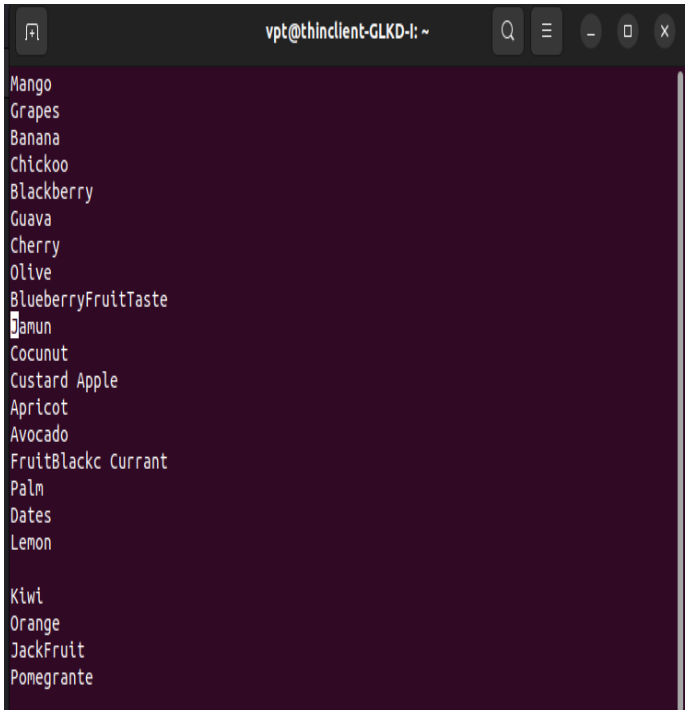



## 6. CTRL+y




## 7. CTRL+l

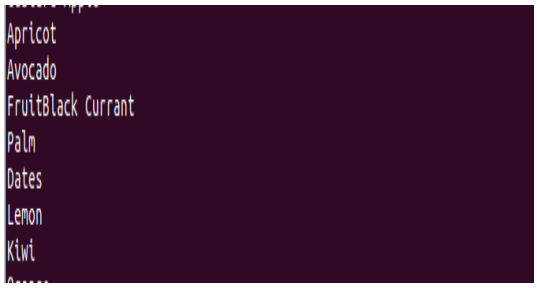


**Deleting Characters**

Here is a list of important commands, which can be used to delete characters and lines in an open file –

| Sr. No. | Command | Description |
|---|---|---|
| 1 | x | Deletes the character under the cursor location |
| 2 | X | Deletes the character before the cursor location |
| 3 | Dw | Deletes from the current cursor location to the next word |
| 4 | d^ | Deletes from the current cursor position to the beginning of the line |
| 5 | d$ | Deletes from the current cursor position to the end of the line |
| 6 | D | Deletes from the cursor position to the end of the current line |
| 7 | dd | Deletes the line the cursor is on |

**1. x**


**2. X**


**3. Dw**


Lemon
Kiwi
Orange
JackFruit
Pomegrante
Pine Apple
~


Lemon
Kiwi
Orange
JackFruit
Pomegrante
~

**4. d^**


Lemon
Kiwi
Orange
JackFruit
Pomegrante
~


Lemon
Kiwi
Orange
JackFruit
e
~

**5. d$**


Lemon
Kiwi
Orange
JackFruit
e
~


Lemon
Kiwi
Orange
Jac
e
~
~

**6. D**


Lemon
Kiwi
Orange
Jac
e
~
~


Lemon
Kiwi
Or
Jac
e
~
~

**7. dd**



**Change Commands**

You also have the capability to change characters, words, or lines in vi without deleting them.

| Sr. No. | Command | Description |
|---|---|---|
| 1 | cc | Removes the contents of the line, leaving you in insert mode. |
| 2 | cw | Changes the word the cursor is on from the cursor to the lowercase w end of the word. |
| 3 | r | Replaces the character under the cursor. vi returns to the command mode after the replacement is entered. |
| 4 | R | Overwrites multiple characters beginning with the character currently under the cursor. You must use Esc to stop the overwriting. |
| 5 | s | Replaces the current character with the character you type. Afterward, you are left in the insert mode. |
| 6 | S | Deletes the line the cursor is on and replaces it with the new text. After the new text is entered, vi remains in the insert mode. |

**1. cc**


**2. cw**


Banana
Chickoo
BlackberryFruit
Cherry
Olive


Banana
Chickoo
BlackberryFruit
Cherr$
Olive
BlueberryFruitTaste

**3. r**


Banana
Chickoo
BlackberryFruit
Cherr$
Olive
BlueberryFruitTaste


Banana
Chickoo
BlackberryFruit
Alive
BlueberryFruitTaste
Jamun

**4. R**


Banana
Chickoo
BlackberryFruit
Alive
BlueberryFruitTaste
Jamun


Banana
Chickoo
BlackberryFruit
Hello
BlueberryFruitTaste
Jamun

**5. s**


Banana
Chickoo
BlackberryFruit
Hello
BlueberryFruitTaste
Jamun


Banana
Chickoo
BlackberryFruit
Hell$
BlueberryFruitTaste
Jamun

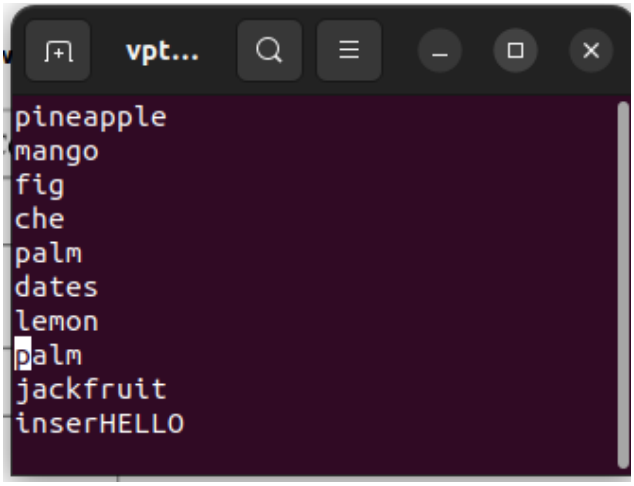
**6. S**



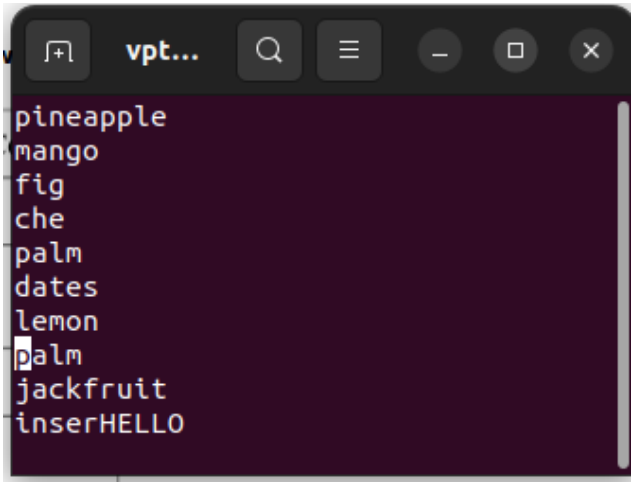
## Copy and Paste Command

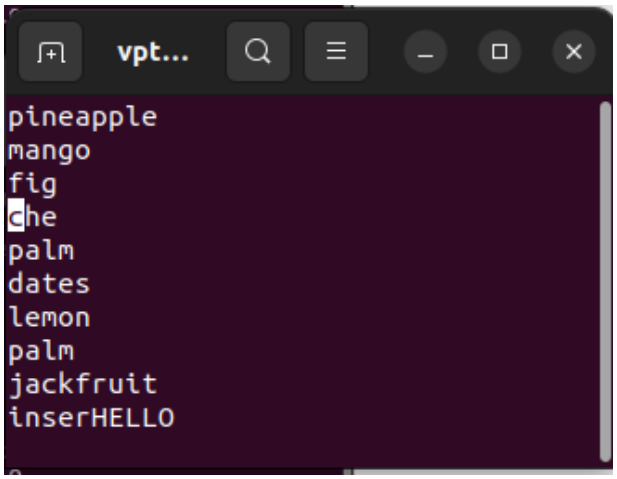
| Sr. No. | Command | Description |
|---|---|---|
| 1 | yy | Copies the current line. |
| 2 | yw- | Copies the current word from the character the lowercase w cursor is on, until the end of the word. |
| 3 | p | Puts the copied text after the cursor |
| 4 | P | Puts the yanked text before the cursor. |

**1. yy**

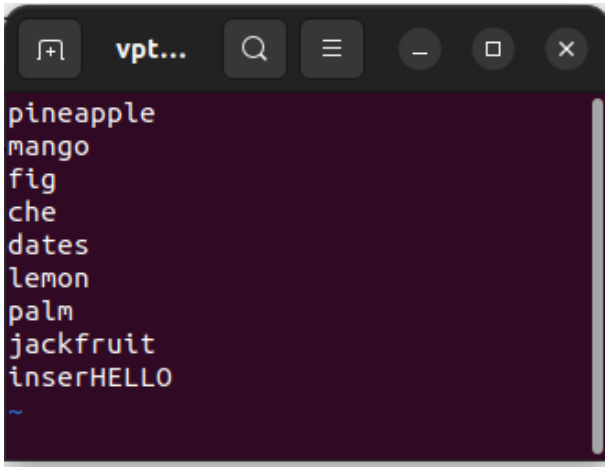


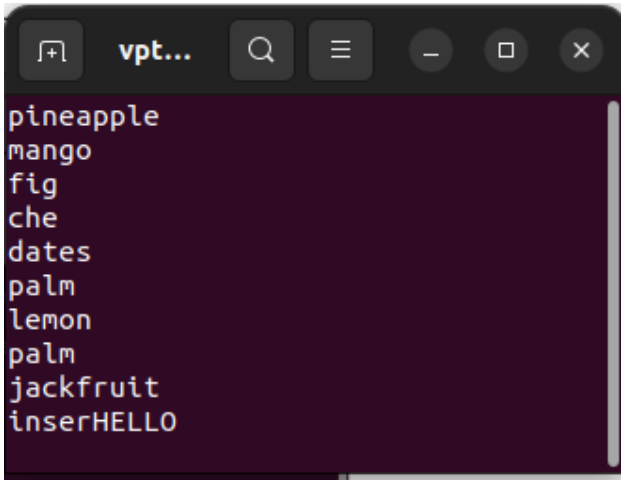
**2. yw-**


vpt...
pineapple
mango
fig
che
palm
dates
lemon
palm
jackfruit
inserHELLO

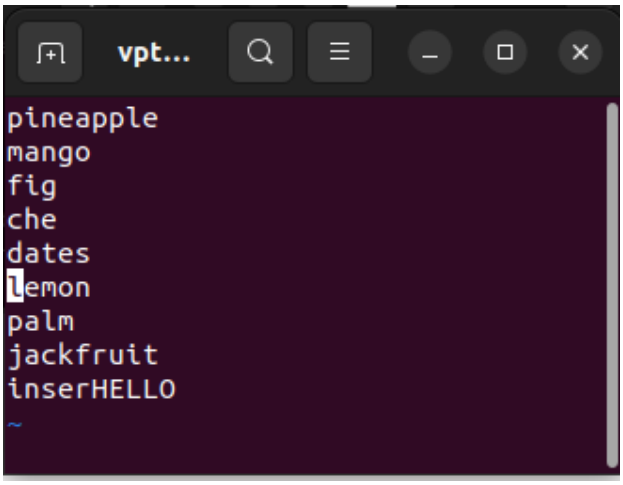

vpt...
pineapple
mango
fig
che
palm
dates
lemon
palm
jackfruit
inserHELLO

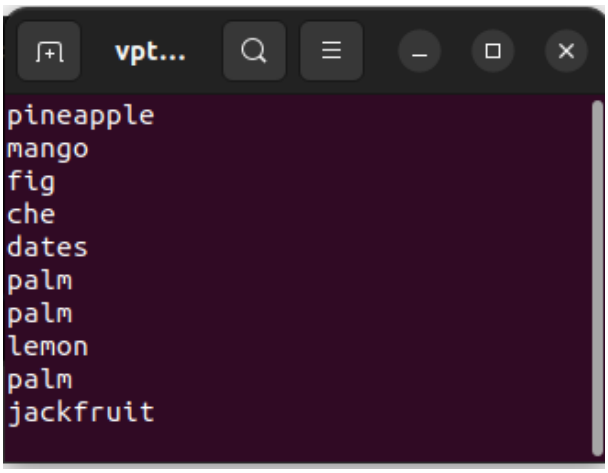
**3. p**


vpt...
pineapple
mango
fig
che
dates
lemon
palm
jackfruit
inserHELLO
~


vpt...
pineapple
mango
fig
che
dates
palm
lemon
palm
jackfruit
inserHELLO

**4. P**


vpt...
pineapple
mango
fig
che
dates
lemon
palm
jackfruit
inserHELLO
~

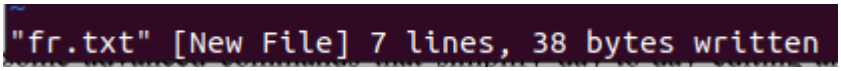

vpt...
pineapple
mango
fig
che
dates
palm
palm
lemon
palm
jackfruit

**Ex mode command**

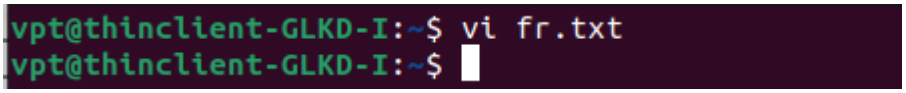
Press Esc key and then (:) colon to enter ex-mode commands. A colon is displayed at the left hand corner of the last line on your screen.

| Command | Action |
|---|---|
| :w | Saves file and remains in editing mode |
| :x | Saves file and quits editing mode |
| :wq | Save and quit |
| :q | Quits vi when no changes are made |
| :q! | Quits vi cancelling the changes |
| :sh | Escape to Unix shell |

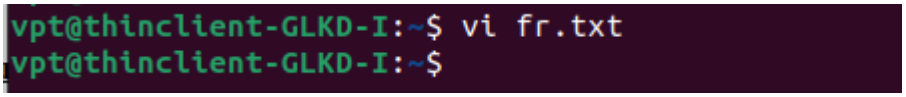
**1. :w**



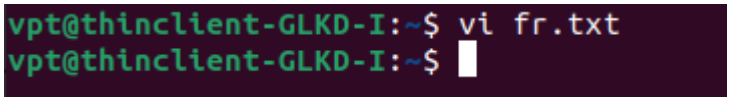

**2. :x**




**3. :wq**




**4. :q**



**5. :q!**

```
vpt@thinclient-GLKD-I:~$ vi fr.txt
vpt@thinclient-GLKD-I:~$
```

**6. :sh**

```
vpt@thinclient-GLKD-I:~$
```

## 3. How to get help?

Ans:

To get help in the vi editor:
In command mode, type :help and press Enter to access the help documentation within the editor. It provides a comprehensive guide to vi commands and usage.

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Prof. Natasha Brahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 10 |
| Title of Experiment | Execute shell script by using if statement |

## SETS QUESTIONS

1) **Write shell script to display square of two numbers.**

➔

```
#!/bin/bash
echo "Enter the first number:"
read num1
echo "Enter the second number:"
read num2
square1=$((num1 * num1))
square2=$((num2 * num2))
echo "The square of $num1 is: $square1"
echo "The square of $num2 is: $square2"
```

**Output:-**

```
Enter the first number:
12
Enter the second number:
11
The square of 12 is: 144
The square of 11 is: 121
```

**2) Write a shell script for menu driven program.**

➔

```
#!/bin/bash

while true;
do
  echo "1. Add two numbers"
  echo "2. Subtract two numbers"
  echo "3. Multiply two numbers"
  echo "4. Divide two numbers"
  echo "5. Exit"
  echo "Enter your choice:"

  read choice

  case $choice in
    1)
      echo "Enter the first number:"
      read num1
      echo "Enter the second number:"
      read num2
      result=$((num1 + num2))
      echo "The result of addition is: $result"
      ;;
    2)
      echo "Enter the first number:"
      read num1
      echo "Enter the second number:"
      read num2
      result=$((num1 - num2))
      echo "The result of subtraction is: $result"
      ;;
    3)
      echo "Enter the first number:"
      read num1
      echo "Enter the second number:"
      read num2
      result=$((num1 * num2))
      echo "The result of multiplication is: $result"
      ;;
```

```
    4)
      echo "Enter the first number:"
      read num1
      echo "Enter the second number:"
      read num2
      if [ $num2 -ne 0 ]; then
        result=$((num1 / num2))
        echo "The result of division is: $result"
      else
        echo "Error: Division by zero is not allowed!"
      fi
      ;;
    5)
      exit 0
      ;;
    *)
      echo "Invalid choice!"
      ;;
  esac
done
```

**Output:-**

```
1. Add two numbers
2. Subtract two numbers
3. Multiply two numbers
4. Divide two numbers
5. Exit
Enter your choice:
3
Enter the first number:
22
Enter the second number:
48
The result of multiplication is: 1056
1. Add two numbers
2. Subtract two numbers
3. Multiply two numbers
4. Divide two numbers
5. Exit
Enter your choice:
4
Enter the first number:
12
Enter the second number:
0
Error: Division by zero is not allowed!
1. Add two numbers
2. Subtract two numbers
3. Multiply two numbers
4. Divide two numbers
5. Exit
Enter your choice:
```

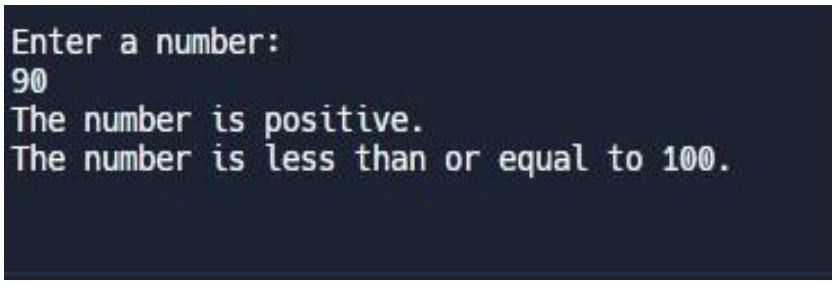
## XII. PRACTRICAL RELATED QUESTIONS.

1) **Write and execute script for nested if statement.**

➔

```
#!/bin/bash

echo "Enter a number:"
read number

if [ $number -gt 0 ]; then
   echo "The number is positive."

   if [ $number -gt 100 ];
   then
      echo "The number is greater than 100."
   else
      echo "The number is less than or equal to 100."
   fi

   elif [ $number -lt 0 ];
   then
         echo "The number is negative."
   else
         echo "The number is zero."
   fi
```

**Output:-**



2) **Write difference between**
   i. **if [condition]**
   ii. **if ((condition))**

➔

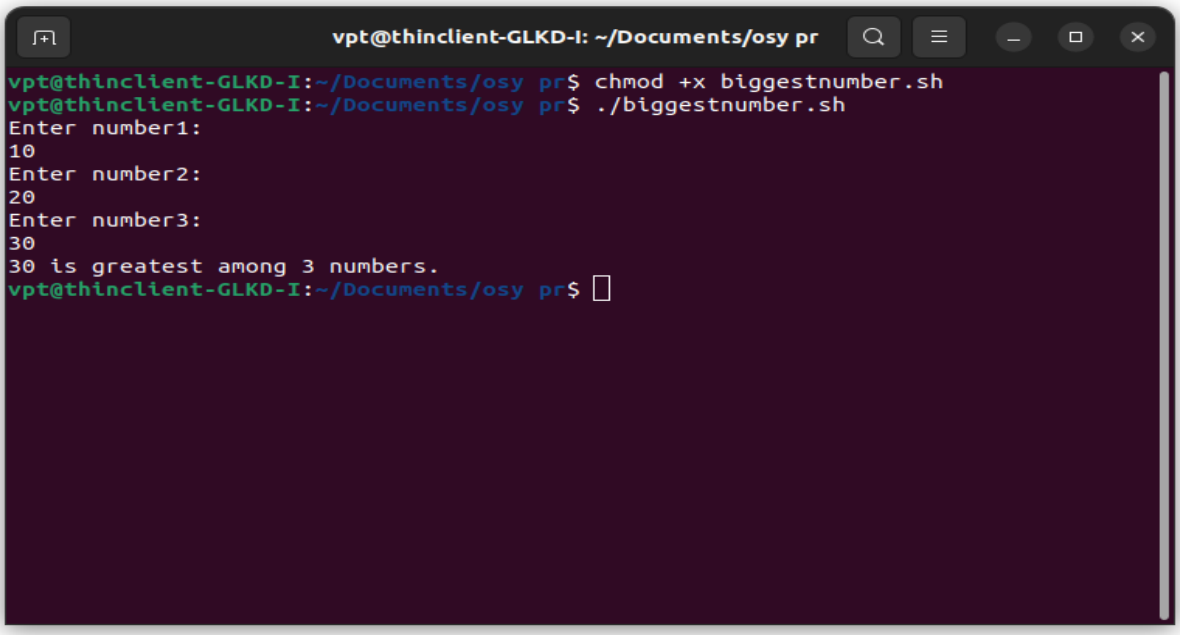
| **if[condition]** | **If((condition))** |
|---|---|
| The single brackets […] is the command | The double parentheses ((…)) is the format for bash arithmetic expansion. |
| It is used to create commands in statements. | It is used to test an arithmetic operation. |
| Syntax:-<br>if [condition]<br>then<br>### series of code<br>fi | Syntax:-<br>if ((condition))<br>then<br>Statement goes here<br>fi |
| Example:-<br>if [“X” -lt “0”]<br>then<br>echo “X is less than zero”<br>fi | Example:-<br>if (($num -eq 42))<br>then<br>echo “num is actually equal to 42”<br>else<br>echo “num is not equal to 42”<br>fi |

**3) Write script for finding greatest number among the given numbers.**

➔

```
#!/bin/bash
echo "Enter number1: "
read num1
echo "Enter number2: "
read num2
echo "Enter number3: "
read num3
if [ "$num1" -ge "$num2" ] && [ "$num1" -ge "$num3" ]
then
echo "$num1 is greatest among 3 numbers."
elif [ "$num2" -ge "$num1" ] && [ "$num2" -ge "$num3" ]
then
echo "$num2 is greatest among 3 numbers."
else
echo "$num3 is greatest among 3 numbers."
fi
```

**Output:-**



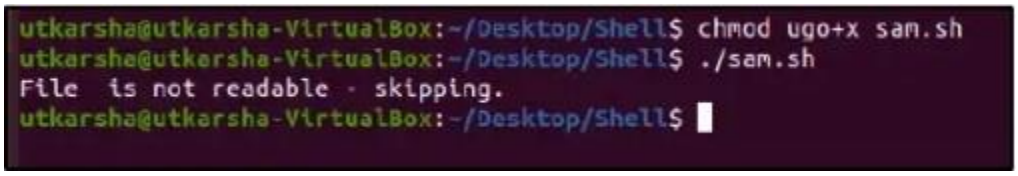
## XIII. EXERCISE.

1) **Correct the following script and write its output.**
   - **i.** **if [! -r "$l"] then echo "File $l is not readable-skipping."; fi**
   - **ii.** **if ["$X" -nt "/etc/passwd"]; then**
     **echo "X is a file which is newer than/etc/passwd"**
     **fi**

➔

a) if [!-r"$l"] then echo "File $l is not readable-skipping."; fi

**Error:-**




**Correct code:**

```
read l
if [! -r "$l"]
then
echo "File $l is not readable-skipping."
fi
```

**b)** **if [“$X” -nt “/etc/passwd”]; then**
**echo “X is a file which is newer than/etc/passwd”**
**fi**

**ans:-**

**Error:-**

```
osboxes@osboxes:~$ ./sam.sh
./sam.sh: line 5: syntax error: unexpected end of file
```

**Correct code:-**

```
read x
if [$X -nt “/etc/passwd”]; then
echo “$X is a file which is newer than/etc/passwd”
fi
```

```
osboxes@osboxes:~$ ./sam.sh
ex.txt
ex.txt is file which is newer than /etc/passwd
```

## CONCLUSION:

We successfully executed shell script by using if statement (Single decision, Double decision, Multiple if conditions)

| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Prof. Natasha Brahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 11 |
| Title of Experiment | Execute shell script by using for statement |

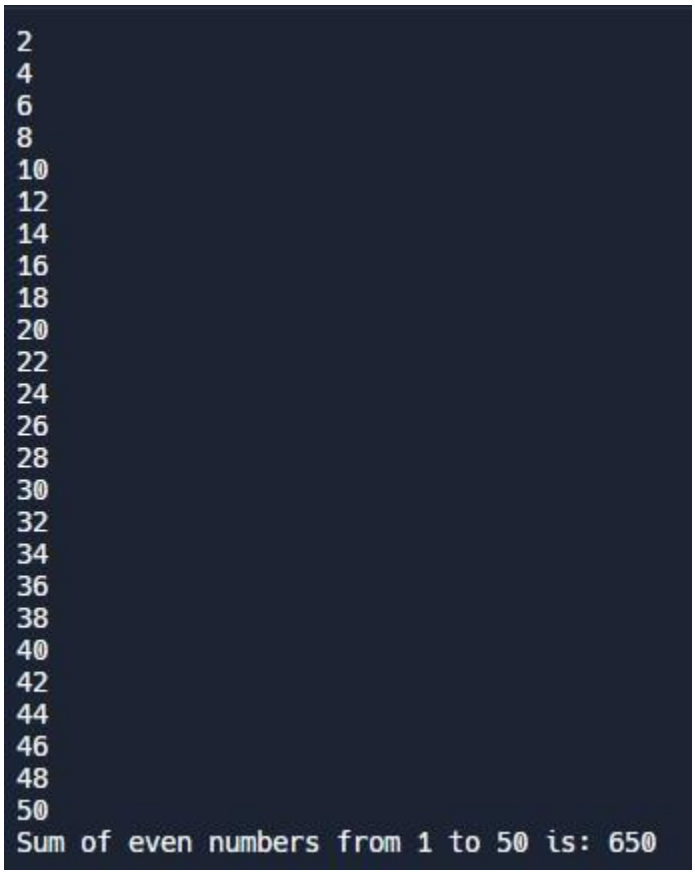
**SET QUESTIONS**

1) **Write a shell script to print even number from 1 to 50 and sum of them**

➔

```
#!/bin/bash
sum=0
for ((i=1; i<=50; i++))
do
  if ((i % 2 == 0));
  then
    echo $i  # Print the even number
    sum=$((sum + i))
  fi
done
echo "Sum of even numbers from 1 to 50 is: $sum"
```

**Output:-**


2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
Sum of even numbers from 1 to 50 is: 650

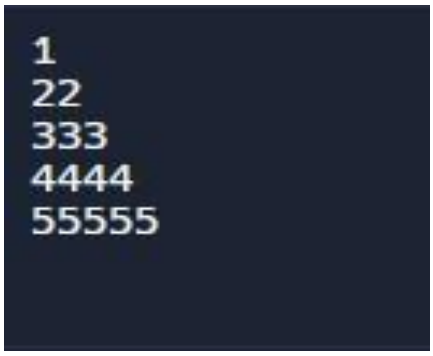
2) **Write shell script to print following output**

**1**
**22**
**333**
**4444**
**5555**

➔

```
#!/bin/bash
for ((i=1; i<=5; i++))
do
  for ((j=1; j<=i; j++))
  do
    echo -n "$i"
  done
  echo
done
```

**Output:-**



## XII. PRACTRICAL RELATED QUESTIONS.

**1) Give output of the following.**

➔

**a)**

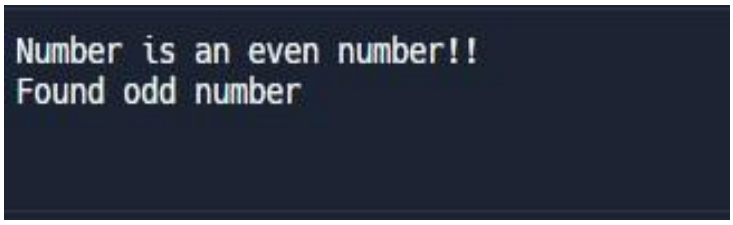
```
#!/bin/sh
NUMBERS="123456 7"
for NUM in $NUMS
do
Q='expr $NUM % 2'
if [ $Q -eq 0]
then
echo "Number is an even number!!"
continue
fi
echo "Found odd number"
done
```

**Ans:-**

```
#!/bin/sh
NUMBERS="123456 7"
for NUM in $NUMBERS
do
Q=$(expr $NUM % 2)
if [ $Q -eq 0 ]
then
 echo "Number is an even number!!"
continue
fi
```

```
echo "Found odd number"

done
```

**Output:-**



**b)**

```
#!/bin/sh
a=0
while ($a-lt 10]
do
echo $a
if [ $a -eq 5]
then
break
fi
a='expr $a + 1'
done
```

**Ans:-**

```
#!/bin/sh
a=0
while [ $a -lt 10 ]
do
 echo $a
if [ $a -eq 5 ]
 then
 break
fi
 a=$(expr $a + 1)
done
```

**Output:-**


**2) State the difference between iteration and recursion.**

➔

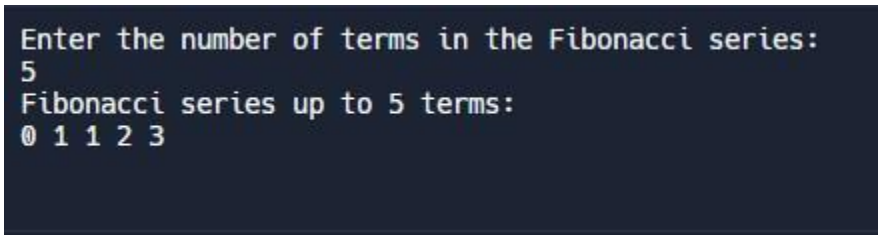
| Property | Recursion | Iteration |
|---|---|---|
| **Definition** | Function calls itself. | A set of instructions repeatedly executed. |
| **Application** | For functions | For loops |
| **Termination** | Through base case, where there will be no function call | When the termination condition for the iterator ceases to be satisfied |
| **Usage** | Used when code size needs to be small and time complexity is not an issue | Used when time complexity needs to be balanced against an expanded code size |
| **Code size** | Smaller code size | Larger code size |
| **Time Complexity** | Very high (generally exponential) time complexity | Relatively lower time complexity (generally polynomial-logarithmic) |

**3) Write a shell script to display Fibonacci series for n numbers.**

➔

```
#!/bin/bash
echo "Enter the number of terms in the Fibonacci series:"
read n
a=0
b=1
echo "Fibonacci series up to $n terms:"
for ((i=0; i<n; i++))
do
  echo -n "$a "
  fn=$((a + b))
  a=$b
  b=$fn
done
echo
```

**Output:-**



**4) Write a shell script to display tables of 2 to 10 numbers (Like 2 * 1 = 2).**

➔

```
#!/bin/bash
for ((i=2; i<=10; i++))
do
  echo "Multiplication table for $i:"
  for ((j=1; j<=10; j++))
  do
    result=$((i * j))
    echo "$i * $j = $result"
  done
  echo
done
```

**Output:-**

```
Multiplication table for 2:
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

Multiplication table for 3:
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

```
Multiplication table for 4:
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

Multiplication table for 5:
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

```
Multiplication table for 6:
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60

Multiplication table for 7:
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

```
Multiplication table for 8:
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80

Multiplication table for 9:
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```

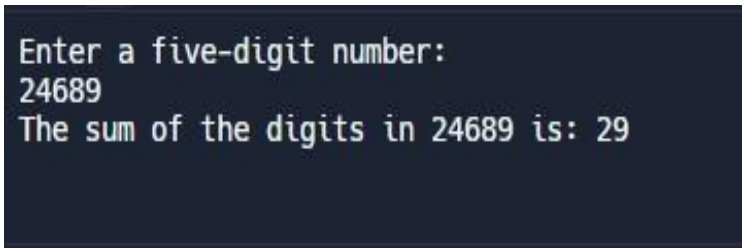
```
Multiplication table for 9:
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

Multiplication table for 10:
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

5) **Write a shell script to accept five-digit number and perform addition of all digits.**

➔

```
#!/bin/bash
echo "Enter a five-digit number:"
read number
if [[ ! $number =~ ^[0-9]{5}$ ]];
then
  echo "Error: Please enter a valid five-digit number."
  exit 1
fi

sum=0
for (( i=0; i<5; i++ ))
do
  digit=${number:i:1}
  sum=$((sum + digit))
done
echo "The sum of the digits in $number is: $sum"
```

**Output:-**



## XIII. EXERCISE.

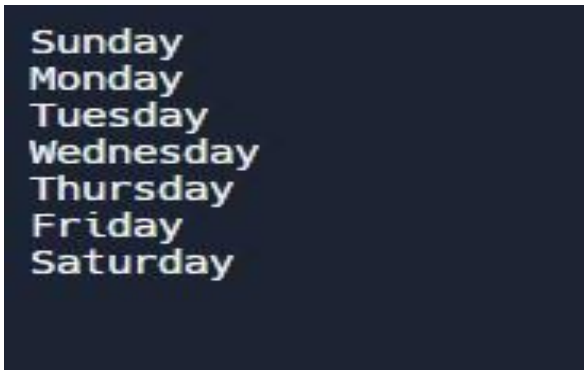
1) **Execute the script for the following.**

   i. **The for loop using days of week.**

➔

```
#!/bin/bash
days=("Sunday" "Monday" "Tuesday" "Wednesday" "Thursday" "Friday"
"Saturday")
for day in "${days[@]}"
do
  echo "$day"
done
```

**Output:-**



**ii. The while loop to print different * patterns.**

➔

```
#!/bin/bash
rows=5
i=1
while [ $i -le $rows ]
do
  for ((j=1; j<=i; j++))
  do
    echo -n "* "
  done
  echo
  i=$((i + 1))
done
```

Output:-

**iii. The case statement for performing various mathematical operations**

➔

```
#!/bin/bash
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
echo "Choose an operation (+, -, *, /):"
read operator
case $operator in
  +)
    result=$((num1 + num2))
    echo "$num1 + $num2 = $result"
    ;;
  -)
    result=$((num1 - num2))
    echo "$num1 - $num2 = $result"
    ;;
  \*)
    result=$((num1 * num2))
    echo "$num1 * $num2 = $result"
    ;;
  /)
    if [ $num2 -ne 0 ]; then
      result=$((num1 / num2))
      echo "$num1 / $num2 = $result"
    else
      echo "Error: Division by zero is not allowed!"
    fi
    ;;
  *)
    echo "Invalid operator!"
    ;;
esac
```

**Output:-**

```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
+
100 + 50 = 150
```

```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
-
100 - 50 = 50
```

```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
*
100 * 50 = 5000
```

```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
/
100 / 50 = 2
```

```
Enter first number:
100
Enter second number:
0
Choose an operation (+, -, *, /):
/
Error: Division by zero is not allowed!
```

## CONCLUSION:

We are able to do different programs using shell script and vi editor and also we successfully implemented shell script.

| | |
|---|---|
| VP Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Prof. Natasha Brahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 12 |
| Title of Experiment | Write a shell script to find out whether-given file exists? |

## XII. PRACTRICAL RELATED QUESTIONS.

1) **What is the command to run the script.**
   **Answer: -** Run the script using ./<filename>

2) **What are file test options with meaning?**
   **Answer: -** The different file test operators are listed below.
   - a: True if the file exists.
   - b: True if it is block special.
   - c: True if it is a character special file.
   - d: True if file exists and is a directory.
   - e: True if the file exists.
   - f: True if the file exists and is a regular file.
   - g: True if the file exists and its SGID bits is set.
   - h: True if the file exists and is a symbolic link.

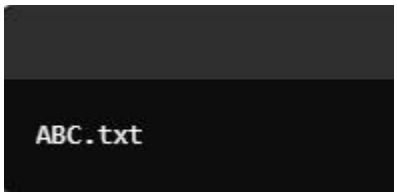
3) **What will be the output of this command (file=ABC.txt)**
   a) **if [ ! -f “$file”];then**
      **echo “$file”**
      **fi**
   b) **test -f “$file” || echo “$file”**
   c) **[ -f “$file”] || echo “$file”**

**Answer: -**

   a) **if [ ! -f “$file”];then**
      **echo “$file”**
      **fi**

**Ans:-**

If ABC.txt does not exist, this command will output:

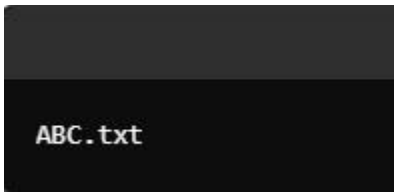



If ABC.txt exists, there will be **no output**.

   b) **test -f “$file” || echo “$file”**

**Ans:-**

If ABC.txt does not exist, this command will output:

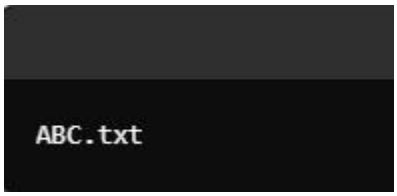



If ABC.txt exists, there will be **no output**.

**c) [ -f “$file”] || echo “$file”**

**Ans:-**

If ABC.txt does not exist, this command will output:




If ABC.txt exists, there will be **no output**.

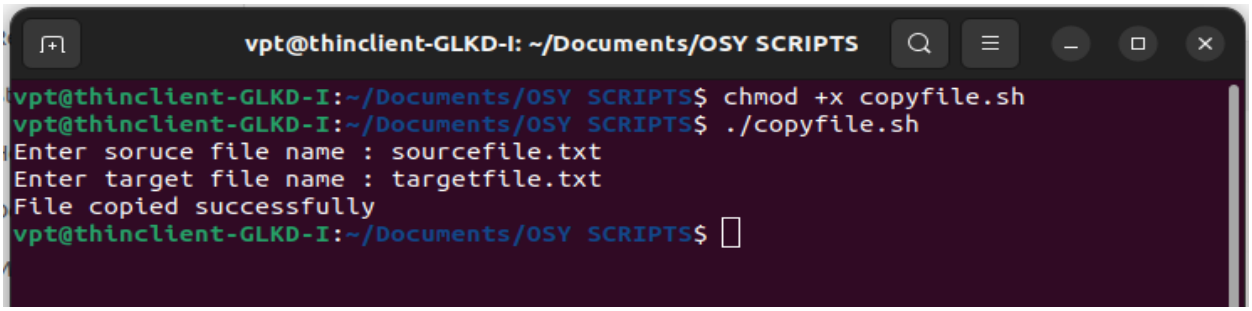
## XIII. EXERCISE.

1) **Write a shell script to copy source file into destination file.**

**Answer: -**

```
#!/bin/bash
echo -n "Enter soruce file name : "
read src
echo -n "Enter target file name : "
read targ

if [ ! -f $src ]
then
 echo "File $src does not exists"
 exit 1
elif [ -f $targ ]
then
 echo "File $targ exist, cannot overwrite"
 exit 2
fi

cp $src $targ

status=$?
if [ $status -eq 0 ]
then
 echo 'File copied successfully'
else
 echo 'Problem copying file'
fi
```

**Output:-**

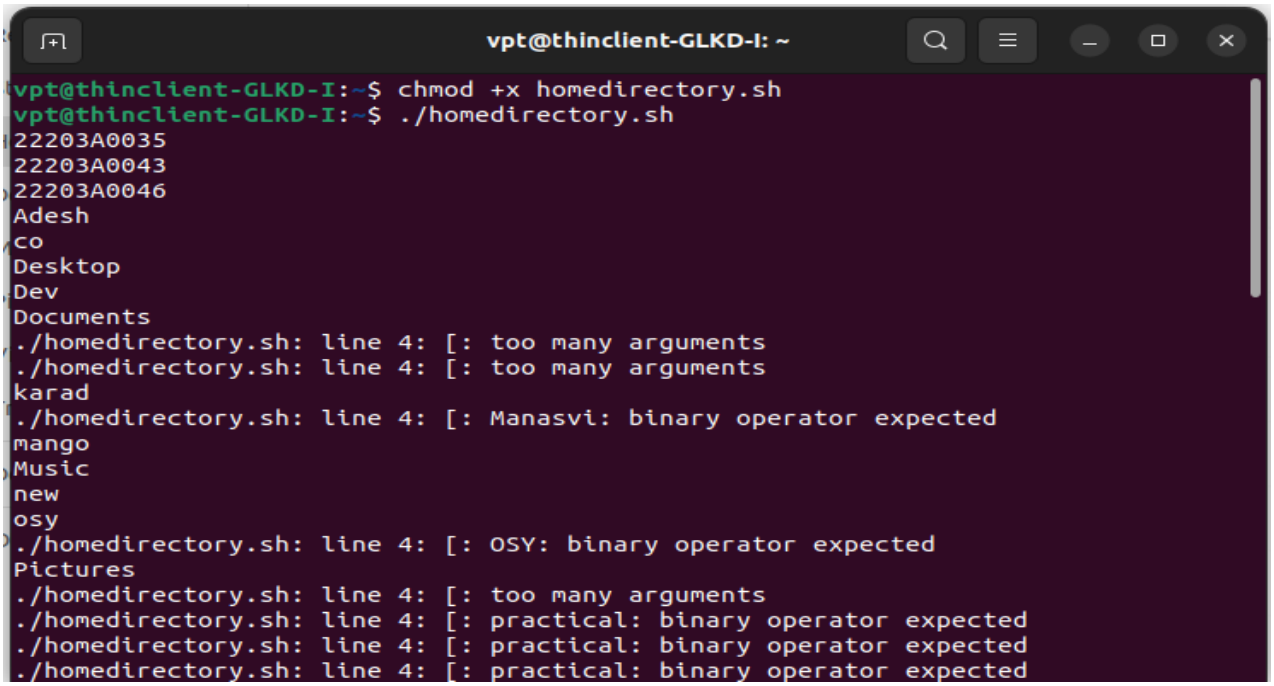



2) **Write a shell script which displays list of all directories in your home directory?**

**Answer: -**

```
#!/bin/bash
for file in *
do
        if [ -d $file ]; then
                echo "$file"
        fi
done
```

**Output:-**

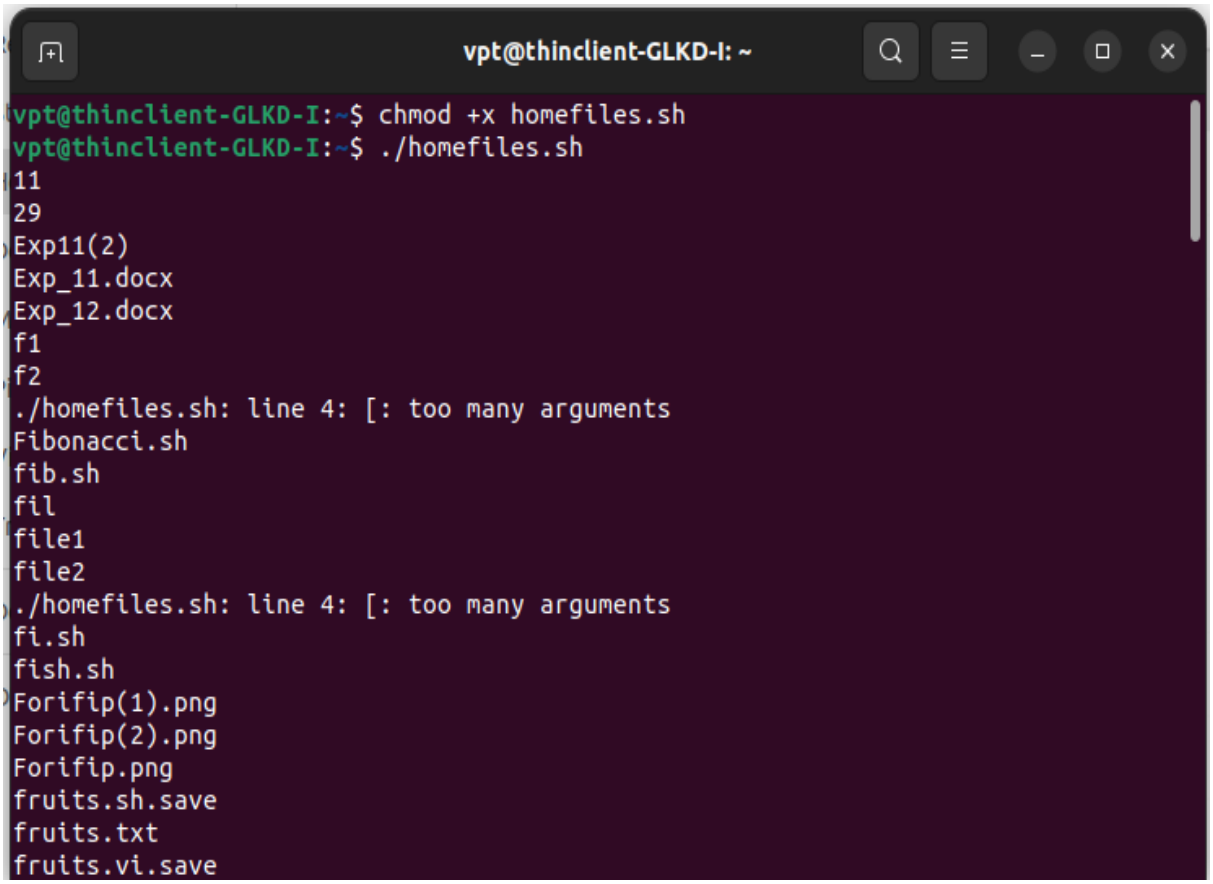


3) **Write a shell script which displays list of all files in your home directory?**

**Answer: -**

```
#!/bin/bash
for file in *
do
        if [ -f $file ]; then
                echo "$file"
        fi
done
```

**Output:-**



4) **Write a file handling program. First check whether it is file or directory, then if it is file the program should ask user for choices of copying, removing and renaming files. Use case statement.**

**Answer: -**

```
echo "Enter the file name"
read file1
if [ -f $file1 ]; then
        echo "It is a file"
elif [ -d $file1 ]; then
        echo "It is a Directory!";
else
        echo "File or Directory does not exist!";
        exit 1
fi

echo -e "1) Copy"
#!/bin/bash
echo -e "2) Remove"
echo -e "3) Rename"
read ch
case $ch in
1)
        echo -n "Enter target file name : "
        read targ
        if [ -f $targ ]
        then
         echo "File $targ exist, cannot overwrite"
         exit 2
fi

        cp $file1 $targ

        status=$?

        if [ $status -eq 0 ]
        then
         echo 'File copied successfully'
        else
         echo 'Problem copying file'
        fi;;
```

2)

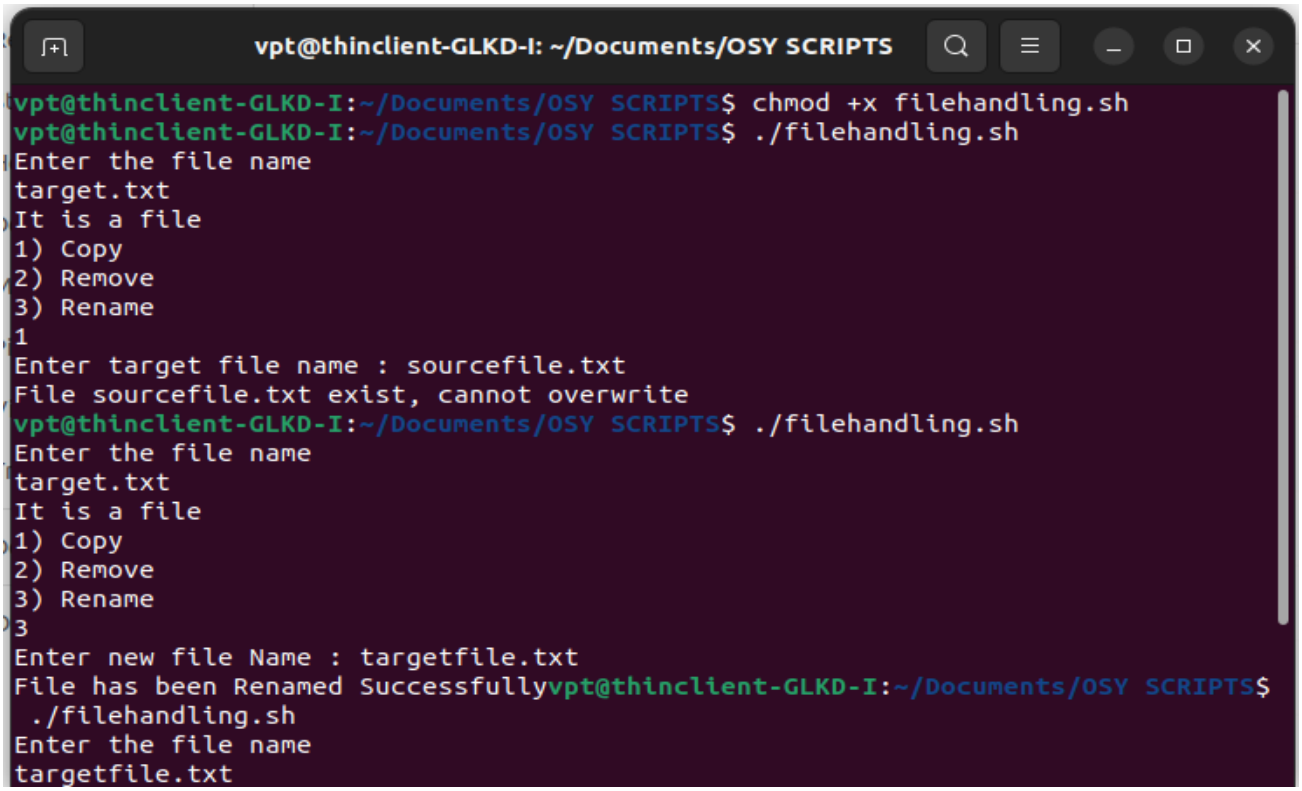
```
        rm -i $file1;;
```

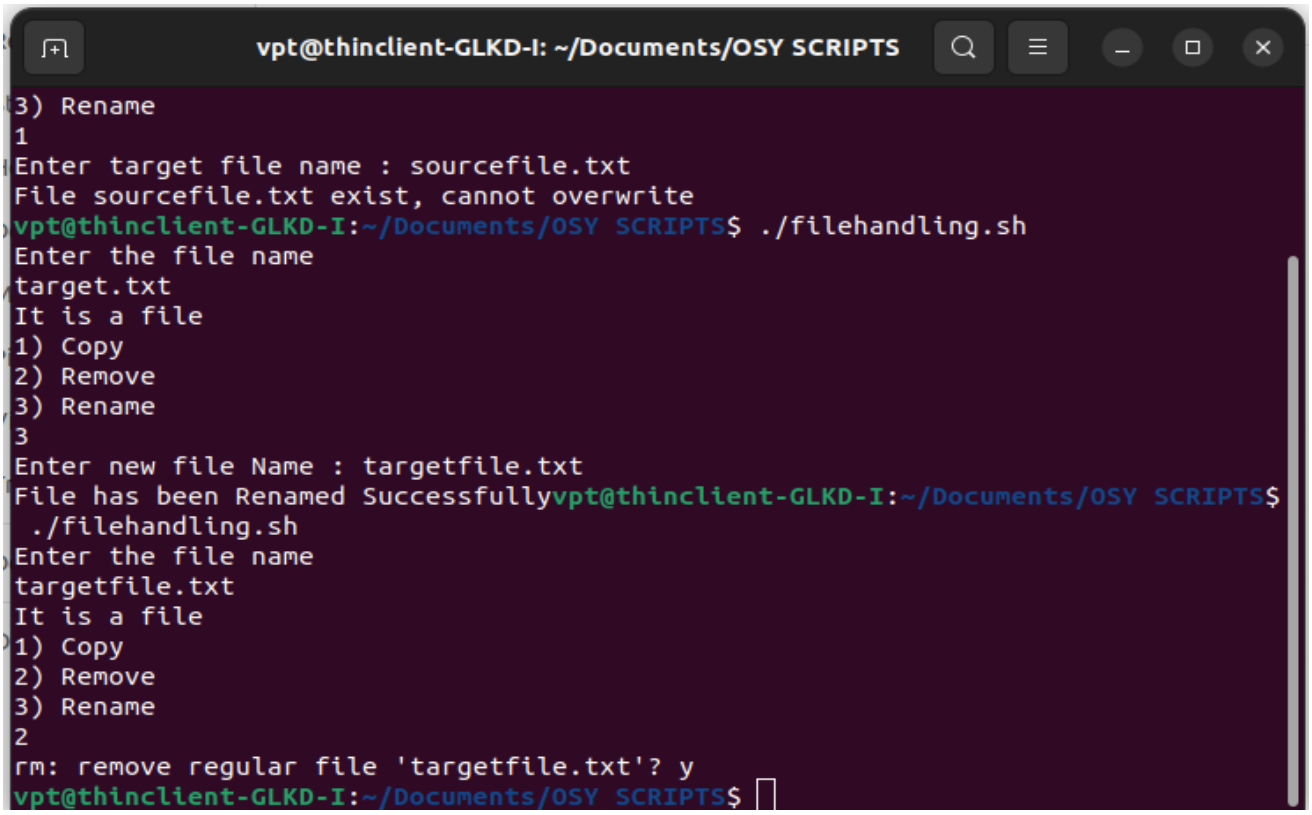
3)

```
        echo -n "Enter new file Name : "
        read file2
        mv $file1 $file2
        echo -n "File has been Renamed Successfully";;
```

esac

**Output:-**



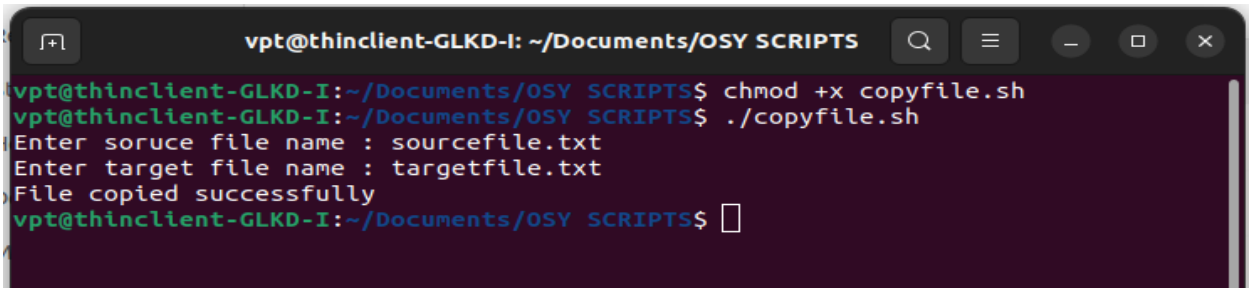

vpt@thinclient-GLKD-I: ~/Documents/OSY SCRIPTS
3) Rename
1
Enter target file name : sourcefile.txt
File sourcefile.txt exist, cannot overwrite
vpt@thinclient-GLKD-I:~/Documents/OSY SCRIPTS$ ./filehandling.sh
Enter the file name
target.txt
It is a file
1) Copy
2) Remove
3) Rename
3
Enter new file Name : targetfile.txt
File has been Renamed Successfullyvpt@thinclient-GLKD-I:~/Documents/OSY SCRIPTS$
./filehandling.sh
Enter the file name
targetfile.txt
It is a file
1) Copy
2) Remove
3) Rename
2
rm: remove regular file 'targetfile.txt'? y
vpt@thinclient-GLKD-I:~/Documents/OSY SCRIPTS$

5) **Write a script to copy source file into destination file.**

**Answer: -**

```
#!/bin/bash
echo -n "Enter soruce file name : "
read src
echo -n "Enter target file name : "
read targ

if [ ! -f $src ]
then
 echo "File $src does not exists"
 exit 1
elif [ -f $targ ]
then
 echo "File $targ exist, cannot overwrite"
 exit 2
fi
cp $src $targ

status=$?
if [ $status -eq 0 ]
then
 echo 'File copied successfully'
else
 echo 'Problem copying file'
fi
```

**Output:-**




## CONCLUSION:

We have successfully completed shell script to find out whether given file exists. In this practical we have studied checking if a file exists. The commonly used file operators are -e and -f.

| | |
|---|---|
| VP Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: OSY | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No: V118 | Name of Subject Teacher: Natasha Brahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 13 |
| Title of Experiment: | Write shell script to check and grant file permission |

- **Practical Related Question**

**1. What are permissions of a file?**

**Ans:** File permissions refer to the control of actions that users are allowed to perform on a file, such as read, write, and execute.

**2. How to assign permission to a file?**

**Ans:**

1. File permissions refer to the control of actions that users are allowed to perform on a file, such as read, write, and execute.

2. Use the ` chmod` command followed by the permission settings and the file name

**3. What happens when exception is thrown by main method?**

**Ans:** The main method should simply terminate if any exception occurs. The throws clause only states that the method throws a checked FileNotFoundException and the calling method should catch or rethrow it.

**4. How to check permissions of all files and directories?**

**Ans:** To check the permissions of all files and directories in a specific directory on a Unix-like system (Linux, macOS), you can use the `ls` command with the `-l`

**5.What are the test commands to check the permission of a file?**

**Ans:** The `test` command (or its synonym `[ ]`) allows you to check specific permissions

- **Program Code:**

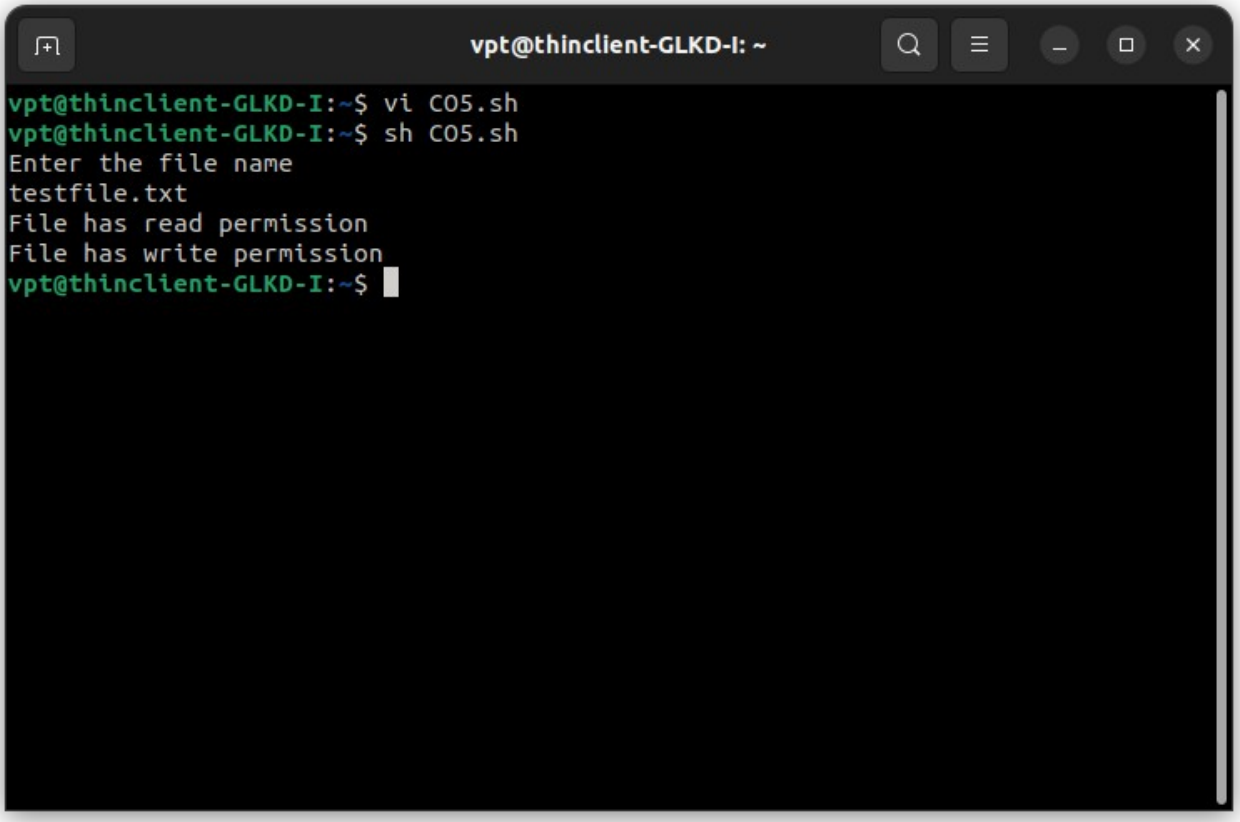
**1) Write a shell script to find out whether file has read write and execute permission.**

**Ans:**

**a] Code:**

```
echo "Enter the file name"
read file1

if [ -r $file1 ]
then
echo "File has read permission"
fi

if [ -w $file1 ]
then
echo "File has write permission"
fi

if [ -x $file1 ]
then
echo "File has Execute permission"
fi
```

**b] Output:**



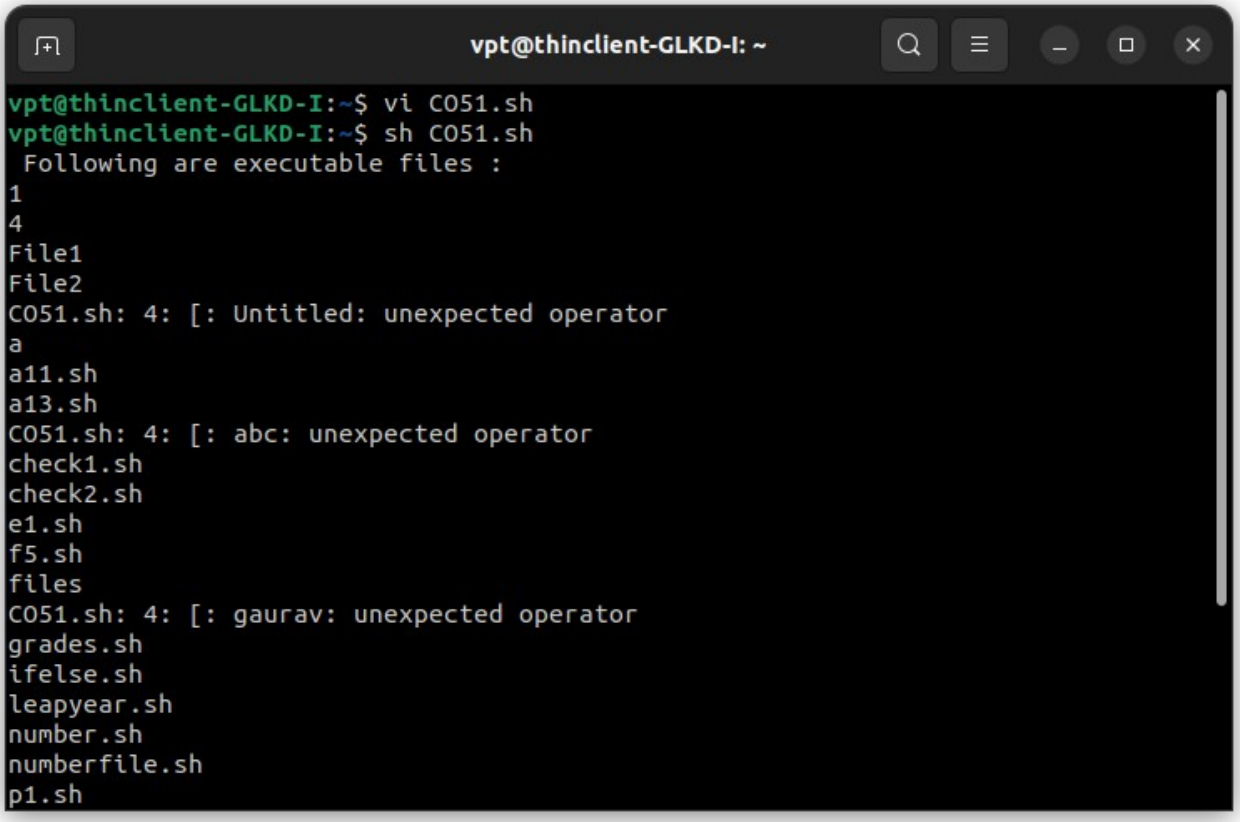

**2) Write a shell script which displays the list of all executable files in the current working directory.**

**Ans:**

**a] Code:**

```
echo " Following are executable files : "
for file1 in *
do
if [ -x $file1 ] && [ ! -d $file1 ]
then
echo $file1
fi
done
```

**b] Output:**



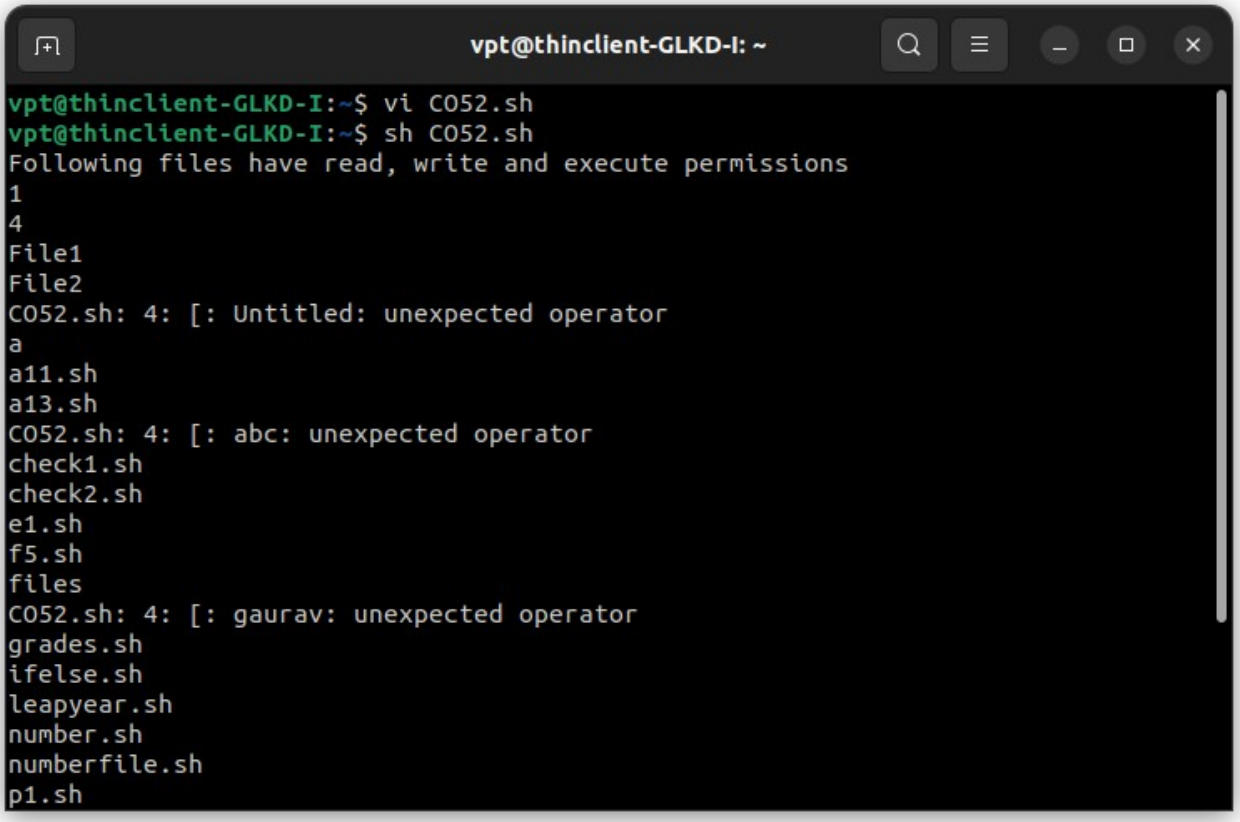

**3) Write a shell script which displays a list of all the files in the current directory to which has read, write, and execute permissions.**

**Ans:**

**a] Code:**

```
echo "Following files have read, write and execute permissions "
for file1 in *
do
if [ -r $file1 ] && [ -w $file1 ] && [ -x $file1 ] && [ ! -d $file1 ];
then
echo "$file1"
fi
done
```

**b] Output:**




**4) Write a shell script which accepts a filename and assigns it all the permissions.**

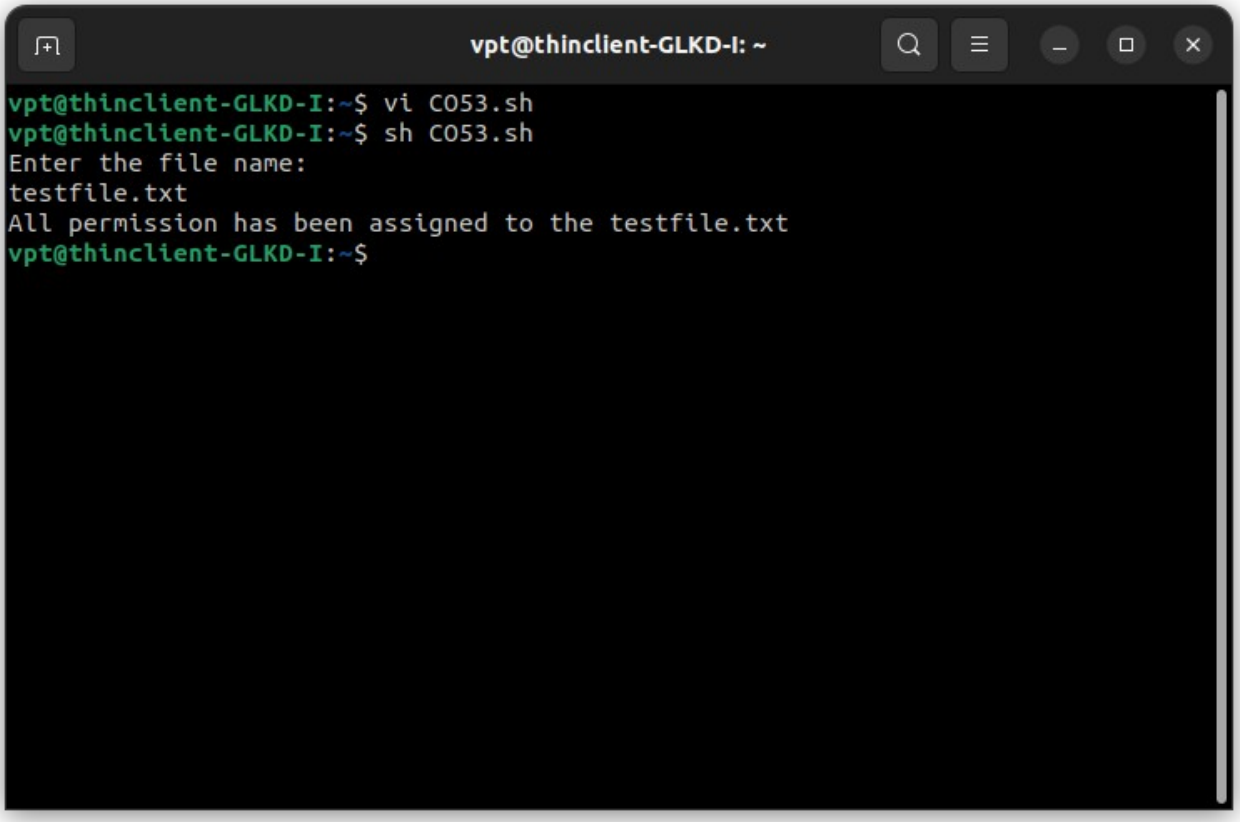
**Ans:**

**a] Code:**

```
echo "Enter the file name: "
read file1

if [ -d $file1 ]
then
echo "$file1 is a directory"
exit 1
elif [ ! -f $file1 ]
then
echo "$file1 does not exits "
exit 2
```

```
else

chmod 777 $file1
echo "All permission has been assigned to the $file1 "
fi
```

**b] Output:**



| | |
|---|---|
| Vidyalankar Polytechnic | DEPARTMENT OF COMPUTER ENGINEERING |

| | |
|---|---|
| Subject: Operating System | Subject Code:22516 |
| Semester:5th Semester | Course: Computer Engineering |
| Laboratory No:V118 | Name of Subject Teacher: Natasha Brahme |
| Name of Student: Siddharth Shah | Roll Id: 22203A0041 |

| | |
|---|---|
| Experiment No: | 14 |
| Title of Experiment | Implement scheduling alogrithms. |

- **Program Code :**

1. **Consider the processes P1,P2,P3,P4 given in the below table, arrives for execution in the same order, with arrival time 0 and given burst time,find using the fcfs scheduling algorithm.**
   **1.Turn around time for each process**
   **2.Waiting time for each process**
   **3.Average turn around time**
   **4.average waiting time**

| Process | Burst Time |
|---|---|
| P1 | 21 |
| P2 | 6 |
| P3 | 3 |
| P4 | 2 |

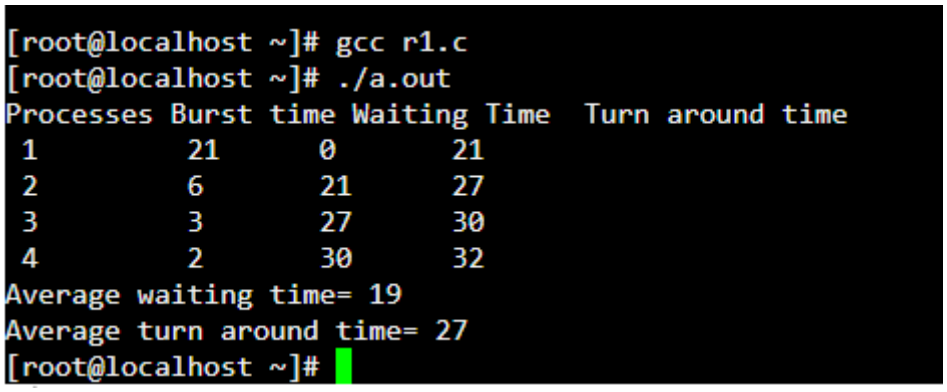
**Ans :**

Code :

```
#include <stdio.h>
void findWaitingTime(int processes[],int n,int bt[],int wt[]){
wt[0] = 0;
for(int i=1;i<n;i++){
wt[i] = bt[i-1]+wt[i-1];

}
}

void findTurnAroundTime(int processes[],int n,int bt[],int wt[],int tat[]){
for(int i=0;i<n;i++){
tat[i]=bt[i]+wt[i];

}
}


void findavgTime(int processes[],int n,int bt[]){
int wt[n],tat[n],total_wt=0,total_tat=0;
findWaitingTime(processes,n,bt,wt);
findTurnAroundTime(processes,n,bt,wt,tat);
printf("Processes Burst time Waiting Time  Turn around time\n");
for(int i=0;i<n;i++){
total_wt=total_wt+wt[i];
total_tat = total_tat+tat[i];
printf(" %d",(i+1));
printf("\t   %d",bt[i]);
printf("\t   %d",wt[i]);
printf("\t   %d\n",tat[i]);
}
int s=(float)total_wt/(float)n;
int t=(float)total_tat/(float)n;
printf("Average waiting time= %d",s);
printf("\n");
printf("Average turn around time= %d\n",t);
```

```
}

int main(){
int processes[]={1,2,3,4};
int n=sizeof processes/sizeof processes[0];
int burst_time[]={21,6,3,2};
findavgTime(processes,n,burst_time);
return 0;



}
```

**Output :**



- **Practical related questions :**

1. **Compare SJF,Priority and RR with respect to turnaround time and average waiting time.**

**Ans :**

| **Priority Scheduling algorithm** | **RR Algorithm** | **SJF algorithm** |
|---|---|---|
| Generally minimizes the turnaround time because it processes the shortest jobs first. | Turnaround time can be higher in RR because every process gets a fair share of CPU time, but it might have to wait multiple cycles before getting enough time to complete. | Turnaround time depends heavily on the priority of the jobs. |
| SJF often results in the minimum average waiting time since shorter jobs do not have to wait behind longer jobs. | RR aims for fairness but typically has a higher average waiting time compared to SJF. | Can be efficient if high-priority tasks are important, but low-priority processes may experience longer waiting times. |

**2. State the conditions for preemptive and non-preemptive scheduling algorithm.**

**Ans :**

**Conditions for Pre-emptive scheduling algorithm :**

1. Higher Priority Arrival: If a process with a higher priority than the currently running process arrives, it can preempt the current process.
2. Arrival of a Shorter Job: In algorithms like Shortest Remaining Time First (SRTF), a newly arrived process with a shorter remaining execution time than the current process can preempt the running process.
3. Time Quantum Expiration: In Round Robin (RR), the currently running process is preempted when its allocated time quantum expires, and the CPU is given to the next process in the queue.
4. System Calls or Interrupts: Certain interrupts or system calls (e.g., I/O operations) may cause the currently running process to be preempted.

**Conditions for non-preemptive scheduling algorithm :**

1. Completion of the Process: A running process will not be interrupted until it has completed its execution.
2. Voluntary Release: If a process voluntarily enters a waiting state (e.g., to perform I/O operations), the CPU is then given to another ready process.
3. Process Termination: The CPU is only available to other processes after the currently running process terminates.

**3. Give the reason of problems arises in FCFS.**

**Ans :**

1. **Long Waiting Time for Short Processes (Convoy Effect)**:

   In FCFS, if a long process arrives before shorter ones, the shorter processes must wait until the long one finishes. This leads to high average waiting times, especially when there's a mix of short and long processes.

2. **No Prioritization**:

   FCFS does not consider the priority of tasks. All processes are treated equally, regardless of their urgency or importance. This means that

critical tasks might have to wait behind less important ones, which is not always ideal for time-sensitive applications.

3. **Poor Response Time**:

   The response time in FCFS can be poor for processes that arrive after others, as they have to wait in line even if the earlier processes are long-running.

4. **Not Suitable for Time-Sharing Systems**:

   FCFS is not well-suited for time-sharing systems where the goal is to ensure that each process gets a fair share of the CPU. Since FCFS runs processes until completion before moving to the next one, it can lead to one process monopolizing the CPU.

**4. Write a formula for turnaround Time**

**Ans :**

Waiting time = Starting time – Arrival time

**5. Write a formula for average waiting time.**

**Ans :**

Turnaround time = Ending time – Arrival time

- **Exercise**

1. The jobs are scheduled for execution as follows :
   Solve the problem by using FCFS and pre-emptive SJF
   Find average waiting time using gantt chart

| Process | Arrival time | Burst time |
|---|---|---|
| P1 | 0 | 10 |
| P2 | 1 | 4 |
| P3 | 2 | 14 |
| P4 | 3 | 8 |

Draw gantt chart for above mention example.

Ans :

Gantt chart for FCFS scheduling algorithm :

| P1 | P2 | P3 | P4 |
|---|---|---|---|

Gantt chart for Pre-emptive SJF scheduling algorithm :

| P1 | P2 | P4 | P1 | P3 |
|---|---|---|---|---|

FCFS scheduling algorithm :

Waiting time of P1 = 0

Waiting time of P2 = 9

Waiting time of P3 = 12

Waiting time of P4= 25

Average waiting time = (0+9+12+25)/4 = 11.5 msec.

SJF pre-emptive scheduling algorithm :

Waiting time of P1 = 11

Waiting time of P2 = 0

Waiting time of P3 = 20

Waiting time of P4= 2

Average waiting time = (11+0+20+2)/4 =8.25 msec.

2. Calculate average waiting time using RR algorithm for the following set of processes with the length of the CPU burst time given in millisecond. (Time quantum 20 ms)

| Process | Burst Time |
|---|---|
| P1 | 12 |
| P2 | 45 |
| P3 | 78 |
| P4 | 90 |

Ans :

Waiting time of P1 = 0 ms

Waiting time of P2 = 52 ms

Waiting time of P3 = 105 ms

Waiting time of P4 = 123 ms

Average waiting time = (0+52+105+123)/4 = 70 ms

Vidyalankar Polytechnic

DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| **Subject: Operating System** | **Subject Code:22516** |
| **Semester: 5th Semester** | **Course: Computer Engineering** |
| **Laboratory No: V118** | **Name of Subject Teacher: Natasha Brahme** |
| **Name of Student: Siddharth Shah** | **Roll Id: 22203A0041** |

| | |
|---|---|
| **Experiment No:** | **15** |
| **Title of Experiment** | **Write a C program to implement FIFO page replacement Algorithm.** |

## X. Program Code

## 1. Page Replacement Stream :

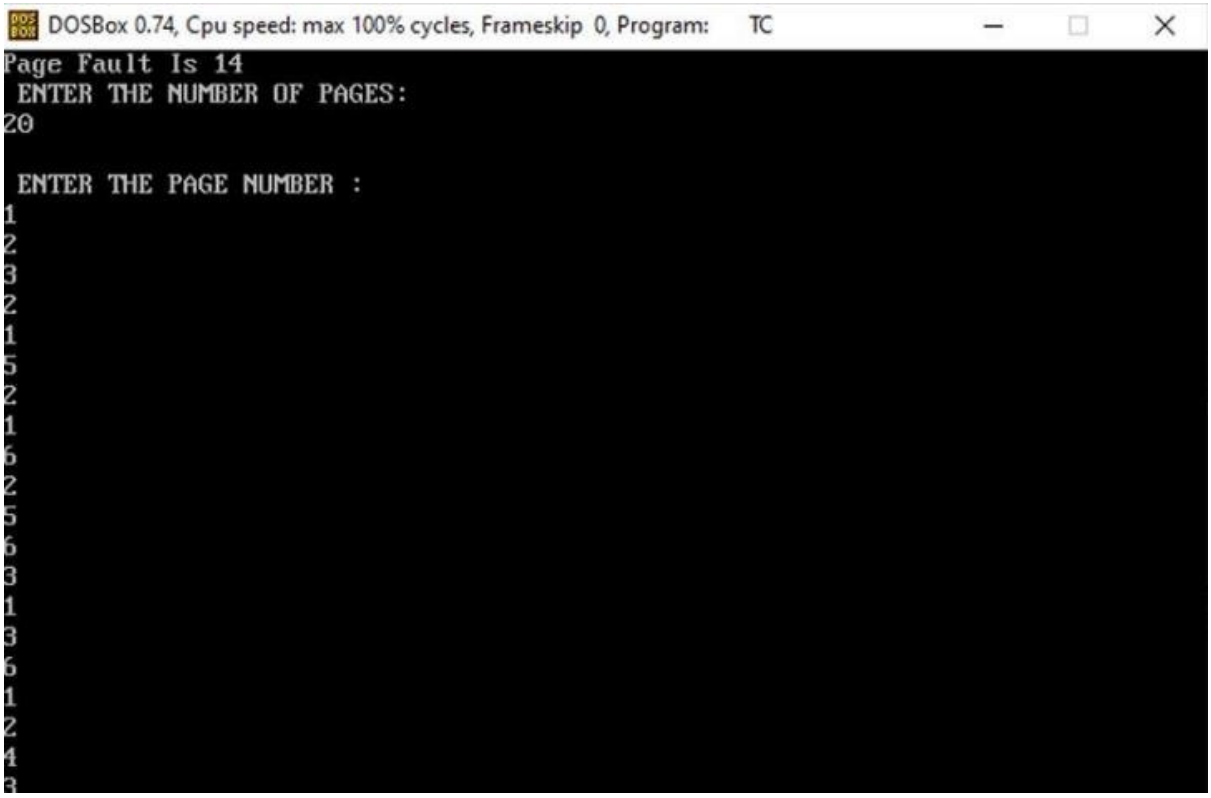
## 12321521625631361243

## Ans:

```
#include <stdio.h> int
main()
{
 int i,j,n,a[50],frame[10],no,k, Flag,count=0; printf("\n
 ENTER THE NUMBER OF PAGES:\n");
 scanf("%d",&n);
 printf("\n ENTER THE PAGE NUMBER :\n");
 for(i=1;i<=n;i++)
 scanf("%d",&a[i]);
 printf("\n ENTER THE NUMBER OF FRAMES :");
 scanf("%d",&no);

 for(i=0;i<=n;i++)
 {
```

```
 printf("%d\t\t",a[i]);
Flag =0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
Flag=1;
 if (Flag==0)
{
frame[j]=a[i];
j=(j+1)%no; count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("Page Fault Is %d",count);
return 0;
getch();
}
```

## XI. Result (Output of code)




## XII. Practical Related Questions

### 1. State the advantages and disadvantages of FIFO

### Ans:

**Advantages:**

FIFO method is easy to understand and operate.

FIFO method is suitable for bulky materials with high unit prices. FIFO method helps to avoid deterioration and obsolescence.5.

Value of closing stock of materials will reflect the current market price.

**Disadvantages :**

FIFO method is improper if many lots are purchased during the period at different prices.

FIFO method overstates profit especially in inflation

## 2. What is page hit ?

If CPU tries to retrieve the needed page from main memory, and that page is existed inthe main memory (RAM), then it is known as PAGE HIT.

## 3. What is page Fault?

A page fault (PF or hard fault) is a type of exception raised by computer hardware when a running program accesses a memory page that is not currently mapped by the memory management unit (MMU) into the virtual address space of a process.

# XIII. Exercise

## 1. Compare FIFO and LRU

| **FIFO** | **LRU** |
|---|---|
| FIFO is First in First out | LRU is Least Recently used |
| Number of page fault is more than LRU | Number of page fault is less than FIFO |
| Simple to implement | Considered to be good |

| **Grade and Dated Signature of Teacher** | **Process Related (35)** | **Product Related (15)** | **Dated Sign** |
|---|---|---|---|
| | | | |

# Practical 15

Code:

```
#include <stdio.h>

int main() {
   int i, j = 0, n, a[50], frame[10], no, k, flag, count = 0;

   printf("\nENTER THE NUMBER OF PAGES:\n");
   scanf("%d", &n);

   printf("\nENTER THE PAGE NUMBERS:\n");
   for (i = 0; i < n; i++) // Start loop from 0 for correct array indexing
      scanf("%d", &a[i]);

   printf("\nENTER THE NUMBER OF FRAMES:\n");
   scanf("%d", &no);

   // Initialize frames
   for (i = 0; i < no; i++)
      frame[i] = -1; // Use -1 to indicate an empty frame

   printf("\nPage Number\tFrames\n");

   for (i = 0; i < n; i++) {
```

```
printf("%d\t\t", a[i]); // Print the current page number
flag = 0; // Reset flag for each page

// Check if page already exists in any frame
for (k = 0; k < no; k++) {
   if (frame[k] == a[i]) {
      flag = 1; // Page hit
      break;
   }
}

// If page fault occurs
if (flag == 0) {
   frame[j] = a[i];       // Replace page in FIFO order
   j = (j + 1) % no;      // Update frame pointer
   count++;               // Increment page fault count
}

// Print the frame status
for (k = 0; k < no; k++) {
   if (frame[k] != -1)
      printf("%d ", frame[k]);
   else
      printf("- "); // Empty frame display
```

```
        }
        printf("\n");
    }

    printf("\nTotal Page Faults: %d\n", count);
    return 0;
}
```