Practical 1

Title: A) INSTALLATION: configuration & customizations of Linux

Solution:

Theory: Linux is an open-source operating system that can be run on a wide range of devices. Linux can be operated without a graphical interface through a text terminal. It is an operating system, like Windows or macOS, that manages your computer's hardware and software. It is free and open-source, meaning that anyone can use and modify it. Linux is known for its stability, security, and performance, and it can run on many different types of devices, from phones to servers.

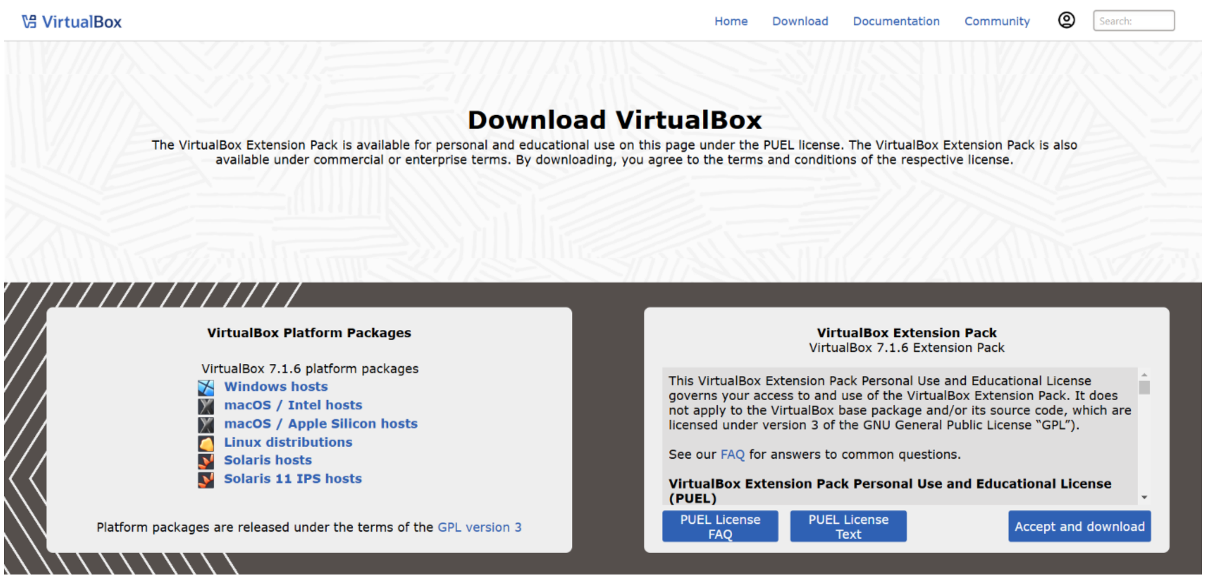
Steps: If we want to experiment with Linux without affecting your main operating system (OS) at all, we can use a virtual machine (VM). We can run Linux directly atop our primary OS, whether it's Mac OS X/macOS or Windows. In this tutorial, we will specifically talk about installing a Linux distribution called Ubuntu inside your VM software. However, the instructions are pretty similar for running other Linux distributions. We'll walk through installing VirtualBox, creating a VM within it, and installing the latest Ubuntu distribution to the VM.

About VirtualBox

VirtualBox is an open-source hypervisor that is developed by Oracle. It is similar to an emulator that creates and runs VMs where we can install different OSs without having to tinker with our disk partitions or worrying about impacting our primary OS. Instead, we run a different OS like Ubuntu as if it were another program running on our computer.

Step 1: Download VirtualBox (VM).
Go to https://www.virtualbox.org/ and download VM for Windows

–



Download VirtualBox

The VirtualBox Extension Pack is available for personal and educational use on this page under the PUEL license. The VirtualBox Extension Pack is also available under commercial or enterprise terms. By downloading, you agree to the terms and conditions of the respective license.

VirtualBox Platform Packages

VirtualBox 7.1.6 platform packages
- Windows hosts
- macOS / Intel hosts
- macOS / Apple Silicon hosts
- Linux distributions
- Solaris hosts
- Solaris 11 IPS hosts

Platform packages are released under the terms of the GPL version 3

VirtualBox Extension Pack
VirtualBox 7.1.6 Extension Pack

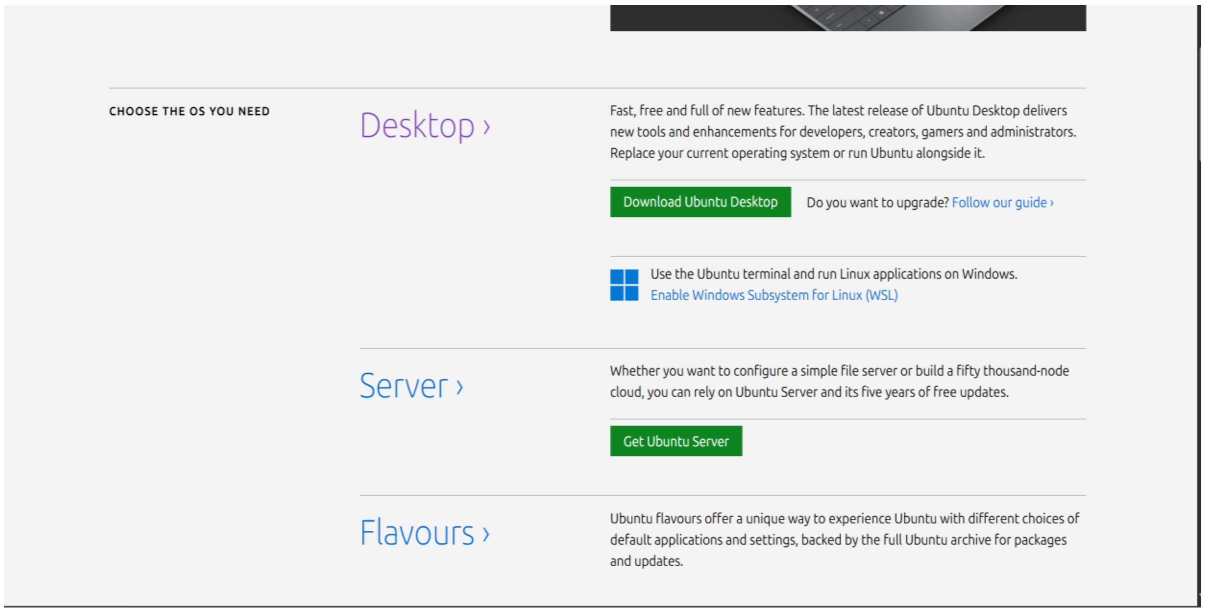
This VirtualBox Extension Pack Personal Use and Educational License governs your access to and use of the VirtualBox Extension Pack. It does not apply to the VirtualBox base package and/or its source code, which are licensed under version 3 of the GNU General Public License "GPL").

See our FAQ for answers to common questions.

VirtualBox Extension Pack Personal Use and Educational License (PUEL)

Step 2: Download Ubuntu Linux ISO File.

Go to https://ubuntu.com/download/desktop and download ISO file for Linux.



CHOOSE THE OS YOU NEED

Desktop ›

Fast, free and full of new features. The latest release of Ubuntu Desktop delivers new tools and enhancements for developers, creators, gamers and administrators. Replace your current operating system or run Ubuntu alongside it.

Use the Ubuntu terminal and run Linux applications on Windows.
Enable Windows Subsystem for Linux (WSL)

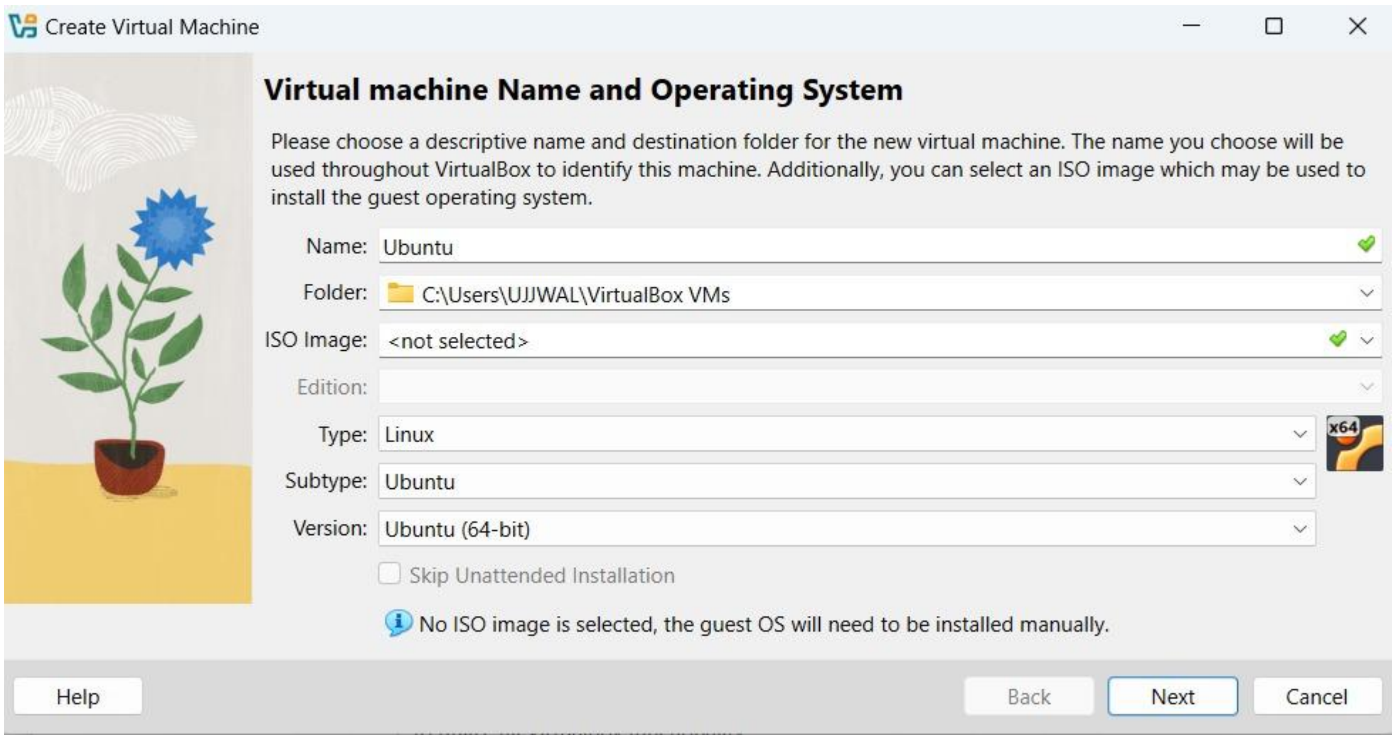
Server ›

Whether you want to configure a simple file server or build a fifty thousand-node cloud, you can rely on Ubuntu Server and its five years of free updates.

Flavours ›

Ubuntu flavours offer a unique way to experience Ubuntu with different choices of default applications and settings, backed by the full Ubuntu archive for packages and updates.

Step 3: Create a new Virtual Machine.



Create Virtual Machine

Virtual machine Name and Operating System

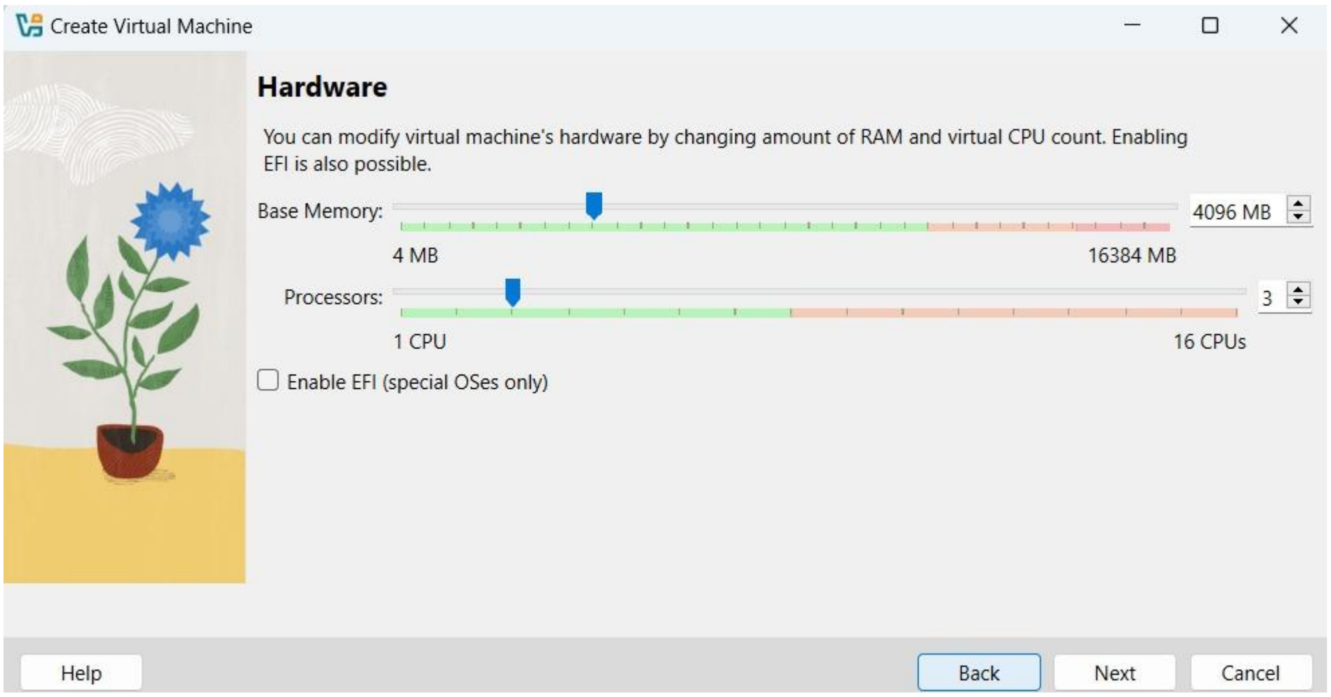
Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.

Name:	Ubuntu
Folder:	C:\Users\UJJWAL\VirtualBox VMs
ISO Image:	<not selected>
Edition:	
Type:	Linux
Subtype:	Ubuntu
Version:	Ubuntu (64-bit)

☐ Skip Unattended Installation

No ISO image is selected, the guest OS will need to be installed manually.

Step 4: Allocate RAM Memory and number of Processors.



Create Virtual Machine

Hardware

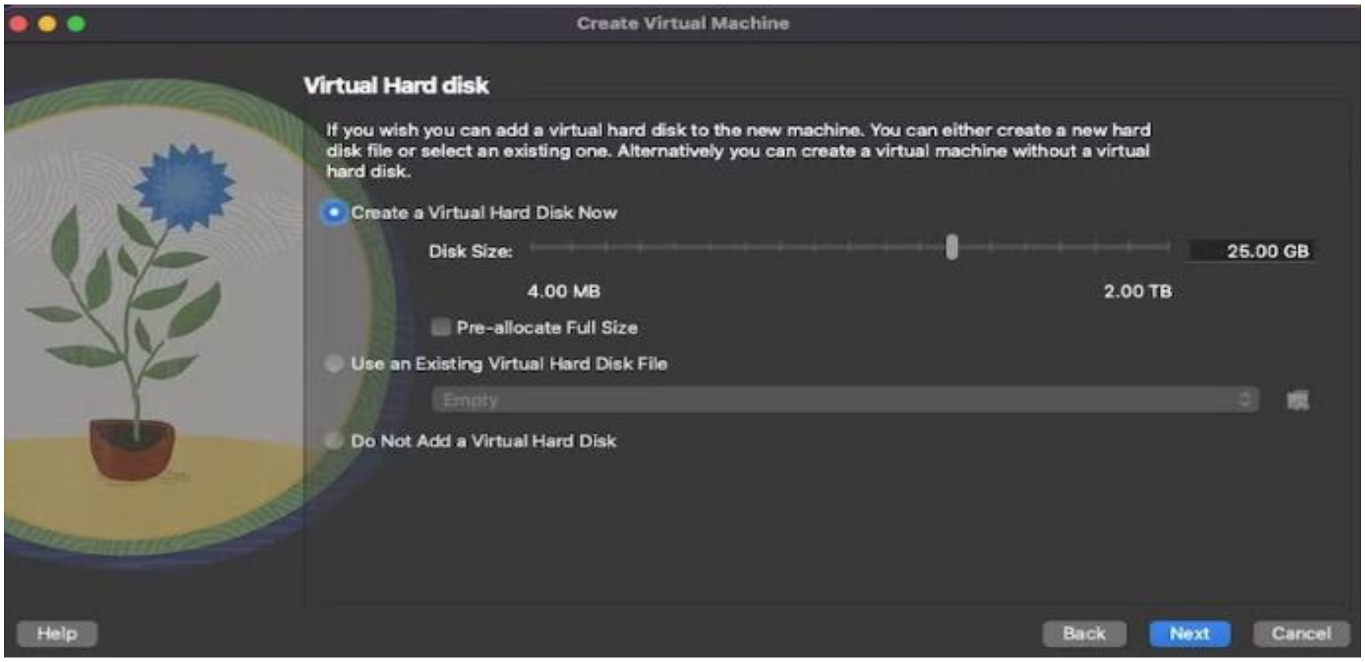
You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

Base Memory: 4096 MB (4 MB – 16384 MB)

Processors: 3 (1 CPU – 16 CPUs)

☐ Enable EFI (special OSes only)

Step 5: Create a Virtual Hard Disk.



Create Virtual Machine

Virtual Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively you can create a virtual machine without a virtual hard disk.

● Create a Virtual Hard Disk Now

Disk Size: 25.00 GB (4.00 MB – 2.00 TB)

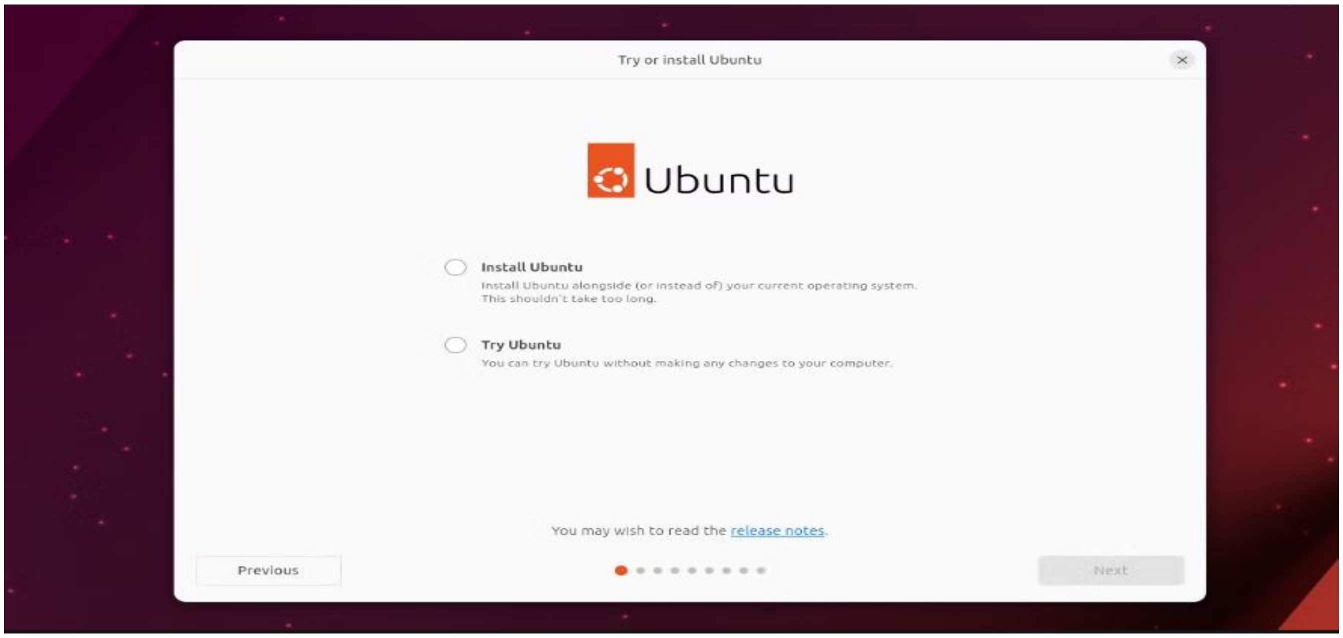
☐ Pre-allocate Full Size

○ Use an Existing Virtual Hard Disk File

Empty

○ Do Not Add a Virtual Hard Disk

Step 6: Install Linux.



Try or install Ubuntu

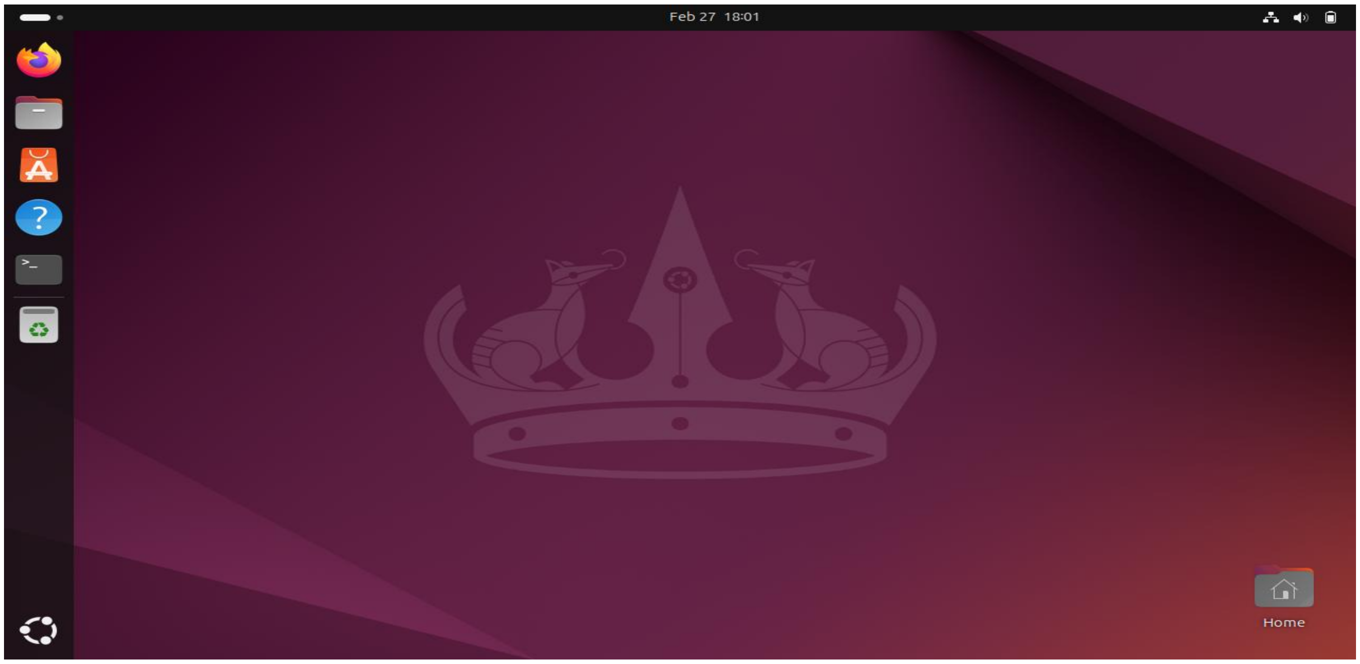
Ubuntu

Install Ubuntu
Install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

Try Ubuntu
You can try Ubuntu without making any changes to your computer.

You may wish to read the release notes.

Linux is now installed on your PC using VirtualBox VM.



Title: B) INTRODUCTION TO GCC COMPILER: basics of GCC, compilation of program, execution of program

Solution: The GCC (GNU Compiler Collection) is a set of compilers that support multiple programming languages. It's one of the most widely used compilers, especially in open-source and Unix-like environments, and is often bundled with Linux distributions.

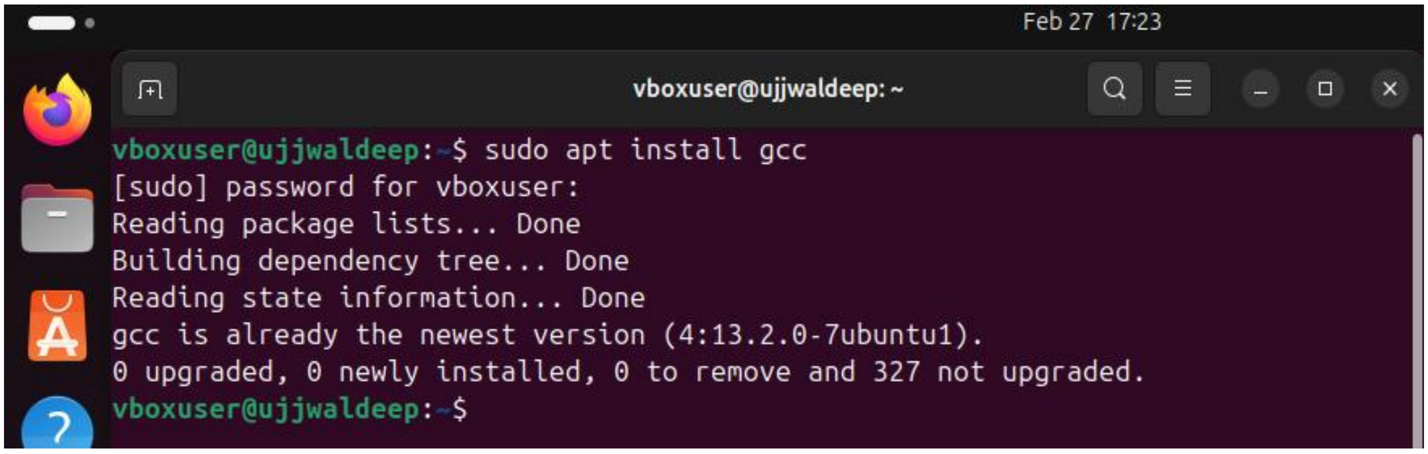
Theory: The GNU Compiler Collection (GCC) is a powerful open-source compiler system developed by the GNU Project. It is widely used for compiling programs written in multiple programming languages such as C, C++, Java. GCC serves as the default compiler for most Linux-based operating systems and is also available for Windows and macOS.

GCC is a collection of compilers that translates **high-level source code** into **machine code** that a computer can execute. It was originally developed by **Richard Stallman** in 1987 as part of the **GNU Project** and has since become a standard compiler for Unix-like systems.

Step 1: Install the GCC Compiler on Linux system with the help of the following command.

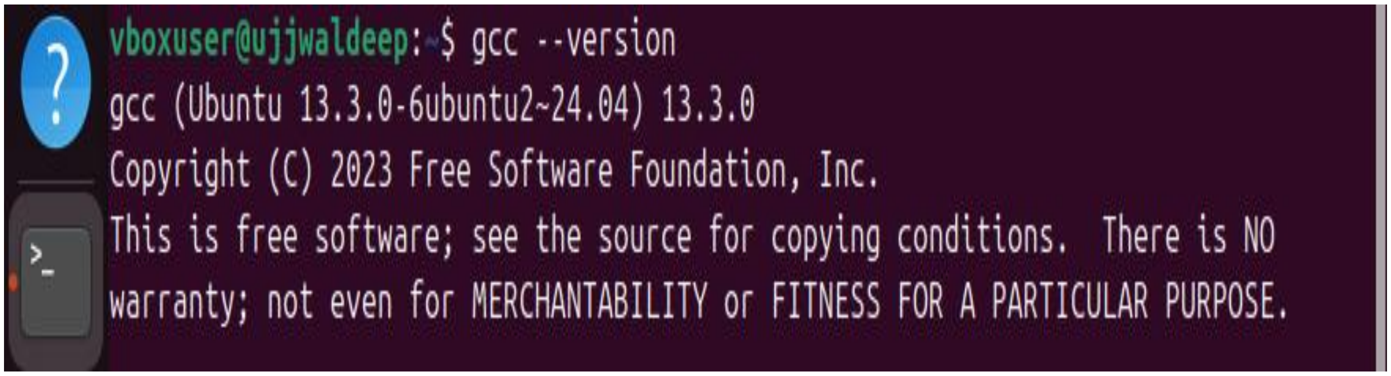
$sudo apt install gcc

This installs the GCC compiler on Debian/Ubuntu systems with superuser permissions.



```
vboxuser@ujjwaldeep:~$ sudo apt install gcc
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:13.2.0-7ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 327 not upgraded.
vboxuser@ujjwaldeep:~$
```

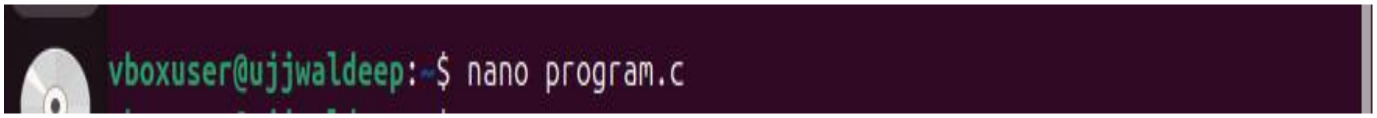
Step 2: On running this command in the Linux terminal, it asks for the password set-up by the user while installing Ubuntu on the system. On typing the password, GCC compiler is now installed on the Linux



```
vboxuser@ujjwaldeep:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

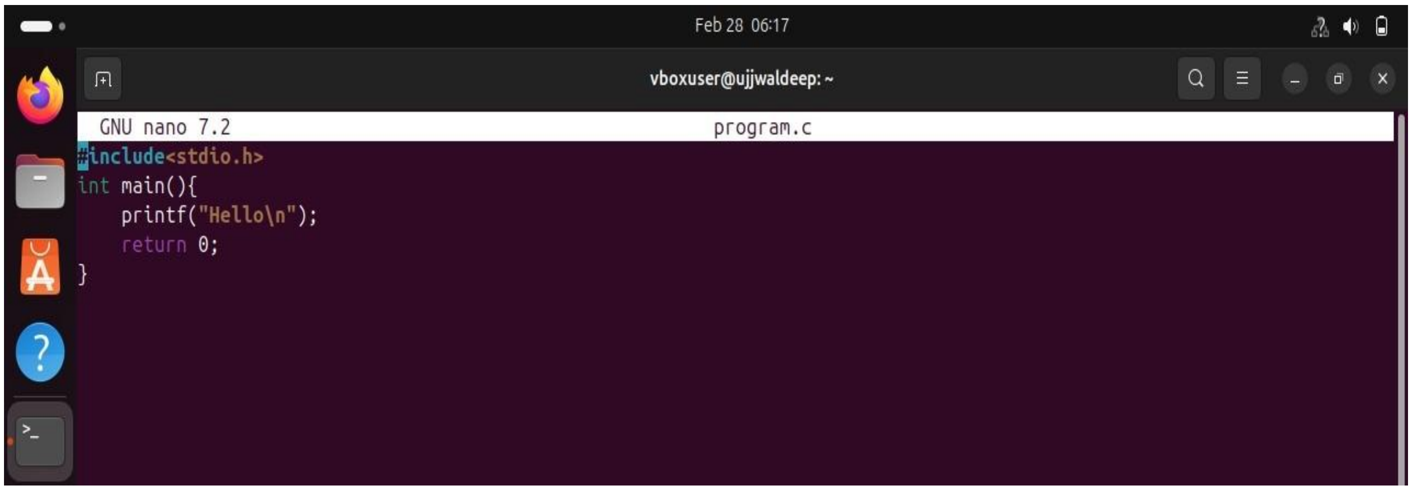
Step 3: Open the terminal on the Linux.

Step 4: write code: nano program.c



```
vboxuser@ujjwaldeep:~$ nano program.c
```

Step 5: Then GNU nano terminal to write a c program will open and then write the c program.



```
  GNU nano 7.2                                  program.c
#include<stdio.h>
int main(){
    printf("Hello\n");
    return 0;
}
```

Title:C) Time Stamping the file.

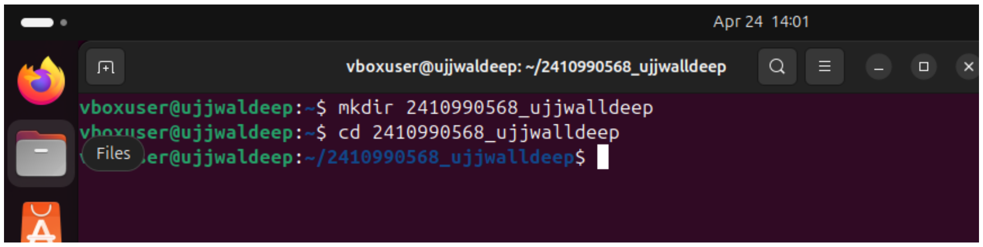
Solution:

Theory: Time stamping in Linux refers to recording the date and time associated with a file or event. Every file in Linux has three main timestamps: Access Time (atime), Modification Time (mtime), and Change Time (ctime). These timestamps help track when a file was last accessed, modified, or its metadata was changed.

Procedure:

Step 1: Create a folder 2410990568_Ujjwaldeep kaur using the following command:

$ mkdir foldername

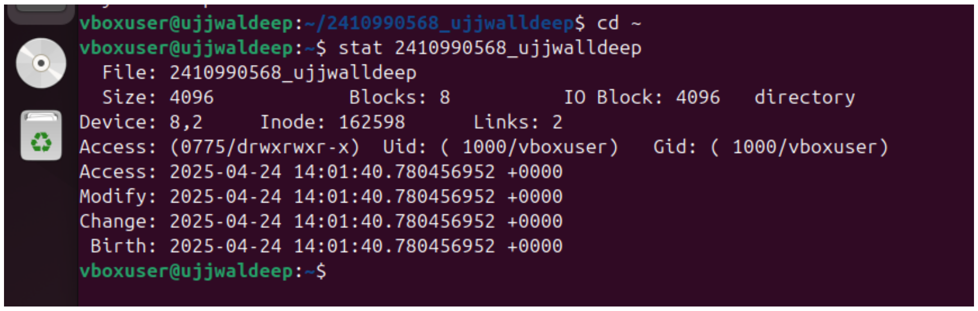


```
vboxuser@ujjwaldeep:~$ mkdir 2410990568_ujjwalldeep
vboxuser@ujjwaldeep:~$ cd 2410990568_ujjwalldeep
vboxuser@ujjwaldeep:~/2410990568_ujjwalldeep$
```

Step 2: To get information about the last status change time (ctime) of a file in Linux, use the following command:

$ stat foldername

Running the command on terminal will give:



```
vboxuser@ujjwaldeep:~/2410990568_ujjwalldeep$ cd ~
vboxuser@ujjwaldeep:~$ stat 2410990568_ujjwalldeep
  File: 2410990568_ujjwalldeep
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 8,2     Inode: 162598      Links: 2
Access: (0775/drwxrwxr-x)  Uid: ( 1000/vboxuser)   Gid: ( 1000/vboxuser)
Access: 2025-04-24 14:01:40.780456952 +0000
Modify: 2025-04-24 14:01:40.780456952 +0000
Change: 2025-04-24 14:01:40.780456952 +0000
 Birth: 2025-04-24 14:01:40.780456952 +0000
vboxuser@ujjwaldeep:~$
```

Time stamping in Linux helps track file activity using atime, mtime, and ctime. Commands like stat filename and ls -l –time=mtime provide detailed timestamp information. Managing timestamps is useful for system monitoring, debugging, and file management.

Title:D) Execution of the program in Linux Terminal.

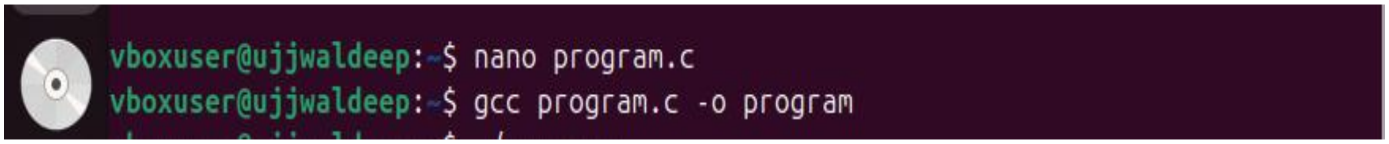
Solution:

Theory: Once a C program is compiled using GCC (GNU Compiler Collection), an executable file is generated, which can be executed in the Linux terminal. In Linux, executables do not have extensions like .exe in Windows. Instead, they are run using the ./ command.

Procedure:

Step 1: When compiling a program with GCC, the -o option is used to specify the name of the output file. The following command compiles code.c and generates an executable

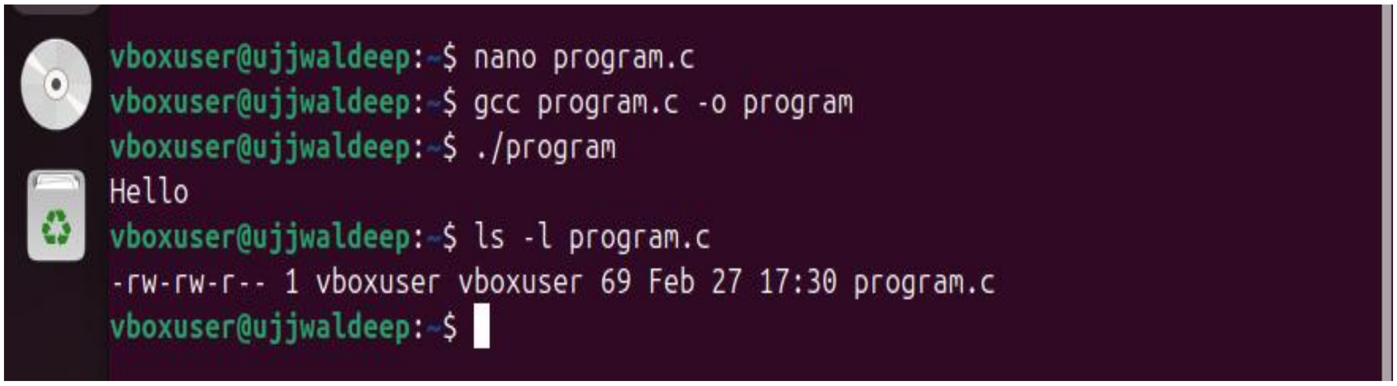
gcc program.c -o program



```
vboxuser@ujjwaldeep:~$ nano program.c
vboxuser@ujjwaldeep:~$ gcc program.c -o program
```

Step 2: Once compiled, the program can be executed using the following command:

./program



```
vboxuser@ujjwaldeep:~$ nano program.c
vboxuser@ujjwaldeep:~$ gcc program.c -o program
vboxuser@ujjwaldeep:~$ ./program
Hello
vboxuser@ujjwaldeep:~$ ls -l program.c
-rw-rw-r-- 1 vboxuser vboxuser 69 Feb 27 17:30 program.c
vboxuser@ujjwaldeep:~$ 
```

Practical 2

Title: Implement the commands that is used for Creating and Manipulating files: cat, cp, mv, rm, ls and its options, touch and their options.

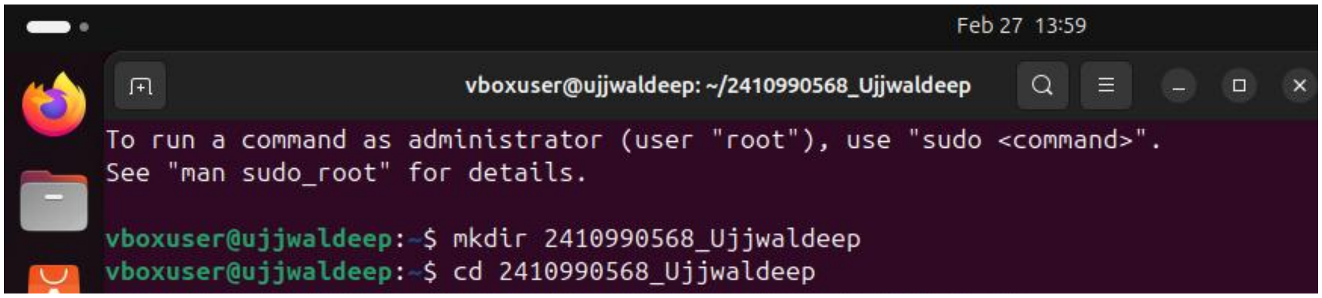
Solution: cat - concatenate and display files: The cat command is used to display the contents of a file or combine multiple files.

Theory: In Linux, file creation and manipulation are essential operations performed using various commands. The cat command is used to create and view file contents, while cp allows copying files and directories. The mv command helps in renaming or moving files, and rm is used for deleting files and directories. The ls command lists files in a directory, providing details like size and permissions. Additionally, the touch command is used to create empty files or update timestamps. These commands enable efficient file management, making Linux a powerful operating system for handling data.

Procedure:

Step 1: Create a folder "2410990568_Ujjwaldeep" using the following command:

$ mkdir foldername



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vboxuser@ujjwaldeep:~$ mkdir 2410990568_Ujjwaldeep
vboxuser@ujjwaldeep:~$ cd 2410990568_Ujjwaldeep
```

Step 2: Create a new file using the following command:

$ cat > filename

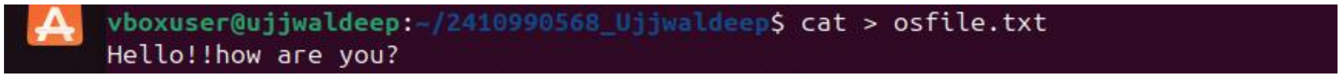
Syntax: stat [options] foldername

- **cat :** Short for "concatenate",but often used to create,view,or combine files.

Common Options

Option	Description
`cat filename`	View contents of the file
`cat file1 file2`	Combine files and display output
`cat -n filename`	Format output using a custom string
`cat > filename`	Create/overwrite a file
`cat >> filename`	Append to a file

Example Usage
cat > note.txt
This is a new file.
Press CTRL + D

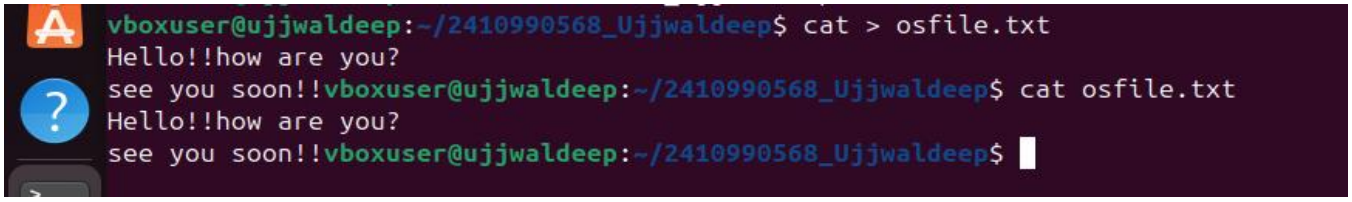


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ cat > osfile.txt
Hello!!how are you?
```

Creates a new file and allows you to enter text. Press **Ctrl + D** to save.

Step 3: To view the contents of a file, use the following command:

$ cat filename



```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ cat > osfile.txt
Hello!!how are you?
see you soon!!vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ cat osfile.txt
Hello!!how are you?
see you soon!!vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ 
```

Displays the content of the file.

Step 5: The cp (copy) command in Linux is used to copy files and directories. To copy a file use the following command:

$ cp source file destination

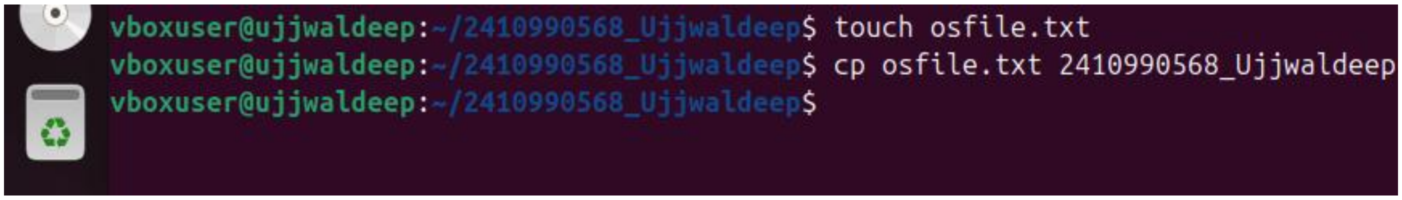
Syntax: cp sourcefile destination

- **cp :** Stands for copy

Common Options

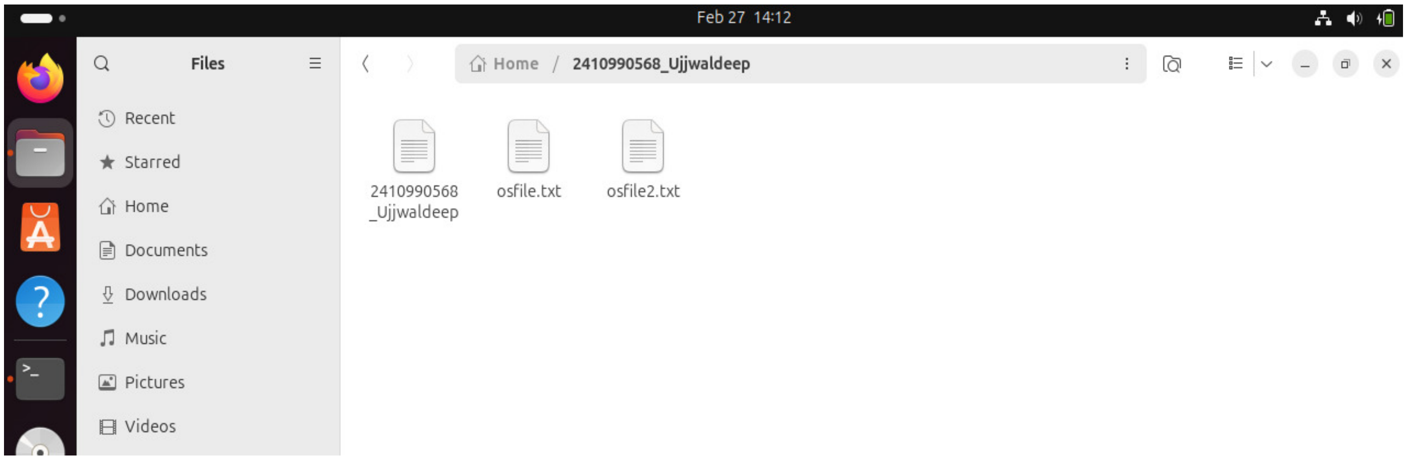
Option	Description
`cp file1 file2`	Copy `file1` to `file2`
`cp file1 dir/`	Copy `file1` into directory
`cp -i file1 file2`	Prompt without overwriting
`cp -r dir1 dir2`	Copy directories recursively
`cp -v file1 file2`	Verbose mode- show files being copied

Example Usage
cp report.txt backup/



```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ touch osfile.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ cp osfile.txt 2410990568_Ujjwaldeep
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

This copies the source_file “osfile.txt” to destination “2410990568_Ujjwaldeep” directory.



This copies the source_file “osfile.txt” to destination “2410990568_Ujjwaldeep” directory.

Step 6: The mv command is used to move or rename files and directories. To rename a file use the following command:

$ mv old_name new_name

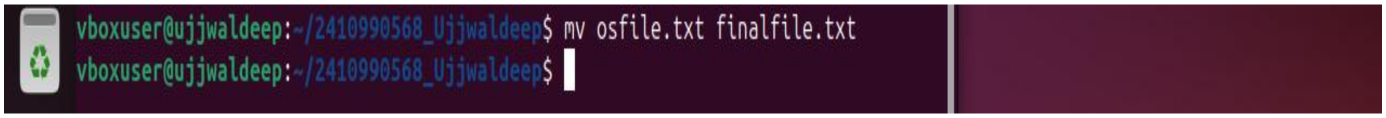
Syntax: mv old_name new_name

- **mv:** Stands for move

Common Options

Option	Description
`mv file1 file2`	rename `file1` to `file2`
`mv file1 dir/`	Move `file1` into directory
`mv -i file1 file2`	Prompt before overwriting
`mv -v file1 file2`	Show what is being moved
`mv dir1 dir2`	Move directory

Example Usage
mv report.txt summary.txt

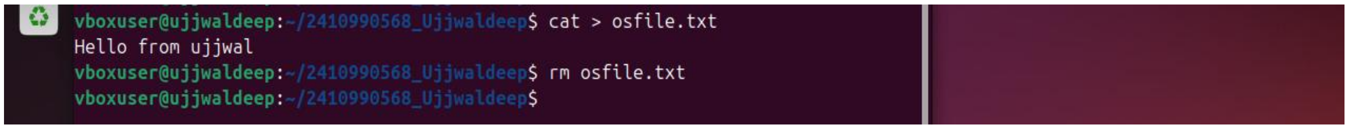


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ mv osfile.txt finalfile.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

This renames old_name “osfile.txt” to new_name “finalfile.txt”.

Step 7: The rm command is used to delete files and directories. To remove a file, use the following command:

$ rm filename



```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ cat > osfile.txt
Hello from ujjwal
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ rm osfile.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

This deletes the file “osfile.txt” from the directory.

Step 8: The ls command is used to list files and directories in a directory. To get the basic list of files, use the following command:

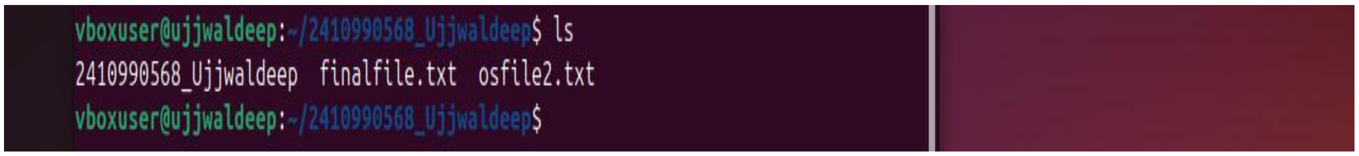
$ls

Syntax: `ls [OPTION] [FILE OR DIRECTORY]`

- `OPTION:` flags that modify command behavior.

Common Options

Option	**Description**
`-l`	Long listing format
`-a`	Show all files
`-r`	Reverse the sort order
`-h`	Human readable file sizes
`-t`	Sort by modification time
`-S`	Sort by file size
`-i`	Show inode numbers
`-R`	List all directories and **all subdirectories**
`-1`	One file per line

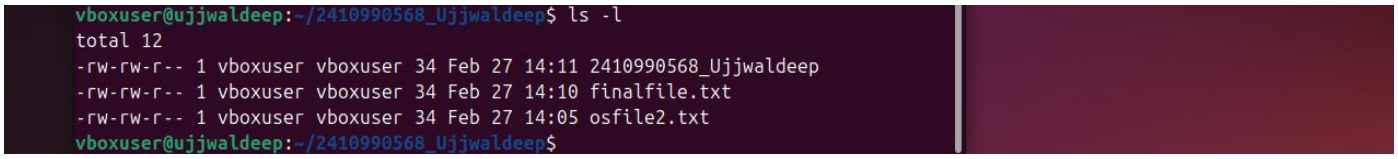


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls
2410990568_Ujjwaldeep  finalfile.txt  osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

This displays the names of files and directories in the current directory.

Step 9: To list files with detailed information, use the following command

$ ls-1

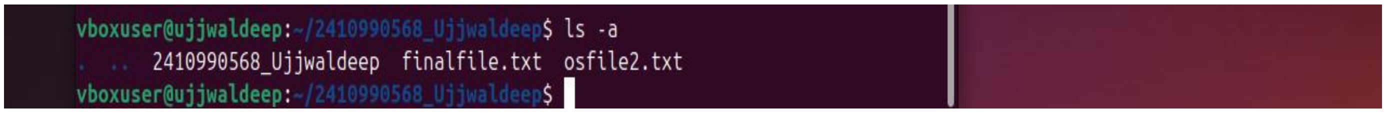


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls -l
total 12
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:11 2410990568_Ujjwaldeep
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:10 finalfile.txt
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:05 osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Shows file details such as permissions, owner, size and modification date

Step 10: To list the hidden files, use the following command:

$ls -a

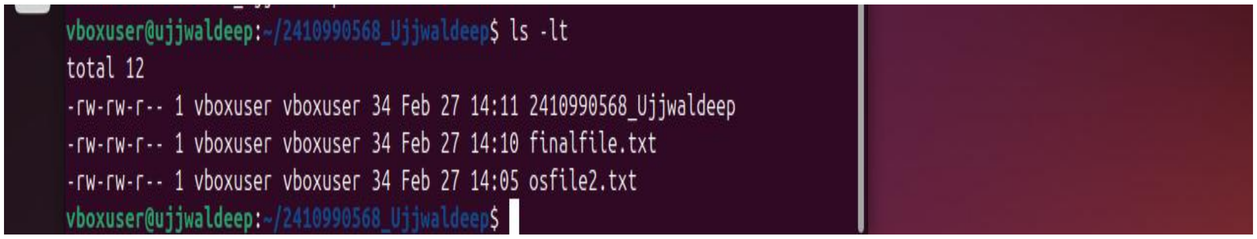


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls -a
.  ..  2410990568_Ujjwaldeep  finalfile.txt  osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Displays all files including hidden ones (files starting with **.**)

Step 11: To list files sorted by Modification Time, use the following command:

$ls -lt

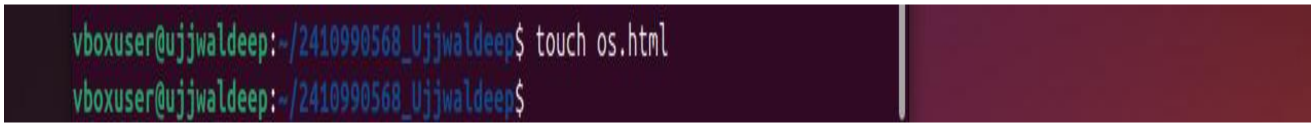


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls -lt
total 12
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:11 2410990568_Ujjwaldeep
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:10 finalfile.txt
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:05 osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

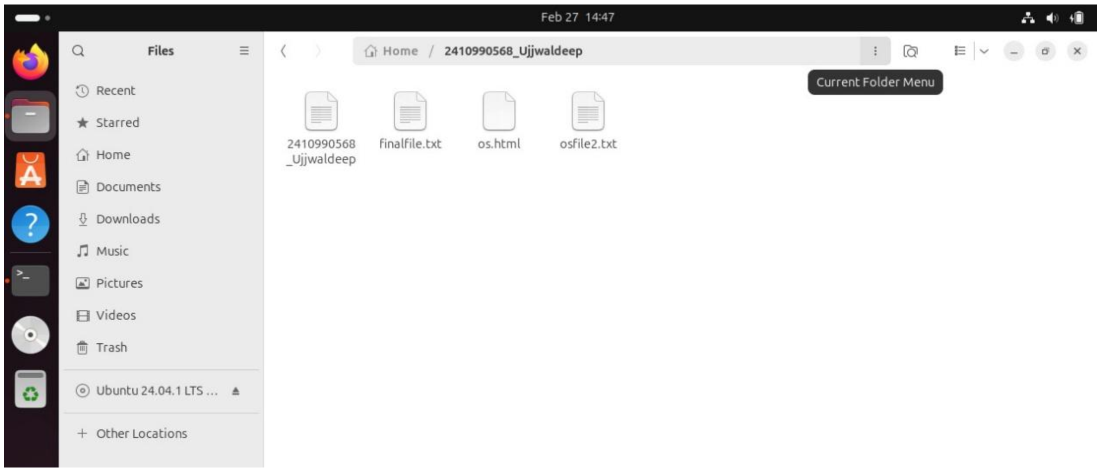
This sorts files by the most recently modified.

Step 12: The touch (Create or Update Files) command is used to create empty files and update timestamps of existing files. To create new empty files, use the following command:

$ touch filename



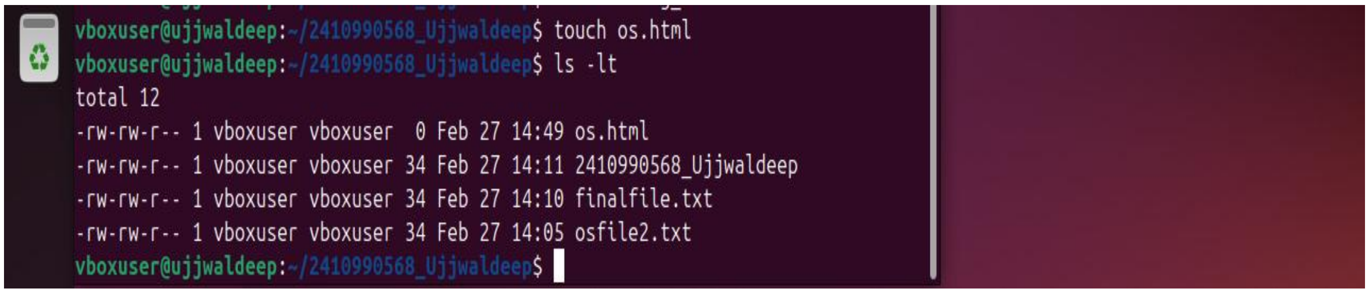
```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ touch os.html
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```



This creates an empty file if it does not already exist.

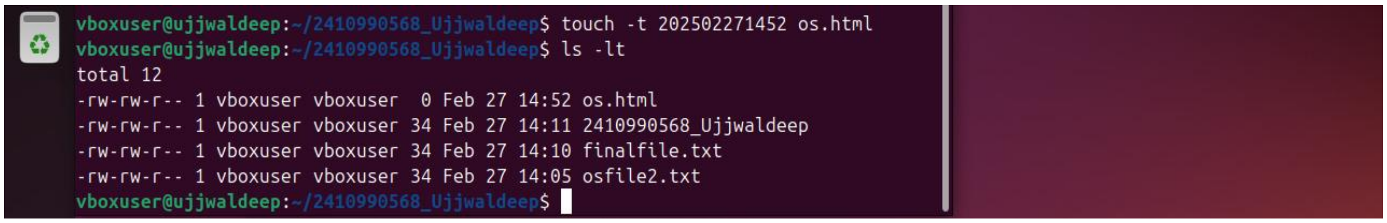
Step 13: To update the timestamp of an existing file, use the following command:

$ touch existing file


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ touch os.html
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls -lt
total 12
-rw-rw-r-- 1 vboxuser vboxuser  0 Feb 27 14:49 os.html
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:11 2410990568_Ujjwaldeep
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:10 finalfile.txt
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:05 osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Step 14: To set a specific timestamp, use the following command:

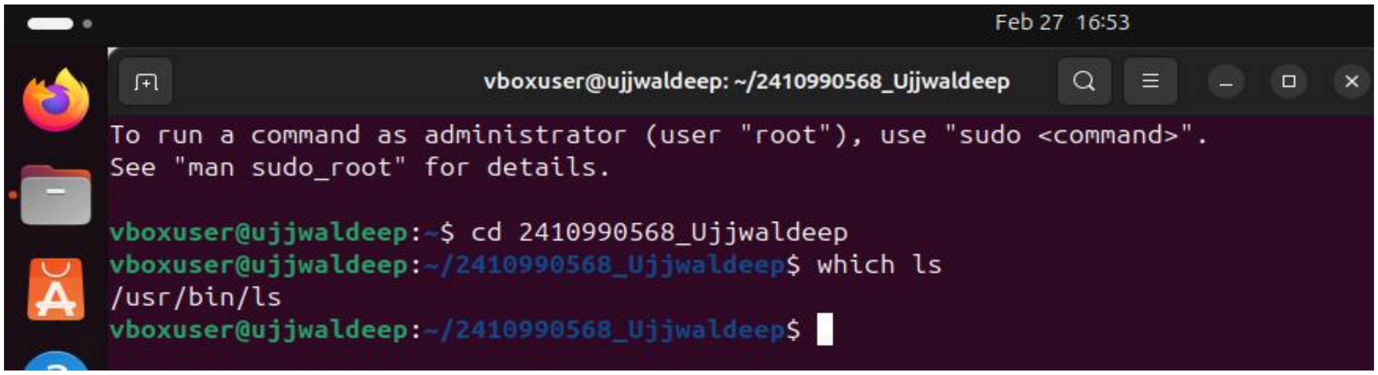
$ touch -t YYYYMMDDhhmm filename


```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ touch -t 202502271452 os.html
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ ls -lt
total 12
-rw-rw-r-- 1 vboxuser vboxuser  0 Feb 27 14:52 os.html
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:11 2410990568_Ujjwaldeep
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:10 finalfile.txt
-rw-rw-r-- 1 vboxuser vboxuser 34 Feb 27 14:05 osfile2.txt
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Changes the timestamp to 4th April 2025, 11:34 PM (YYYYMMDDhhmm format).

Step15: which: This command is used to locate the binary file associated with a given command.

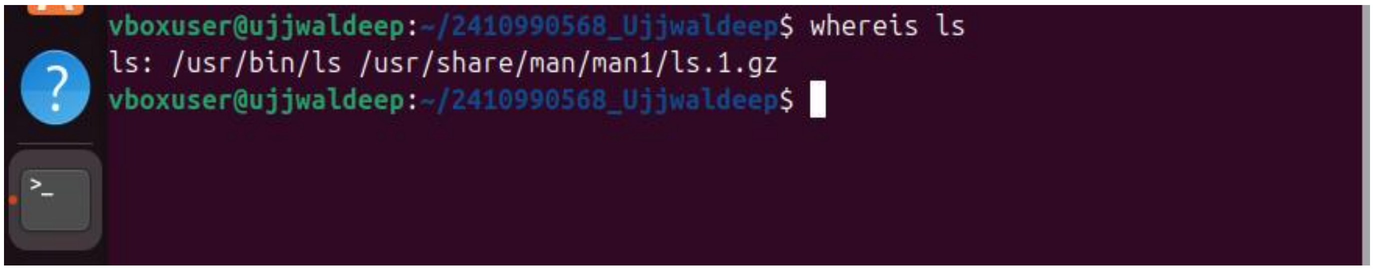
$ Which ls



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vboxuser@ujjwaldeep:~$ cd 2410990568_Ujjwaldeep
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ which ls
/usr/bin/ls
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Step16: whereis This command is used to locate the binary, course, and manual page files for a command.

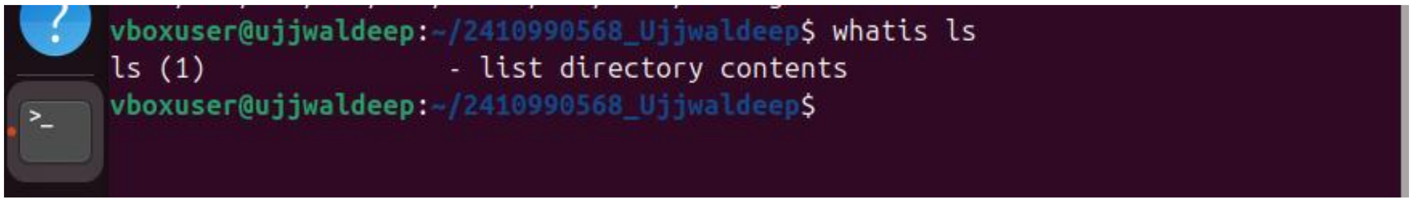
$ Whereis ls



```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Step17: whatisThis command is used to display a short description of a command

$ **Whatis ls**



```
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$ whatis ls
ls (1)               - list directory contents
vboxuser@ujjwaldeep:~/2410990568_Ujjwaldeep$
```

Practical 3

Title: Implementing Directory oriented commands like: cd, pwd, mkdir, rmdir.

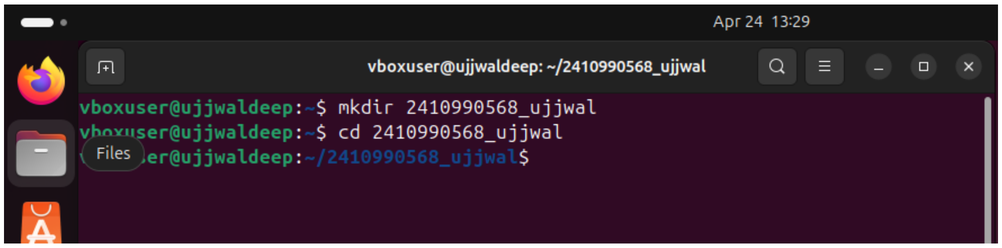
Solution:

Theory: Directory-oriented commands in Linux are used to navigate, create, modify, and delete directories efficiently. The pwd command displays the current directory path, while cd allows movement between directories. New directories can be created using mkdir, and empty ones can be removed with rmdir.

Steps:

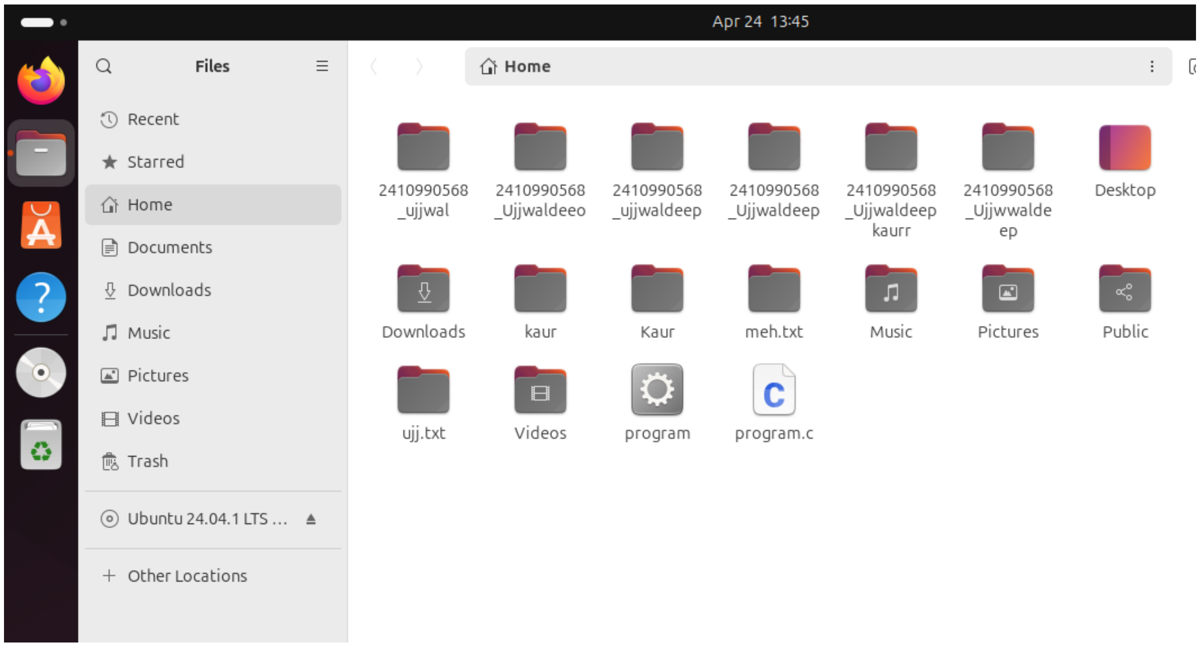
Step 1: To create a folder in the Linux, use the following command on the Linux terminal:

$mkdir foldername



```
vboxuser@ujjwaldeep:~$ mkdir 2410990568_ujjwal
vboxuser@ujjwaldeep:~$ cd 2410990568_ujjwal
vboxuser@ujjwaldeep:~/2410990568_ujjwal$
```

This command creates the following folder on Linux.



Step 2: The cd (Change Directory) command in Linux is used to navigate between directories. Using the following command, you can navigate between the directories:

$cd foldername

- The `cd` command (**change directory**) is used to navigate between directories (folders) in a Linux/Unix terminal.

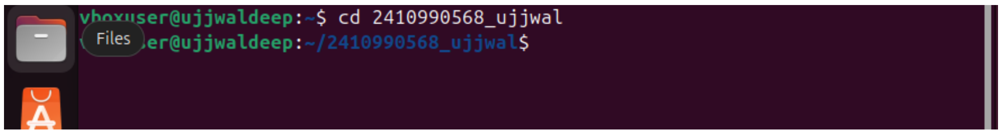
Syntax: `cd foldername`

Common Options

Option	**Description**
`cd Documents`	Moves into the Documents folder inside the current directory
`cd /home/userDocuments`	Moves directly to an **absolute path**
`cd ..`	Moves **up** one level to the parent directory
`cd ..//..`	Moves **up** two levels
`cd /`	Moves to the **root directory**
`cd ~`	Moves to your **home directory**
`cd -`	Moves back to the **previous directory you were in.**

Example Usage

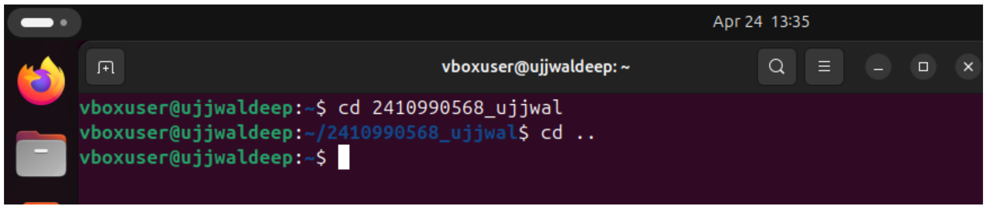
If you are in:
`/home/user`
and you run:
`cd documents`
Your prompt will now show:
`/home/user/documents`



```
vboxuser@ujjwaldeep:~$ cd 2410990568_ujjwal
vboxuser@ujjwaldeep:~/2410990568_ujjwal$
```

Step 3: To go back to the previous directory, use the following command on the terminal:

$cd ..

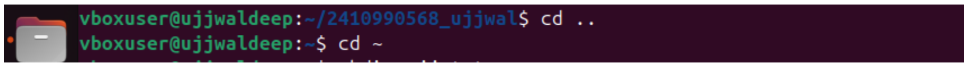


```
vboxuser@ujjwaldeep:~$ cd 2410990568_ujjwal
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ cd ..
vboxuser@ujjwaldeep:~$ 
```

This command moves one level up from code to its parent directory.

Step 4: To return to the home directory from the file, use the following command:

$ cd~



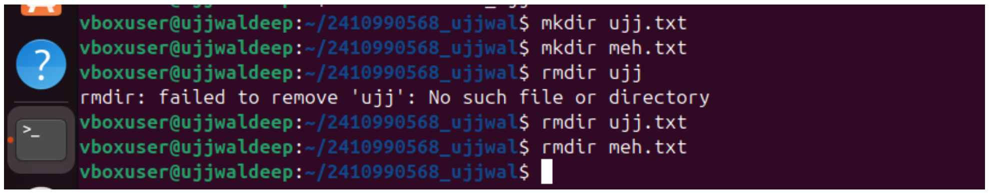
```
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ cd ..
vboxuser@ujjwaldeep:~$ cd ~
```

This command moves back to the home directory from the nested directory

Step 5: The rmdir command in Linux is used to remove empty directories using the following command on the terminal:

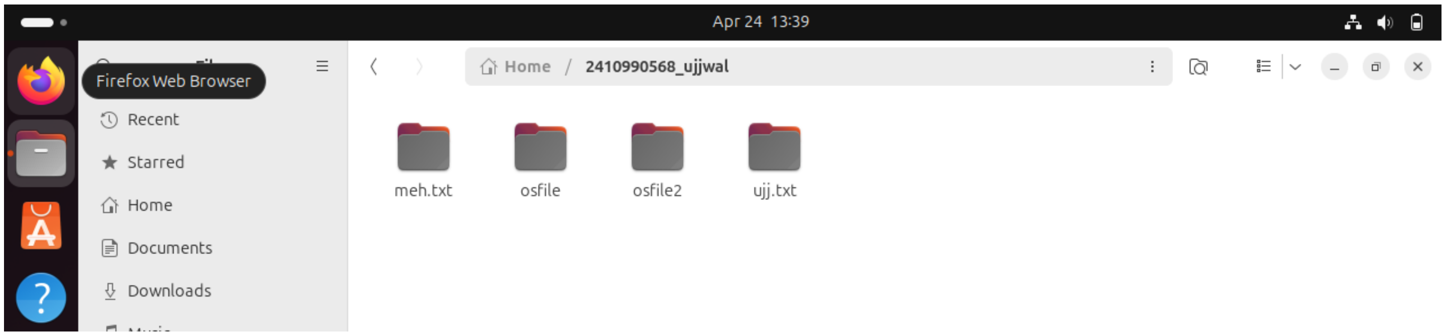
$rmdir filename

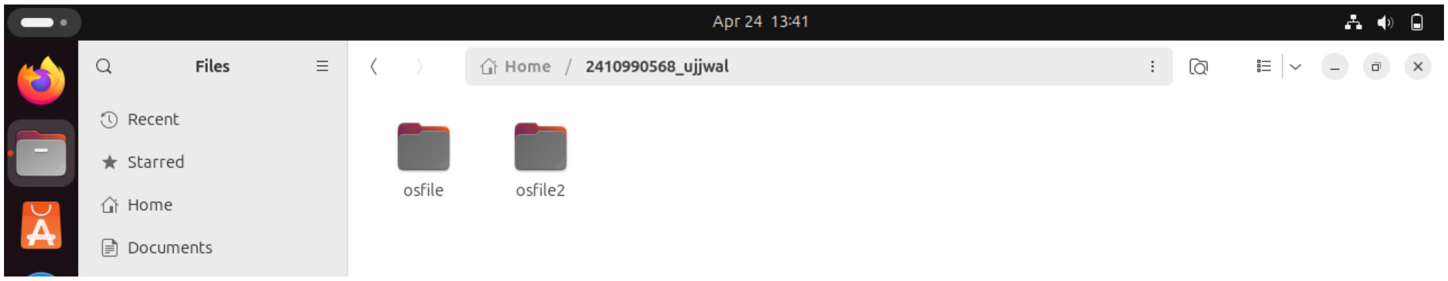
First, create an empty directory using the mkdir command, such as **$mkdir foldername**. Once the directory is created, it can be removed using the rmdir command, like **$rmdir foldername**, but only if it is empty.



```
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ mkdir ujj.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ mkdir meh.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ rmdir ujj
rmdir: failed to remove 'ujj': No such file or directory
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ rmdir ujj.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ rmdir meh.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwal$ 
```

Running the command removes the folder from the directory.


Apr 24 13:39
Firefox Web Browser
Recent
Starred
Home
Documents
Downloads
Home / 2410990568_ujjwal
meh.txt
osfile
osfile2
ujj.txt


Apr 24 13:41
Files
Recent
Starred
Home
Documents
Home / 2410990568_ujjwal
osfile
osfile2

Practical 4

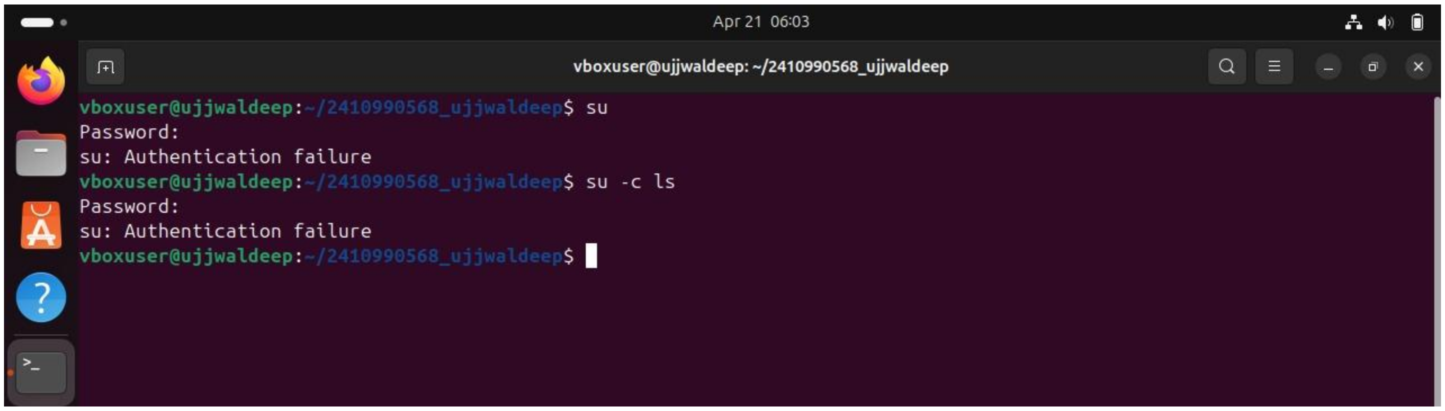
Title: Implement the basic and user status commands like: su, sudo, man, help, history, who, whoami, id, uname, uptime, free, tty, cal, date, hostname, reboot, clear

Solution:

1. **su (Switch User) :** The su command allows you to switch users.

SYNTAX:
su [OPTION] [USER]

Common Options:
- -l: Start a login shell.
- -c: Execute a command as the target user.

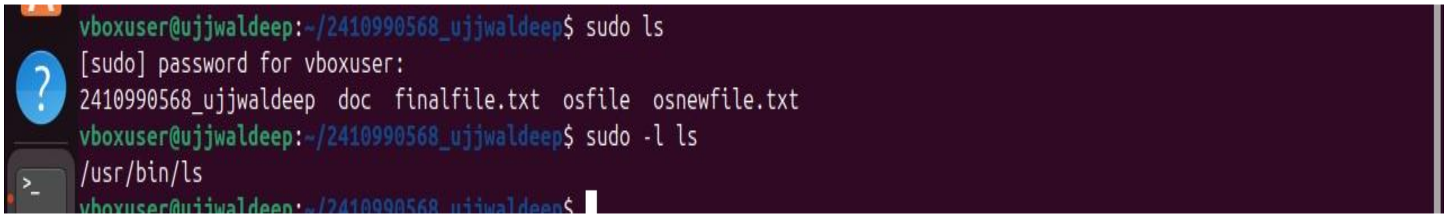


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ su
Password:
su: Authentication failure
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ su -c ls
Password:
su: Authentication failure
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

2. **sudo (Superuser Do) :** sudo is used to run a command as a superuser.

SYNTAX:
sudo [OPTION] COMMAND [ARGUMENTS]

Common Options:
- -u: Run a command as a specific user.
- -l: list allow for user
- -v: update user credentials

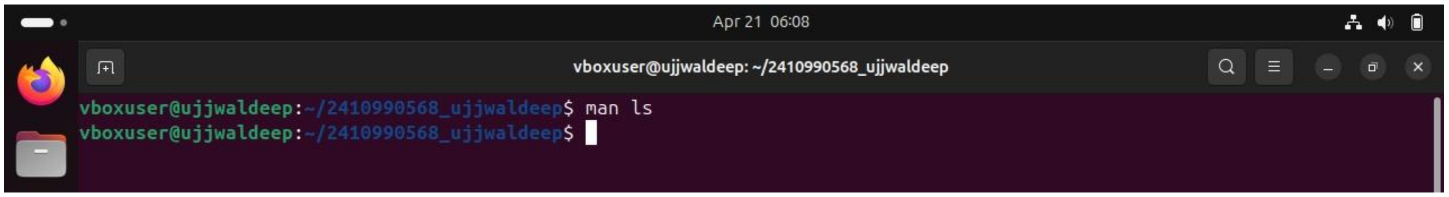


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ sudo ls
[sudo] password for vboxuser:
2410990568_ujjwaldeep  doc  finalfile.txt  osfile  osnewfile.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ sudo -l ls
/usr/bin/ls
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

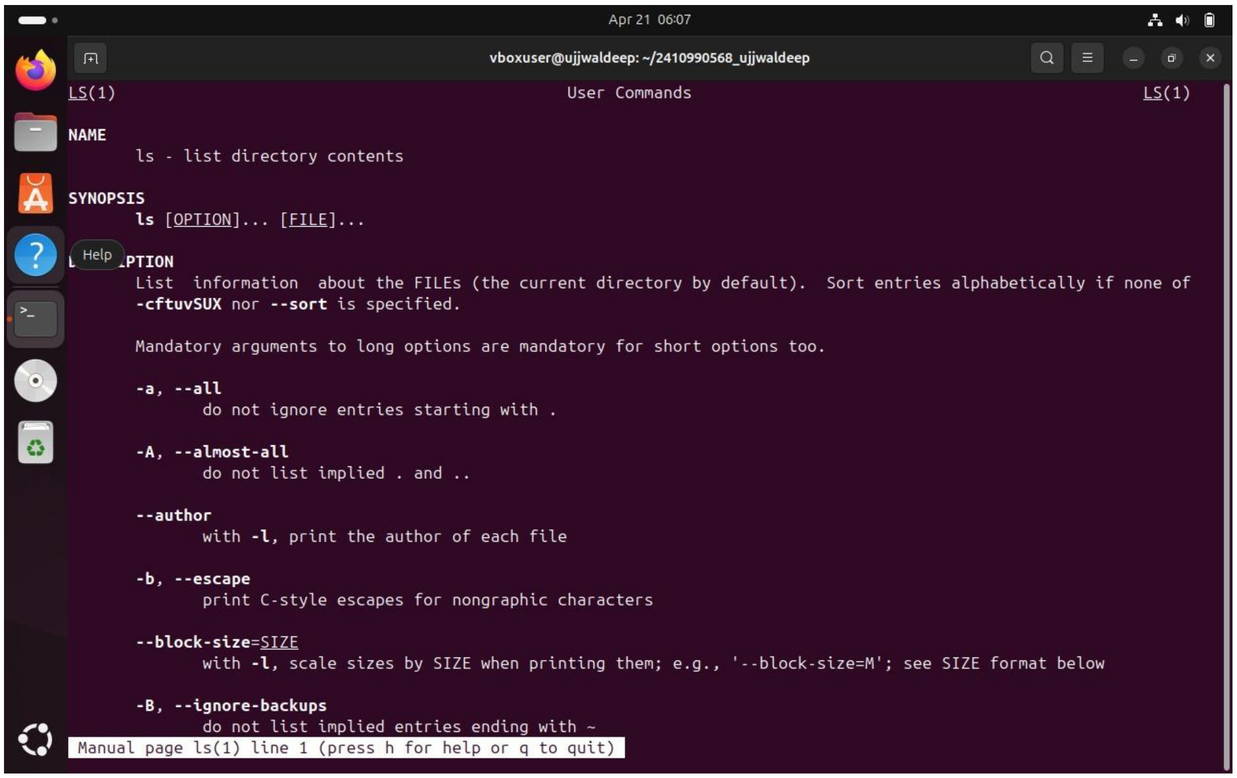
3. **man (Manual Pages) :** The man command shows the manual page for a command.

SYNTAX:
man [OPTION] COMMAND

Common Options:
• -k: Search for a keyword in manual pages.
• -f: Display short descriptions of commands.



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ man ls
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

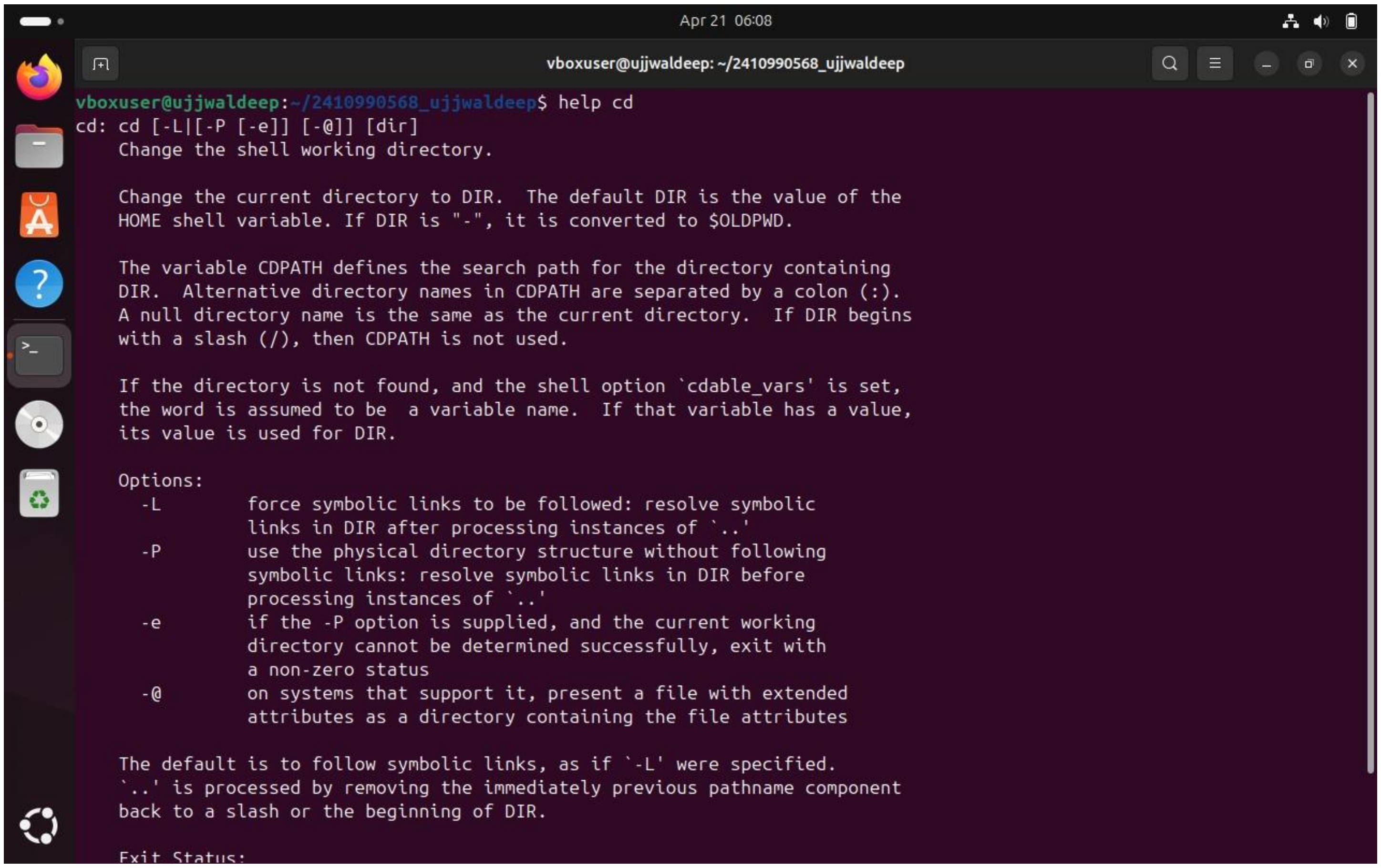


```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).  Sort entries alphabetically if none of
       -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~
 Manual page ls(1) line 1 (press h for help or q to quit)
```

4. help (Help Command) : The help command is used to display a list of built-in shell commands or get information on a specific command.

SYNTAX
help [COMMAND]

Common Options
None

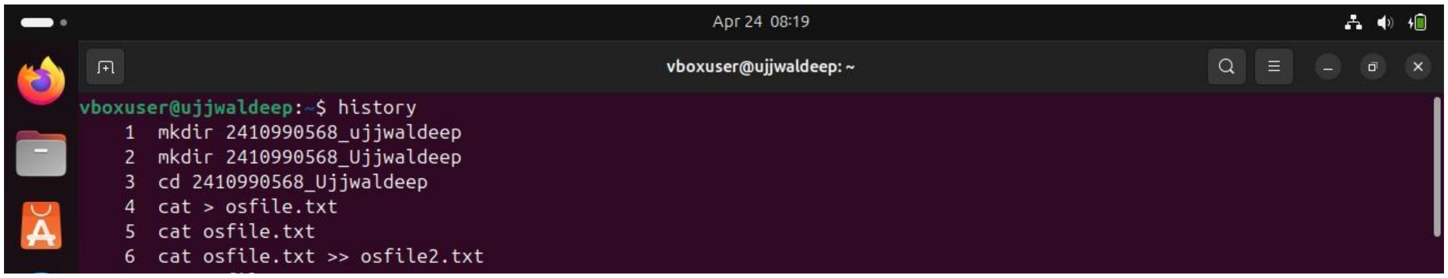


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable. If DIR is "-", it is converted to $OLDPWD.

    The variable CDPATH defines the search path for the directory containing
    DIR.  Alternative directory names in CDPATH are separated by a colon (:).
    A null directory name is the same as the current directory.  If DIR begins
    with a slash (/), then CDPATH is not used.

    If the directory is not found, and the shell option `cdable_vars' is set,
    the word is assumed to be  a variable name.  If that variable has a value,
    its value is used for DIR.

    Options:
      -L        force symbolic links to be followed: resolve symbolic
                links in DIR after processing instances of `..'
      -P        use the physical directory structure without following
                symbolic links: resolve symbolic links in DIR before
                processing instances of `..'
      -e        if the -P option is supplied, and the current working
                directory cannot be determined successfully, exit with
                a non-zero status
      -@        on systems that support it, present a file with extended
                attributes as a directory containing the file attributes

    The default is to follow symbolic links, as if `-L' were specified.
    `..' is processed by removing the immediately previous pathname component
    back to a slash or the beginning of DIR.

    Exit Status:
```

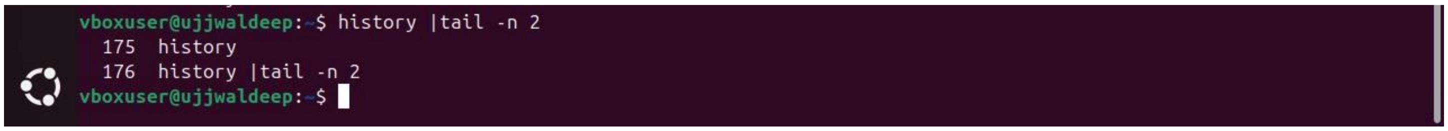
5. history (Command History): The history command shows the list of previously executed commands.

SYNTAX:
history [OPTION]

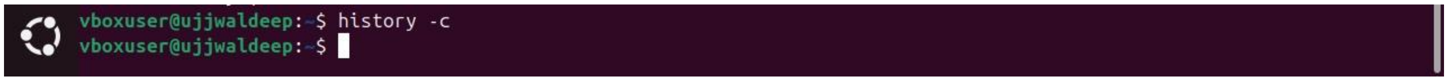
Common Options:
• -c: Clear the history list.
• n: Show the last n commands.



```
vboxuser@ujjwaldeep:~$ history
    1  mkdir 2410990568_ujjwaldeep
    2  mkdir 2410990568_Ujjwaldeep
    3  cd 2410990568_Ujjwaldeep
    4  cat > osfile.txt
    5  cat osfile.txt
    6  cat osfile.txt >> osfile2.txt
```



```
vboxuser@ujjwaldeep:~$ history |tail -n 2
  175  history
  176  history |tail -n 2
vboxuser@ujjwaldeep:~$ 
```

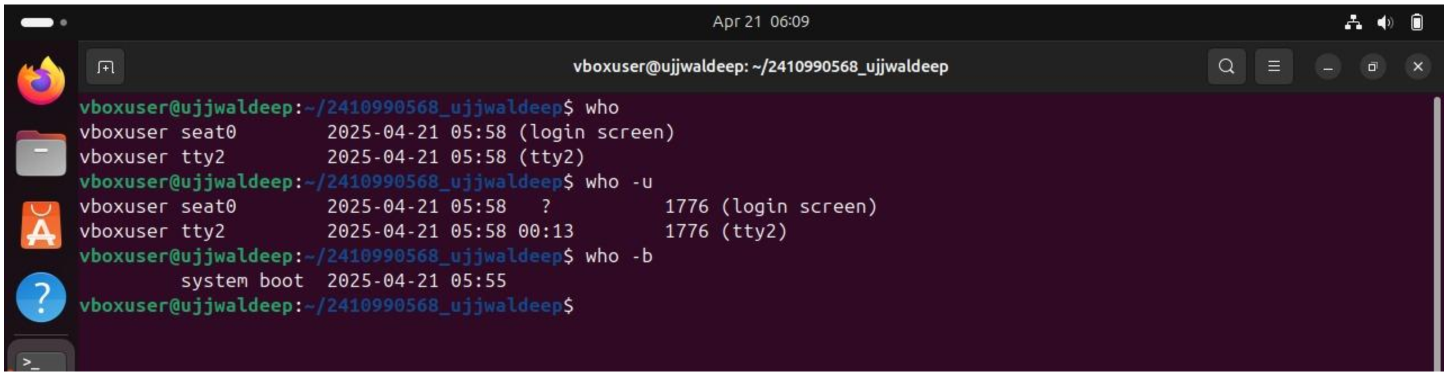


```
vboxuser@ujjwaldeep:~$ history -c
vboxuser@ujjwaldeep:~$ 
```

6. **who (Who is Logged In) :** The who command shows who is logged into the system.

   Syntax:
   who [OPTION]

   Common Options:
   • -b: Show the last system boot time.
   • -H: display column headers
   • -u: show idle time

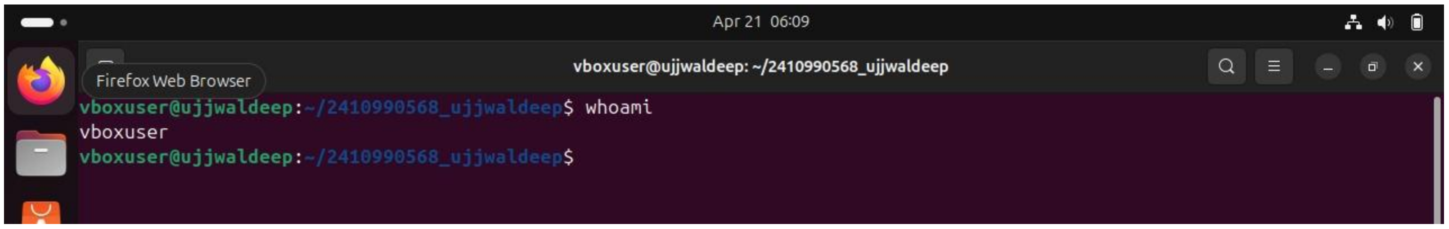


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ who
vboxuser seat0        2025-04-21 05:58 (login screen)
vboxuser tty2         2025-04-21 05:58 (tty2)
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ who -u
vboxuser seat0        2025-04-21 05:58   ?          1776 (login screen)
vboxuser tty2         2025-04-21 05:58 00:13        1776 (tty2)
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ who -b
         system boot  2025-04-21 05:55
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

7. **whoami (Current User)** : The whoami command shows the current user.

Syntax:
Whoami

Common Options:
none

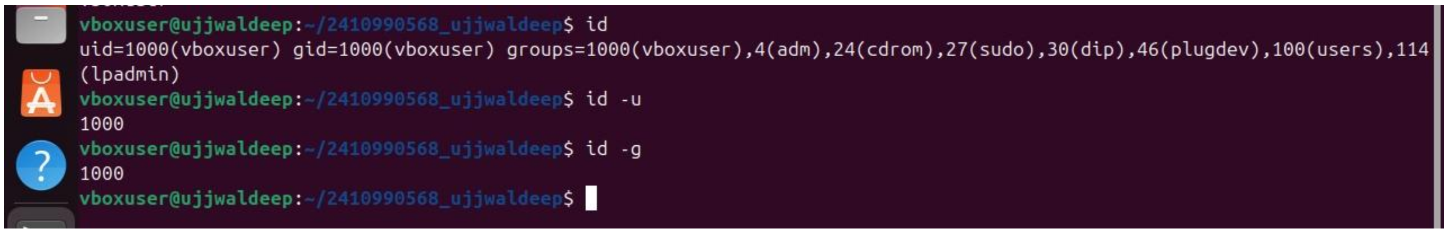


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ whoami
vboxuser
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

8. **id (User and Group Information) :** The id command displays user and group IDs.

Syntax:
id [OPTION]

Common Options:
• -u: Show the user ID (UID).
• -g: Show the group ID (GID).

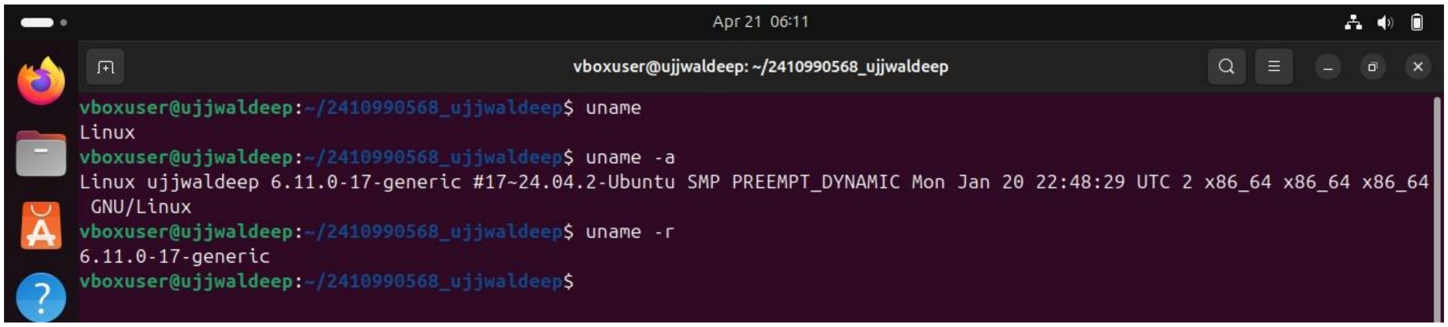


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ id
uid=1000(vboxuser) gid=1000(vboxuser) groups=1000(vboxuser),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114
(lpadmin)
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ id -u
1000
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ id -g
1000
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

9. **uname (System Information) :** The uname command shows system information.

Syntax:
uname [OPTION]

Common Options:
• -a: Show all system information.
• -r: Show kernel release.
• -m: Show machine hardware name.

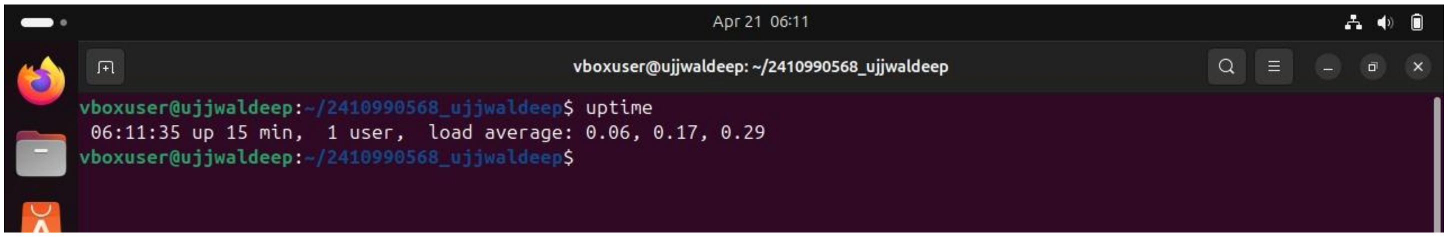


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ uname
Linux
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ uname -a
Linux ujjwaldeep 6.11.0-17-generic #17~24.04.2-Ubuntu SMP PREEMPT_DYNAMIC Mon Jan 20 22:48:29 UTC 2 x86_64 x86_64 x86_64
 GNU/Linux
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ uname -r
6.11.0-17-generic
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

10. uptime (System Uptime) : The uptime command shows how long the system has been running

Syntax:
Uptime

Options:
None

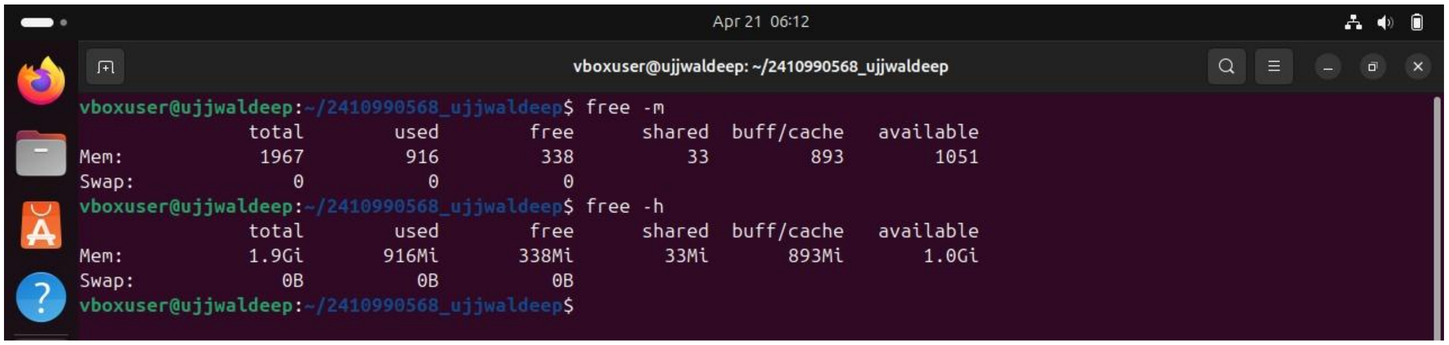


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ uptime
 06:11:35 up 15 min,  1 user,  load average: 0.06, 0.17, 0.29
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

11. free (Memory Usage) : The free command shows memory usage.

Syntax:
free [OPTION] Common

Options:
- -h: Show human-readable memory information (e.g., MB, GB).
- -m: Show memory usage in megabytes

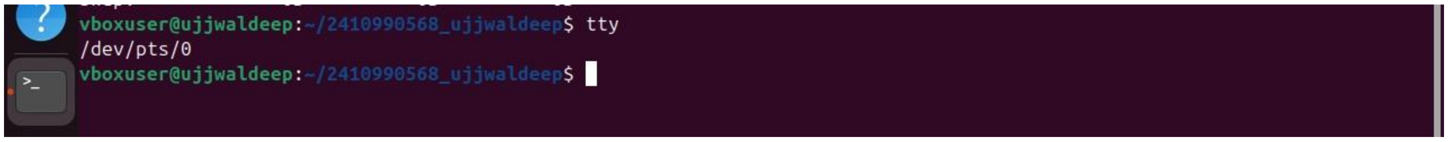


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ free -m
               total        used        free      shared  buff/cache   available
Mem:            1967         916         338          33         893        1051
Swap:              0           0           0
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ free -h
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       916Mi       338Mi        33Mi       893Mi       1.0Gi
Swap:             0B          0B          0B
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

12. tty (Terminal Information) : The tty command shows the terminal type.

Syntax:
tty

Common Options:
none

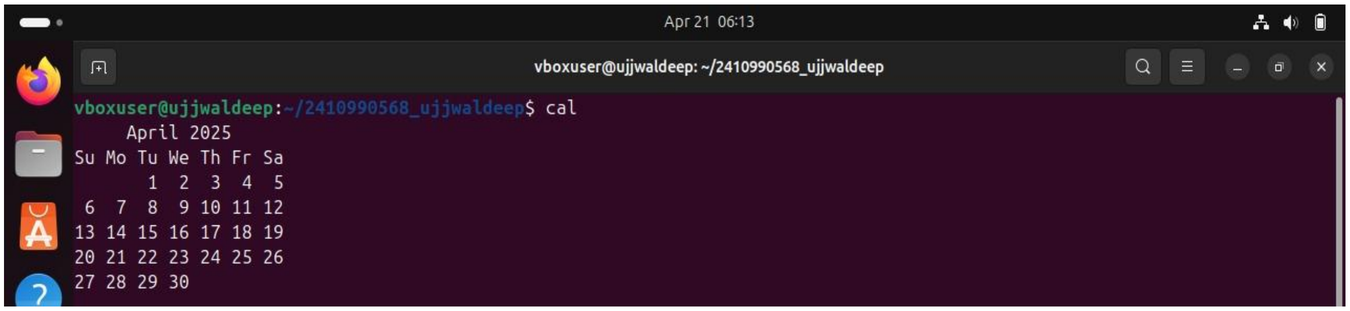


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ tty
/dev/pts/0
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

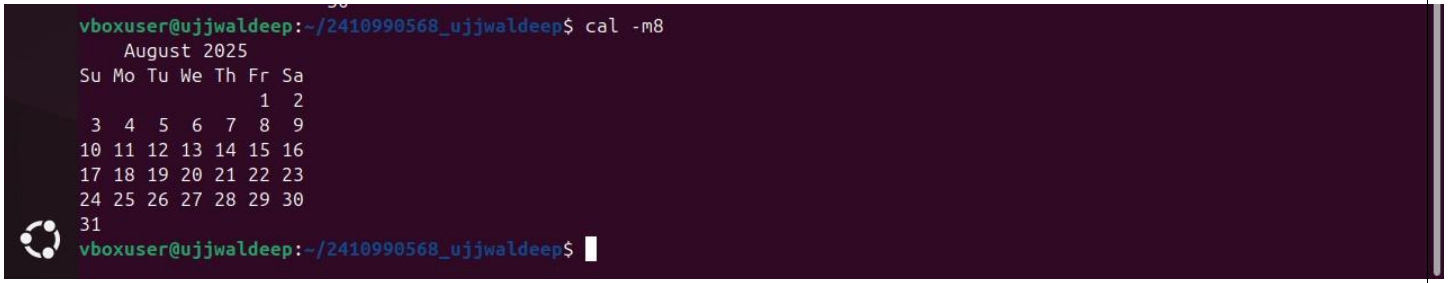
13. cal (Calendar) : The cal command shows the current month's

Syntax:
calendar cal [OPTION]

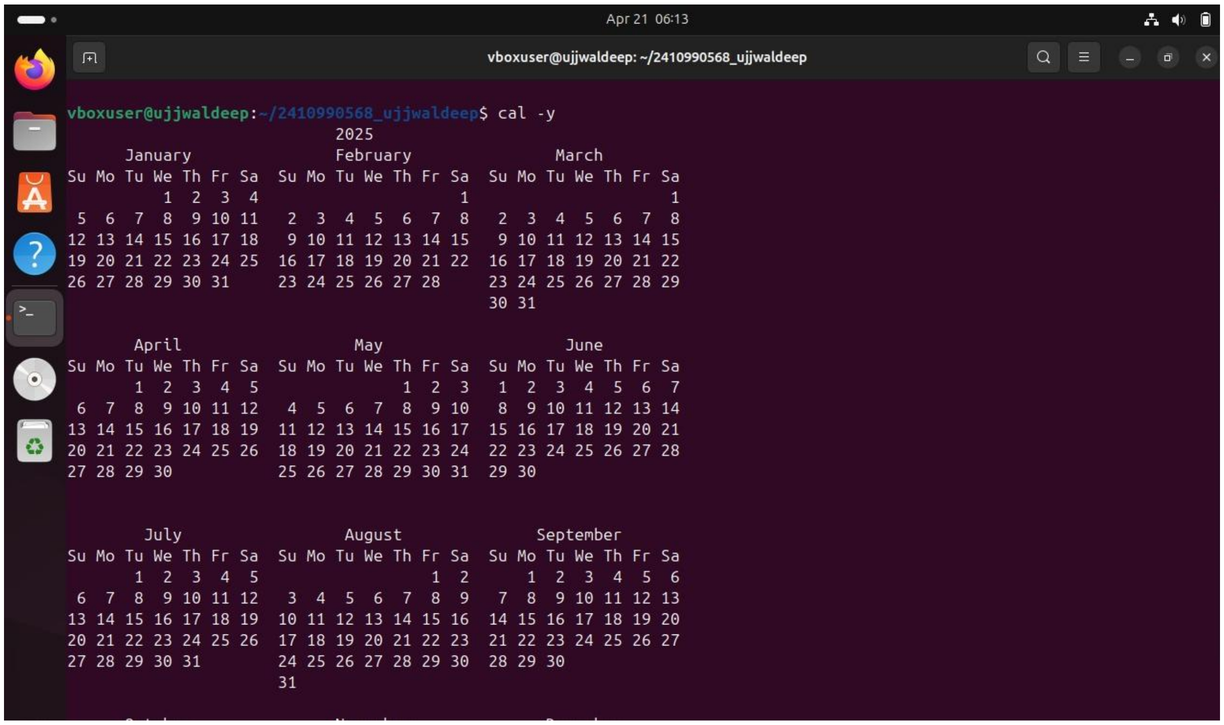
Common Options:
• -y : Show the calendar for the current year.
• -m : Specify the month.



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ cal
     April 2025
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
```



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ cal -m8
    August 2025
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
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```



Apr 21 06:13

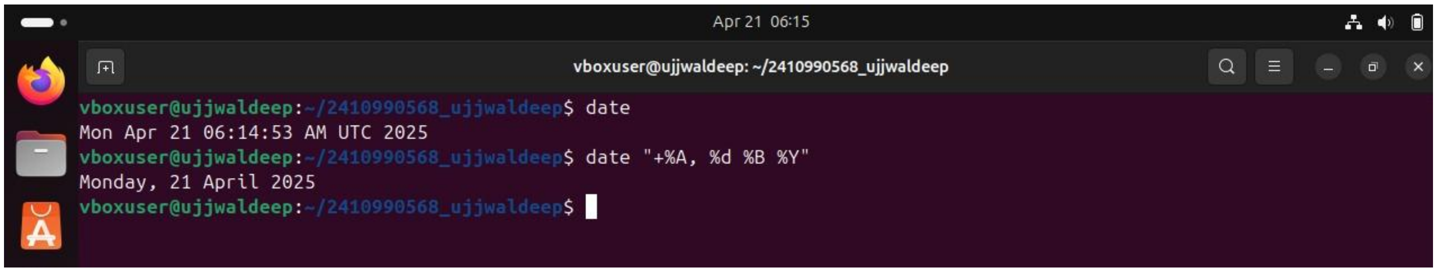
vboxuser@ujjwaldeep: ~/2410990568_ujjwaldeep

```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ cal -y
                            2025
      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  3  4                     1                     1
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   2  3  4  5  6  7  8
12 13 14 15 16 17 18   9 10 11 12 13 14 15   9 10 11 12 13 14 15
19 20 21 22 23 24 25  16 17 18 19 20 21 22  16 17 18 19 20 21 22
26 27 28 29 30 31     23 24 25 26 27 28     23 24 25 26 27 28 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  5               1  2  3   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   4  5  6  7  8  9 10   8  9 10 11 12 13 14
13 14 15 16 17 18 19  11 12 13 14 15 16 17  15 16 17 18 19 20 21
20 21 22 23 24 25 26  18 19 20 21 22 23 24  22 23 24 25 26 27 28
27 28 29 30           25 26 27 28 29 30 31  29 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  2  3  4  5                  1  2      1  2  3  4  5  6
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   7  8  9 10 11 12 13
13 14 15 16 17 18 19  10 11 12 13 14 15 16  14 15 16 17 18 19 20
20 21 22 23 24 25 26  17 18 19 20 21 22 23  21 22 23 24 25 26 27
27 28 29 30 31        24 25 26 27 28 29 30  28 29 30
                      31
```

14. **date (Current Date and Time) :** The date command shows the current date and time.

Syntax:
date [OPTION]

Common Options:
• +%Y-%m-%d: Format the date output (e.g., 2025-03-19).

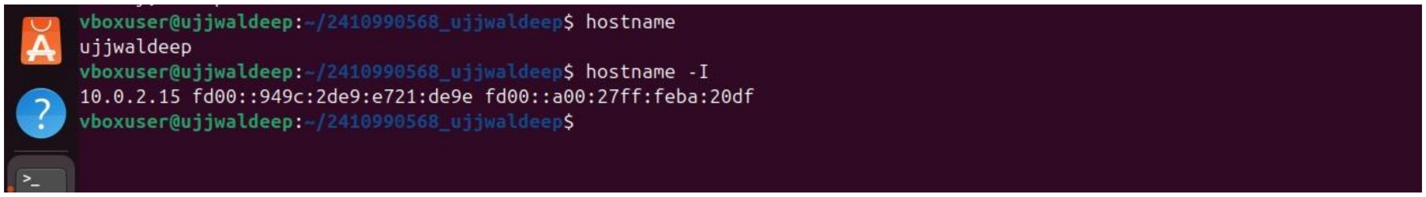


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ date
Mon Apr 21 06:14:53 AM UTC 2025
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ date "+%A, %d %B %Y"
Monday, 21 April 2025
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

15. hostname (System Hostname) : The hostname command shows the system's hostname.

Syntax:
hostname

Common Options:
• -I : Show all IP addresses.
• -s : Display the short hostname.

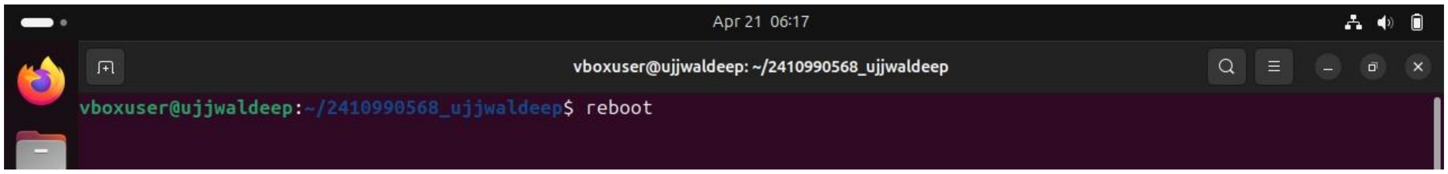


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ hostname
ujjwaldeep
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ hostname -I
10.0.2.15 fd00::949c:2de9:e721:de9e fd00::a00:27ff:feba:20df
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

16. reboot (Reboot the System): The reboot command restarts the system. It doesn't have many options, but it requires root privileges.

Syntax:
Reboot

Common Options:
none

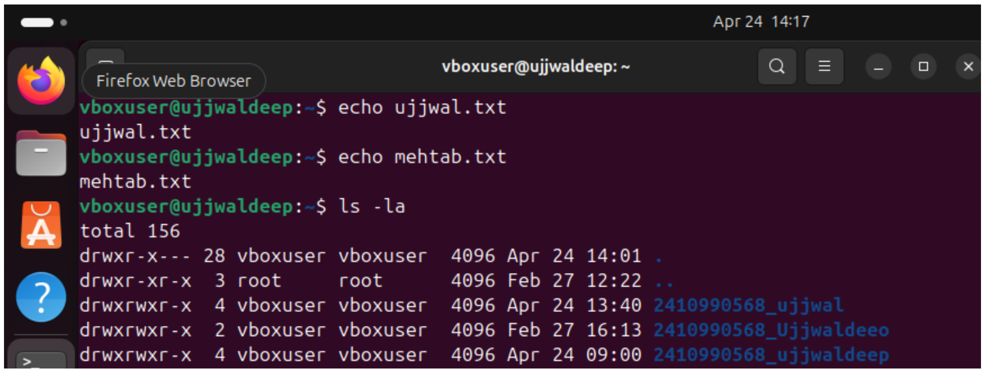


```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ reboot
```

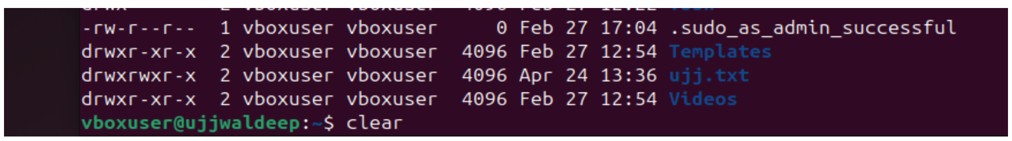
17. clear (Clear Terminal) : The clear command clears the terminal screen**.**

Syntax:
Clear

Common Options:
none



```
vboxuser@ujjwaldeep:~$ echo ujjwal.txt
ujjwal.txt
vboxuser@ujjwaldeep:~$ echo mehtab.txt
mehtab.txt
vboxuser@ujjwaldeep:~$ ls -la
total 156
drwxr-x--- 28 vboxuser vboxuser  4096 Apr 24 14:01 .
drwxr-xr-x  3 root     root      4096 Feb 27 12:22 ..
drwxrwxr-x  4 vboxuser vboxuser  4096 Apr 24 13:40 2410990568_ujjwal
drwxrwxr-x  2 vboxuser vboxuser  4096 Feb 27 16:13 2410990568_Ujjwaldeeo
drwxrwxr-x  4 vboxuser vboxuser  4096 Apr 24 09:00 2410990568_ujjwaldeep
```



```
-rw-r--r--  1 vboxuser vboxuser     0 Feb 27 17:04 .sudo_as_admin_successful
drwxr-xr-x  2 vboxuser vboxuser  4096 Feb 27 12:54 Templates
drwxrwxr-x  2 vboxuser vboxuser  4096 Apr 24 13:36 ujj.txt
drwxr-xr-x  2 vboxuser vboxuser  4096 Feb 27 12:54 Videos
vboxuser@ujjwaldeep:~$ clear
```

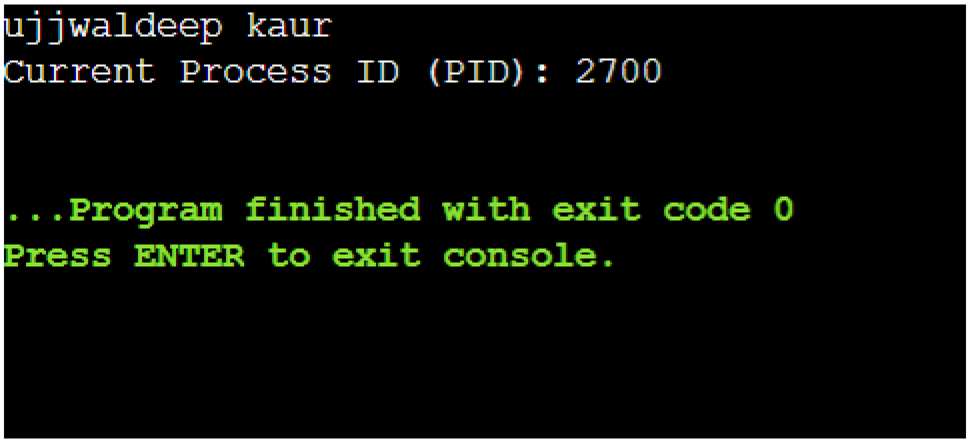
Practical 5

Aim: Implement Process concepts using C language by Printing process Id

CODE:

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

int main() {
printf("Ujjwaldeepkaur");
pid_t pid = getpid();

printf("Current Process ID (PID): %d\n", pid);

return 0;
}
```

Output :



```
ujjwaldeep kaur
Current Process ID (PID): 2700


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 6

Aim: Implement FCFS, SJF scheduling algorithms in C language.

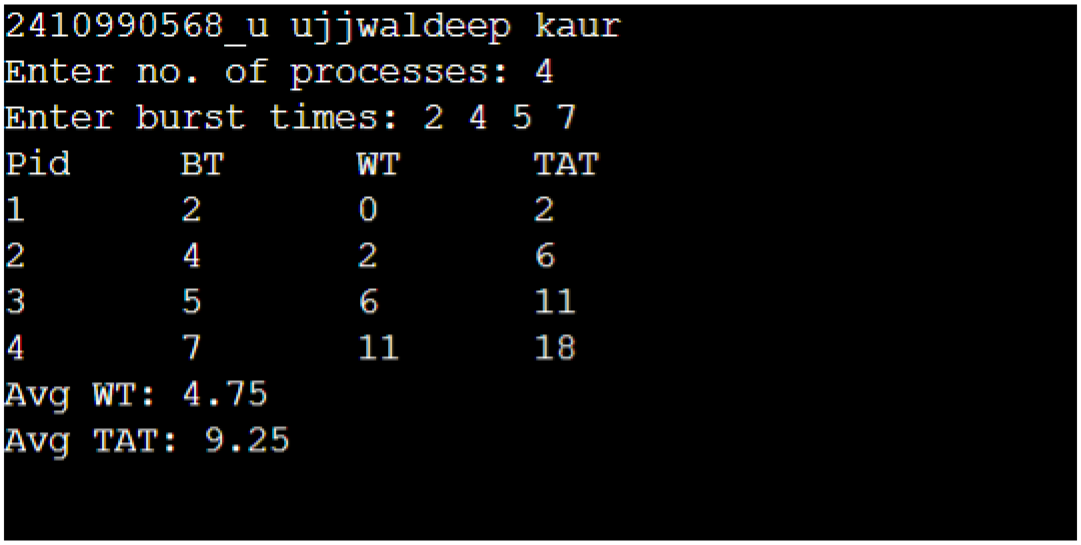
Solution:

FCFS ALGORITHM WHEN ARRIVAL TIME IS ZERO

CODE–

```
#include <stdio.h>
int main() {
    int n, i, BT[10], WT[10], TAT[10];
    float total_wt = 0, total_tat = 0;

    printf("2410990568_Ujjwaldeepkaur\n");
    printf("Enter no. of processes: ");
    scanf("%d", &n);

    printf("Enter burst times: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &BT[i]);
    }


    WT[0] = 0;
    for (i = 1; i < n; i++) {
        WT[i] = BT[i-1] + WT[i-1];
        total_wt += WT[i];
    }


  for (i = 0; i < n; i++) {
        TAT[i] = BT[i] + WT[i];
        total_tat += TAT[i];
    }
    printf("Pid\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\n", i+1, BT[i], WT[i], TAT[i]);
    }
    printf("Avg WT: %.2f\n", total_wt / n);
    printf("Avg TAT: %.2f\n", total_tat / n);
    return 0;
}
```

Output -



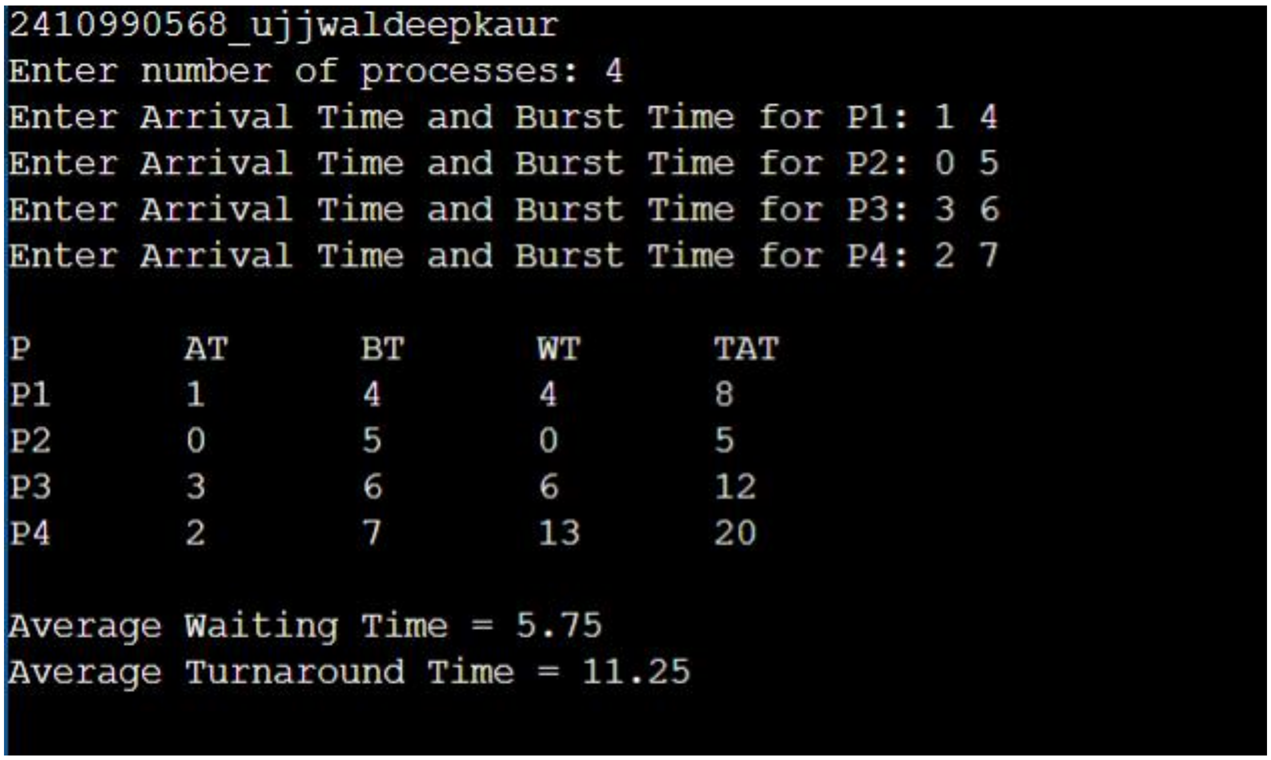
```
2410990568_u ujjwaldeep kaur
Enter no. of processes: 4
Enter burst times: 2 4 5 7
Pid     BT      WT      TAT
1       2       0       2
2       4       2       6
3       5       6       11
4       7       11      18
Avg WT: 4.75
Avg TAT: 9.25
```

SJF ALGORITHM (NON - PREEMPTIVE)

CODE –

```
#include <stdio.h>
int main() {
    int n, i, j, min, time = 0, completed = 0;
    int at[20], bt[20], ct[20], tat[20], wt[20], pid[20];
    int is_done[20] = {0};
    float avg_wt = 0, avg_tat = 0;
    printf("2410990568_Ujjwaldeepkaur\n");
    printf("Enter number of processes: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("Enter Arrival Time and Burst Time for P%d: ", i + 1);
        scanf("%d%d", &at[i], &bt[i]);
        pid[i] = i + 1;



    }
    while (completed < n) {
        min = -1;
        for (i = 0; i < n; i++) {
            if (at[i] <= time && !is_done[i]) {
                if (min == -1 || bt[i] < bt[min]) {
                    min = i;
                }
            }
        }
        if (min == -1) {
            time++;
        } else {
            wt[min] = time - at[min];
            time += bt[min];
            ct[min] = time;
            tat[min] = ct[min] - at[min];
       is_done[min] = 1;

            completed++;
            avg_wt += wt[min];
            avg_tat += tat[min];
        }
    }
    printf("\nP\tAT\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\n", pid[i], at[i], bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time = %.2f\n", avg_wt / n);
    printf("Average Turnaround Time = %.2f\n", avg_tat / n);


    return 0;
}
```

<u>Output –</u>



```
2410990568_ujjwaldeepkaur
Enter number of processes: 4
Enter Arrival Time and Burst Time for P1: 1 4
Enter Arrival Time and Burst Time for P2: 0 5
Enter Arrival Time and Burst Time for P3: 3 6
Enter Arrival Time and Burst Time for P4: 2 7

P       AT      BT      WT      TAT
P1      1       4       4       8
P2      0       5       0       5
P3      3       6       6       12
P4      2       7       13      20

Average Waiting Time = 5.75
Average Turnaround Time = 11.25
```

Practical 7

Aim: Implement priority scheduling, and RR scheduling algorithms in C language.

Solution:

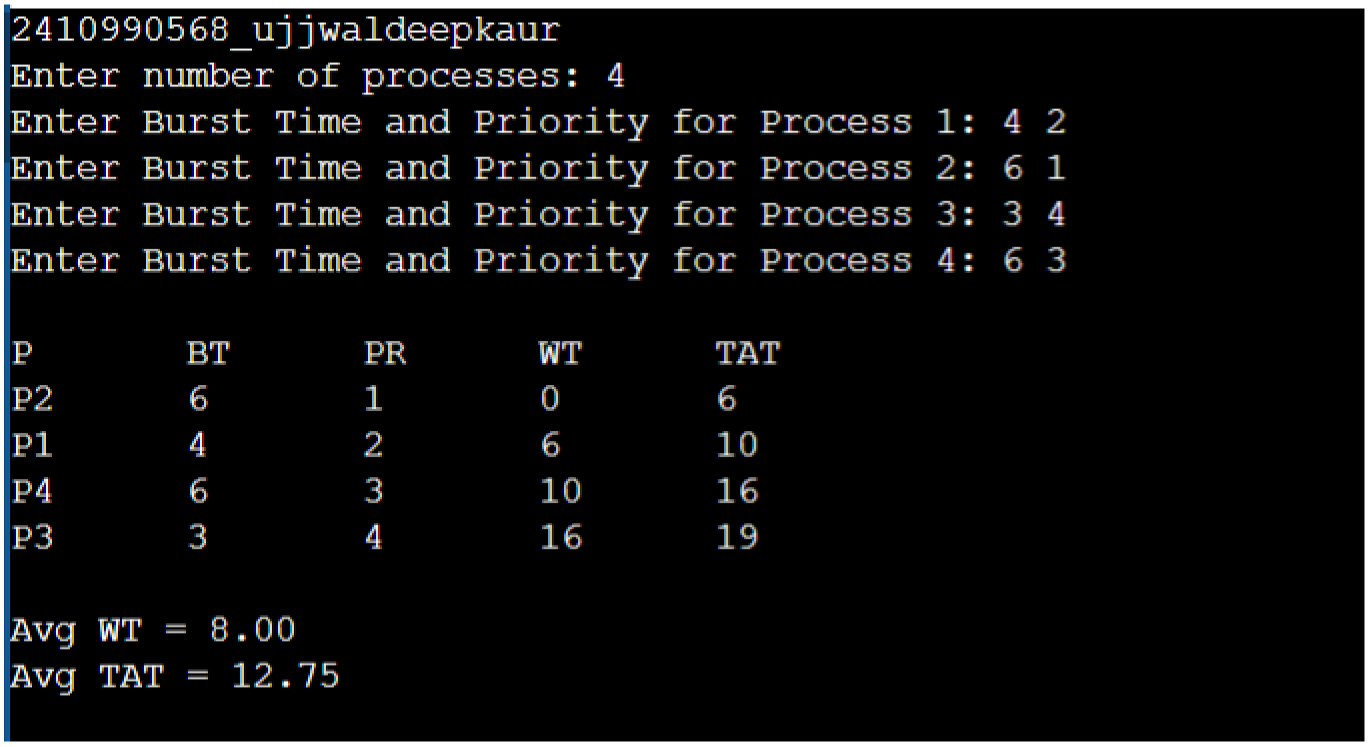
PRIORITY SCHEDULING (NON - PREEMPTIVE)

CODE –

```
#include <stdio.h>

int main() {
    int n, i, j, temp;
    int pid[20], bt[20], pr[20], wt[20], tat[20];
    float avg_wt = 0, avg_tat = 0;
    printf("2410990568_Ujjwaldeepkaur\n");
    printf("Enter number of processes: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("Enter Burst Time and Priority for Process %d: ", i + 1);
        scanf("%d%d", &bt[i], &pr[i]);
        pid[i] = i + 1;
    }
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (pr[i] > pr[j]) {
                temp = pr[i]; pr[i] = pr[j]; pr[j] = temp;
                temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = pid[i]; pid[i] = pid[j]; pid[j] = temp;
            }
        }
    }
    wt[0] = 0;
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
    }

    for (i = 0; i < n; i++) {
        tat[i] = wt[i] + bt[i];
```

```
 avg_wt += wt[i];
        avg_tat += tat[i];
    }
    printf("\nP\tBT\tPR\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\n", pid[i], bt[i], pr[i], wt[i], tat[i]);



    }
    printf("\nAvg WT = %.2f\n", avg_wt / n);
    printf("Avg TAT = %.2f\n", avg_tat / n);
    return 0;
}
```

Output –



```
2410990568_ujjwaldeepkaur
Enter number of processes: 4
Enter Burst Time and Priority for Process 1: 4 2
Enter Burst Time and Priority for Process 2: 6 1
Enter Burst Time and Priority for Process 3: 3 4
Enter Burst Time and Priority for Process 4: 6 3

P       BT      PR      WT      TAT
P2      6       1       0       6
P1      4       2       6       10
P4      6       3       10      16
P3      3       4       16      19

Avg WT = 8.00
Avg TAT = 12.75
```

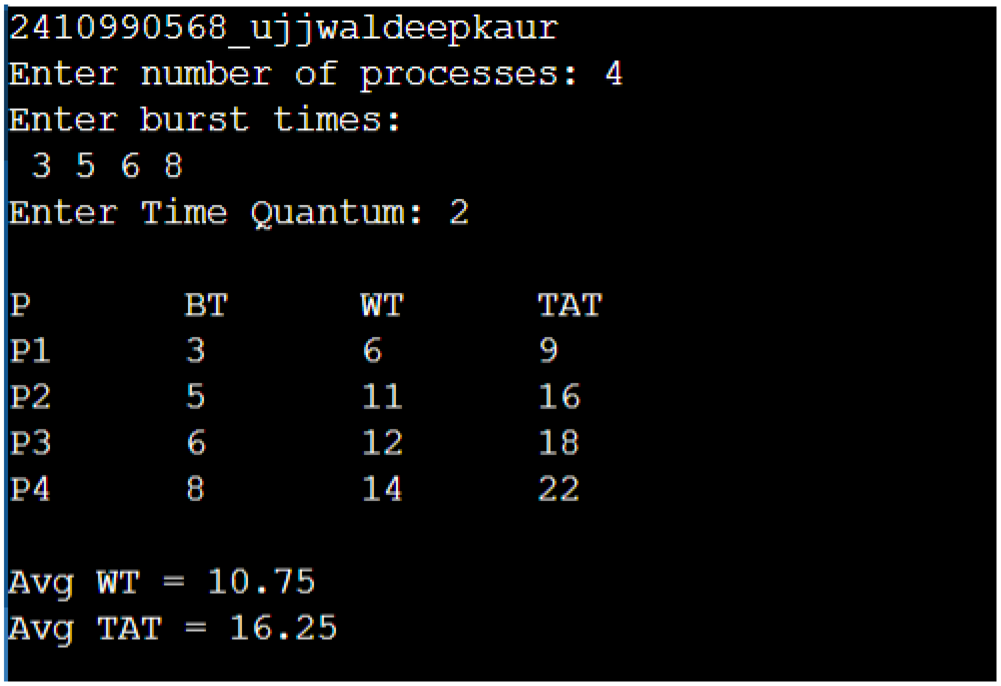
ROUND ROBIN SCHEDULING

CODE:

```
#include <stdio.h>
int main() {
    int n, i, tq, time = 0, remain;
    int bt[20], rt[20], wt[20], tat[20];
    float avg_wt = 0, avg_tat = 0;

    printf("2410990568_Ujjwaldeepkaur\n");
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");

     for (i = 0; i < n; i++) {
        scanf("%d", &bt[i]);
        rt[i] = bt[i];
    }
    printf("Enter Time Quantum: ");
    scanf("%d", &tq);
    remain = n;
    while (remain > 0) {
        for (i = 0; i < n; i++) {
            if (rt[i] > 0) {
                if (rt[i] <= tq) {
                    time += rt[i];
                    tat[i] = time;
                    wt[i] = tat[i] - bt[i];
                    rt[i] = 0;
                    remain--;
                } else {
                    rt[i] -= tq;
                    time += tq;
                }
            }
        }
    }
    for (i = 0; i < n; i++) {
        avg_wt += wt[i];
        avg_tat += tat[i];
    }
```

```
   printf("\nP\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }
    printf("\nAvg WT = %.2f\n", avg_wt / n);
    printf("Avg TAT = %.2f\n", avg_tat / n);
    return 0;
}
```

Output –



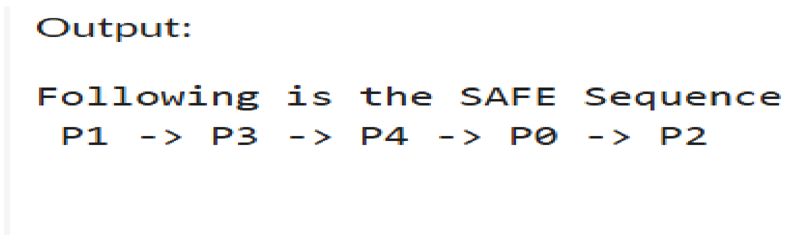
```
2410990568_ujjwaldeepkaur
Enter number of processes: 4
Enter burst times:
 3 5 6 8
Enter Time Quantum: 2

P       BT      WT      TAT
P1      3       6       9
P2      5       11      16
P3      6       12      18
P4      8       14      22

Avg WT = 10.75
Avg TAT = 16.25
```

Practical 8

Aim: Implement deadlock detection in C. (Banker's Algorithm)

CODE:

```
#include <stdio.h>
int main()
{
   int n, m, i, j, k;
   n = 5;                          // Number of processes
   m = 3;                          // Number of resources
   int alloc[5][3] = {{0, 1, 0},  // P0 // Allocation Matrix
                      {2, 0, 0},  // P1
                      {3, 0, 2},  // P2
                      {2, 1, 1},  // P3
                      {0, 0, 2}}; // P4

   int max[5][3] = {{7, 5, 3},  // P0 // MAX Matrix
                    {3, 2, 2},  // P1
                    {9, 0, 2},  // P2
                    {2, 2, 2},  // P3
                    {4, 3, 3}}; // P4

   int avail[3] = {3, 3, 2}; // Available Resources

   int f[n], ans[n], ind = 0;
   for (k = 0; k < n; k++)
   {
      f[k] = 0;
   }
   int need[n][m];
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
         need[i][j] = max[i][j] - alloc[i][j];
   }
   int y = 0;
   for (k = 0; k < 5; k++)
   {
      for (i = 0; i < n; i++)
      {
         if (f[i] == 0)  {
            int flag = 0;
            for (j = 0; j < m; j++)
            {
                if (need[i][j] > avail[j])
                {

                    flag = 1;
                    break;   }
            }
            if (flag == 0)
            {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y];
                f[i] = 1;   }
        }
    }
}
int flag = 1;
for (int i = 0; i < n; i++)
{
    if (f[i] == 0)
    {
        flag = 0;
        printf("The following system is not safe");
        break;
    }
}
if (flag == 1)
{
    printf("Following is the SAFE Sequence\n");
    for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
    printf(" P%d", ans[n - 1]);
}
return (0);  }
```



```
Output:

Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
```

Practical 9

Aim: File system: Introduction to File system, File system Architecture and File Types.

Solution:
Theory: A file system is a method used by operating systems to organize, store, and manage files on storage devices like hard drives, SSDs, and flash memory. It determines how data is structured, accessed, and secured, ensuring efficient retrieval and storage.

File System Architecture

A file system consists of multiple layers:
1. Application Layer – Users interact with files through applications.
2. Logical File System – Manages metadata, access permissions, and directory structure.
3. File Control Layer– Handles file operations like open, read, write, and close.
4. Storage Layer – Allocates disk space efficiently.
5. Physical Storage – The actual storage medium, such as HDDs or SSDs.

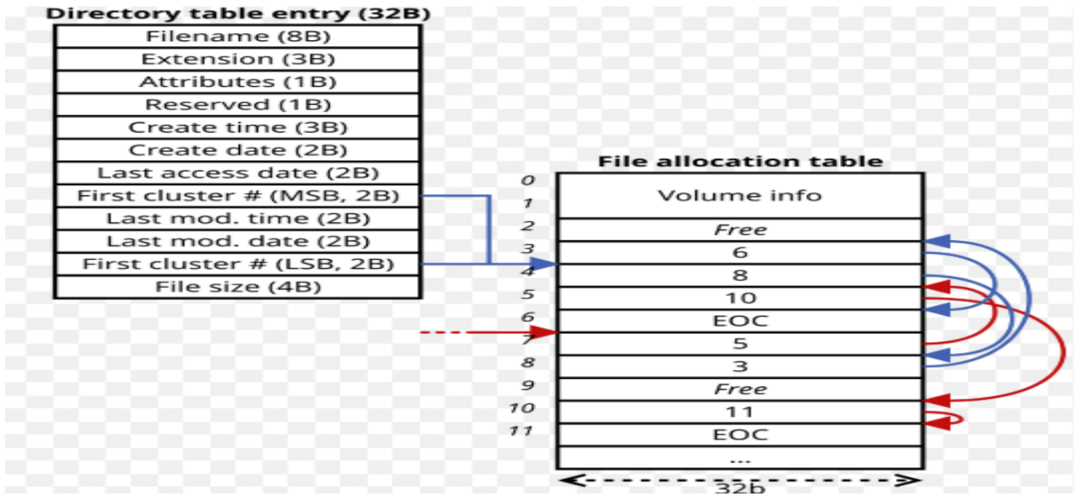
File Types

Different types of files exist within a file system, including:
. Regular Files – Documents, images, videos, executables.
. Directory Files – Structures that store multiple files.
. System Files – OS-related files necessary for system operation.
. Device Files – Represent hardware components like drives and printers.
Common file systems include NTFS, FAT32, ext4, HFS+, each optimized for specific environments.
Files: Files are the basic units of storage in a file system. They contain user data and are organized into directories.
Directories: Directories (or folders) are containers used to organize and manage files. They can contain other directories and files.
Metadata: Metadata includes information about files and directories such as file name, file size, file type, ownership, permissions, timestamps (creation time, last modified time), and pointers to data blocks.
File Access Methods: File systems provide methods to access and manipulate files, including read, write, create, delete, open, close, and seek operations.

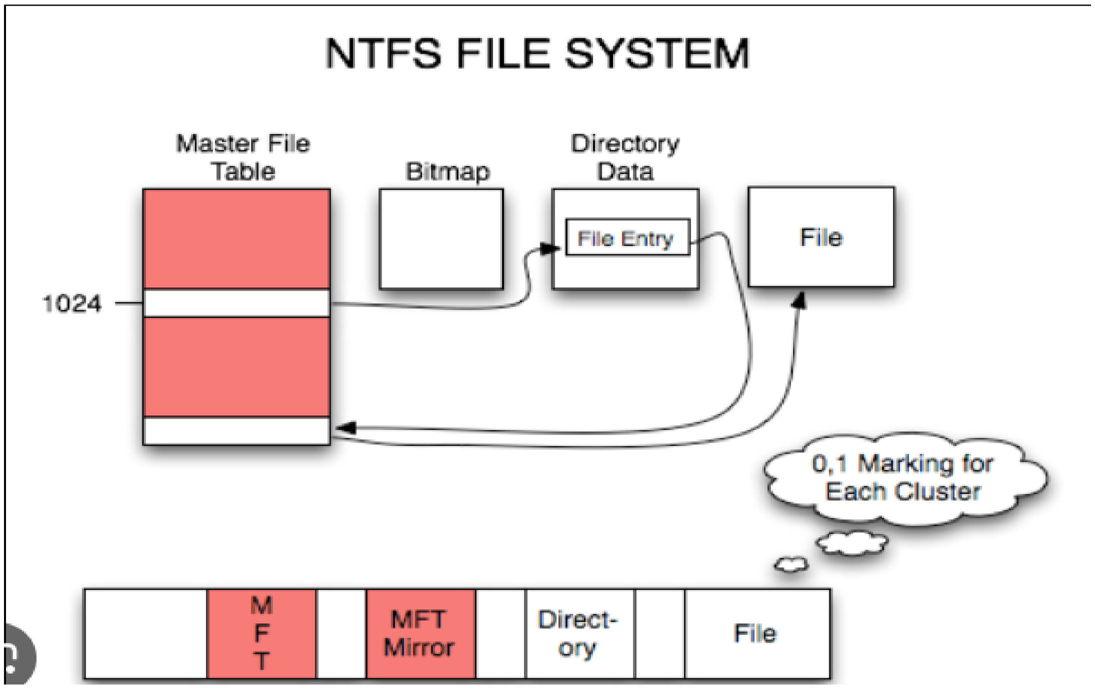
File System Types:

Disk-Based File Systems:

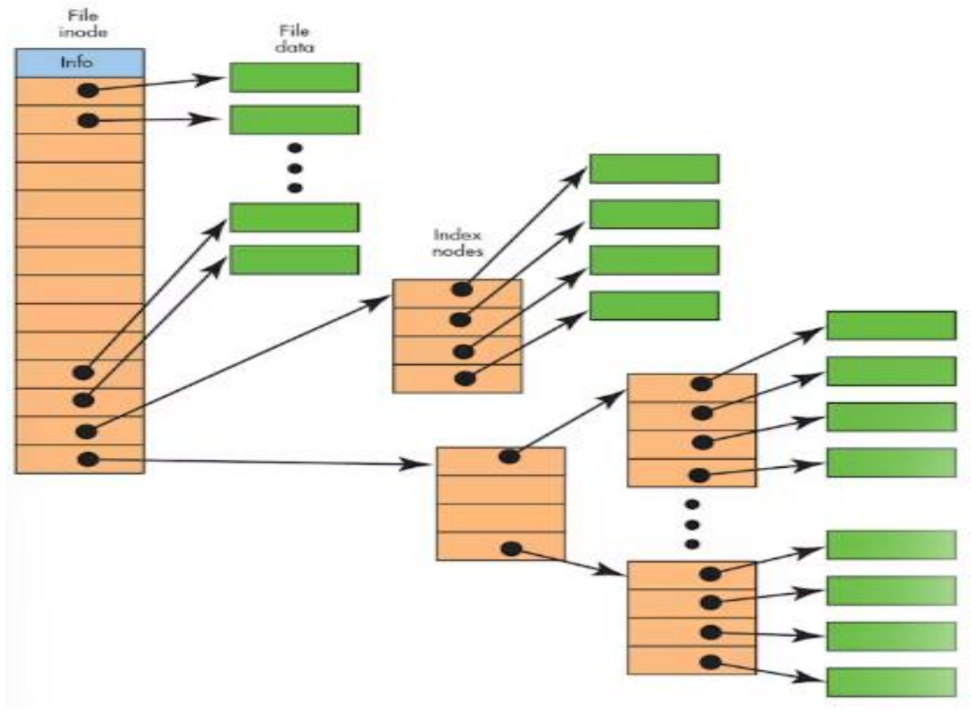
FAT (File Allocation Table): Used primarily with older versions of Windows. Simple structure but limited features.



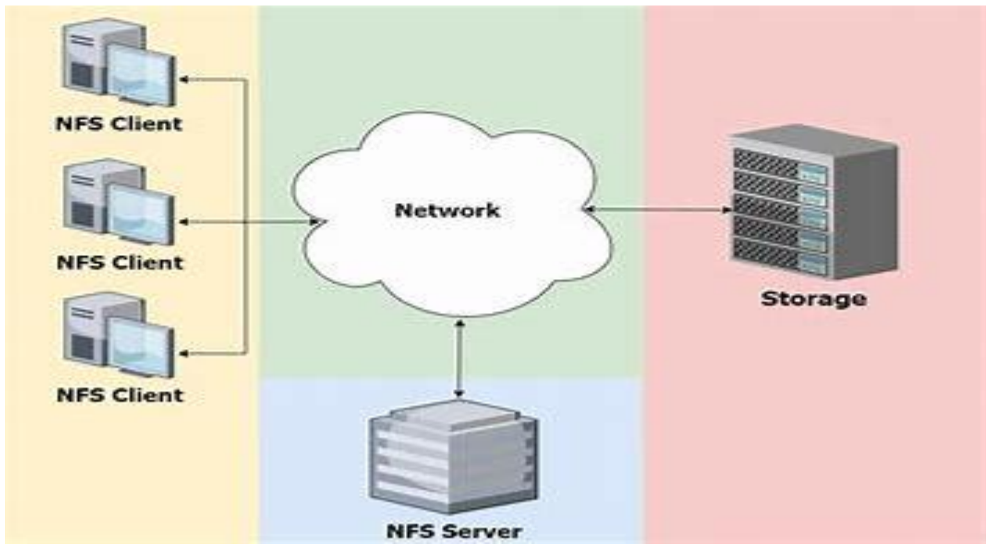
NTFS (New Technology File System): Developed by Microsoft, offering features like file compression, encryption, and support for large file sizes and volumes.



ext4 (Fourth Extended File System): Commonly used in Linux distributions, providing features like journaling, support for large file systems, and efficient handling of large files.
Network File Systems:



NFS (Network File System): Allows file sharing and access across a network, commonly used in Unix/Linux environments.



CIFS (Common Internet File System): Allows file sharing over the Internet, commonly used in Windows environments.

Distributed File Systems:

- **HDFS (Hadoop Distributed File System):** Designed to store large volumes of data across multiple machines in a Hadoop cluster.
- **Ceph File System:** Provides a distributed file system with features like fault tolerance, scalability, and performance
- Specialized File Systems:
- ISO 9660: Standard file system used for optical disc media such as CDs and DVDs.
- **exFAT**: Extended File Allocation Table file system designed for flash drives and external storage devices.

  File System Architecture:
- **Disk Layout:**
  The disk is divided into blocks or sectors. File systems manage these blocks efficiently for data storage and retrieval.
- **File Allocation:**
  File systems allocate storage space for files, manage free space, and track the location of file data blocks.

Metadata Management:
Each file and directory has associated metadata stored in an inode (index node). Metadata includes file attributes and pointers to data blocks.

File Operations:
File systems provide operations to manipulate files such as create, read, write, delete, open, close, and seek.

Directory Structure:
Directories organize files into a hierarchical structure, making it easy to navigate and locate files.

Security and Permissions:
File systems enforce access control through permissions (read, write, execute) assigned to users and groups.

Error Handling:
File systems use techniques like journaling and checksums to ensure data integrity and recover from system failures.
Understanding file system concepts and types is essential for efficient data management and storage in computing systems. Different file systems offer various features and optimizations tailored to specific use cases and environments. The choice of a file system depends on factors such as the operating system, storage requirements, performance considerations, and compatibility with existing infrastructure

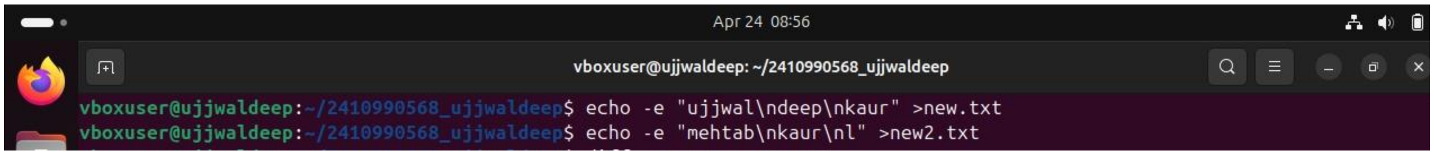
Practical 10

Title: Implement File System commands: Comparing Files using diff, cmp, comm

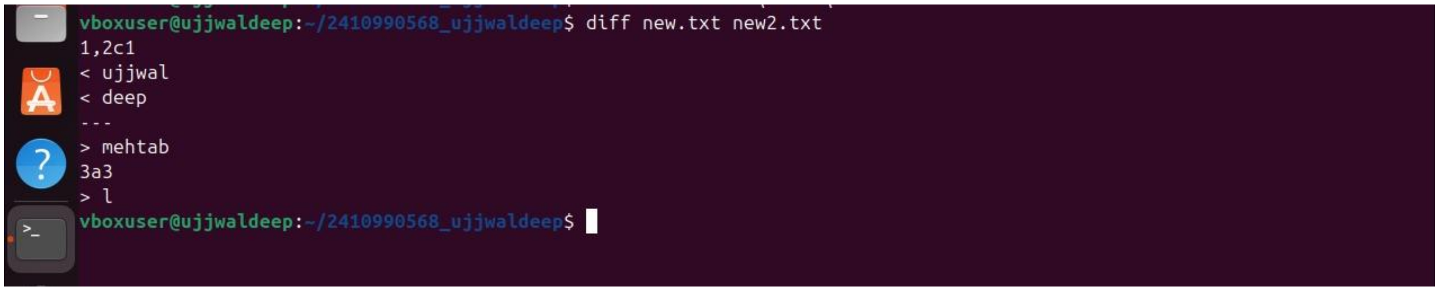
Solution:

1. **diff** – Compare Two Files Line-by-Line

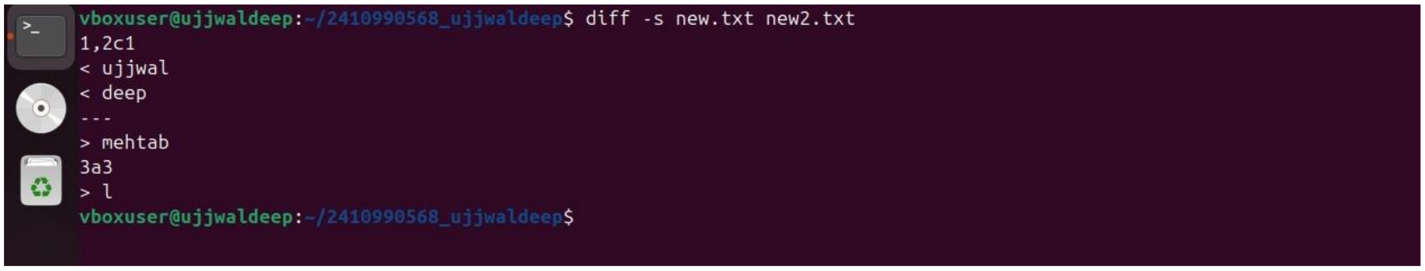
   Common options:
   • diff file1 file2
   • -s: Report if files are the same
   • -q: Report only if files differ



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ echo -e "ujjwal\ndeep\nkaur" >new.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ echo -e "mehtab\nkaur\nl" >new2.txt
```



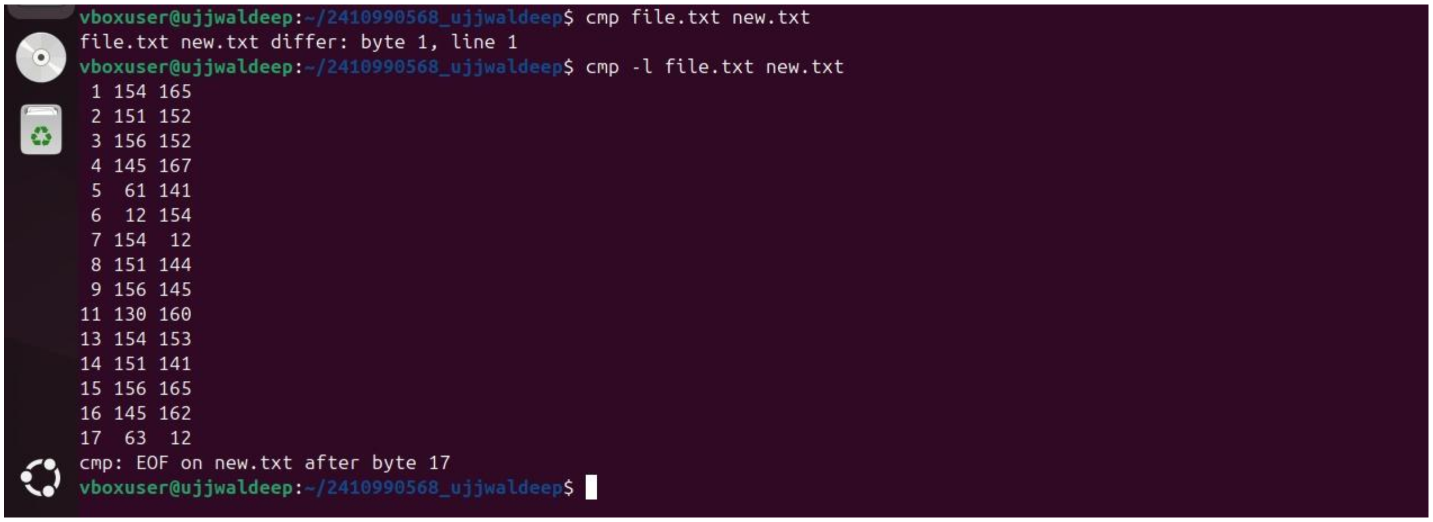
```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ diff new.txt new2.txt
1,2c1
< ujjwal
< deep
---
> mehtab
3a3
> l
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ diff -s new.txt new2.txt
1,2c1
< ujjwal
< deep
---
> mehtab
3a3
> l
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

2. **cmp** – Byte-by-Byte Comparison

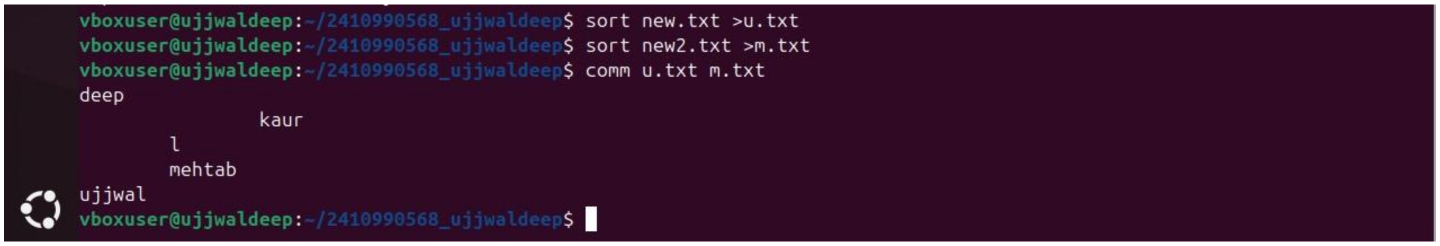
   Common options:
   • cmp file1 file2
   • -l: Print byte differences
   • -s: Silent (just exit code)



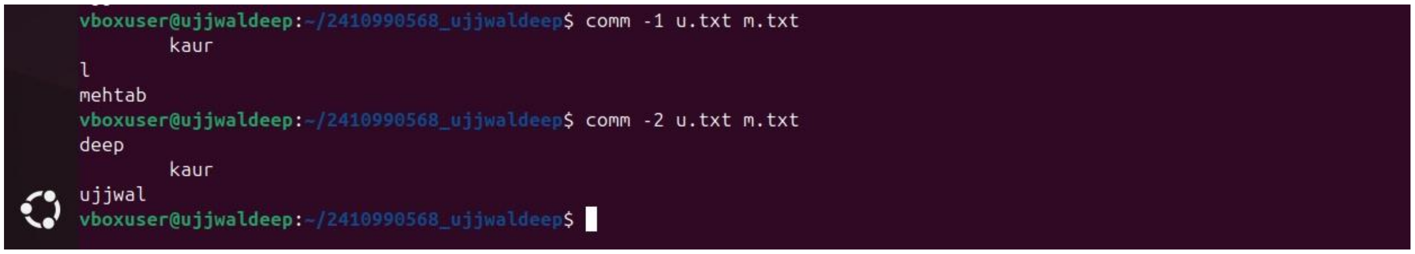
```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ cmp file.txt new.txt
file.txt new.txt differ: byte 1, line 1
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ cmp -l file.txt new.txt
 1 154 165
 2 151 152
 3 156 152
 4 145 167
 5  61 141
 6  12 154
 7 154  12
 8 151 144
 9 156 145
11 130 160
13 154 153
14 151 141
15 156 165
16 145 162
17  63  12
cmp: EOF on new.txt after byte 17
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```

3. **comm** – Compare Sorted Files Line-by-Line

   Common options:
   • comm file1 file2
   • -1: suppress column 1 (lines only in file1)
   • -2: suppress column 2
   • -3: suppress column 3 (common lines)



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ sort new.txt >u.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ sort new2.txt >m.txt
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ comm u.txt m.txt
deep
                kaur
        l
        mehtab
ujjwal
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```



```
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ comm -1 u.txt m.txt
        kaur
l
mehtab
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$ comm -2 u.txt m.txt
deep
        kaur
ujjwal
vboxuser@ujjwaldeep:~/2410990568_ujjwaldeep$
```