**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Practical Name** | **Teacher Sign** |
| **1** | 1. **Installation: Configuration & Customizations of Linux** 2. **Introduction to GCC compiler: Basics of GCC, Compilation of program, Execution of program.** 3. **Time stamping in Linux.** 4. **Automating the execution using Make file.** |  |
| **2** | **Implement the commands that is used for creating and manipulating files: cat, cp, mv, rm, ls and its options, touch and their options, which is, where is, what is** |  |
| **3** | **Implement Directory oriented commands: cd, pwd, mkdir, rmdir, Comparing Files using diff, cmp, comm.** |  |
| **4** | **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** |  |
| **5** | **Write a program and execute the same to demonstrate how to use terminal commands in C program (using system() function)** |  |
| **6** | **Write a program to implement process concepts using C language by printing process Id.** |  |
| **7** | **Write a program to create and execute process using fork() and exec() system calls.** |  |
| **8** | **Write a C program to implement FCFS (First Come First Serve) and SJF (Shortest Job First) scheduling algorithms.** |  |
| **9** | **Write a C program to implement Priority Scheduling and RR (Round-Robin) scheduling algorithms.** |  |
| **10** | **Write a program to demonstrate the concept of deadlock in C** |  |
| **11** | **Introduction to File system, file system types and architecture.** |  |

**Experiment 1**

Installation: Configuration & Customizations of Linux.

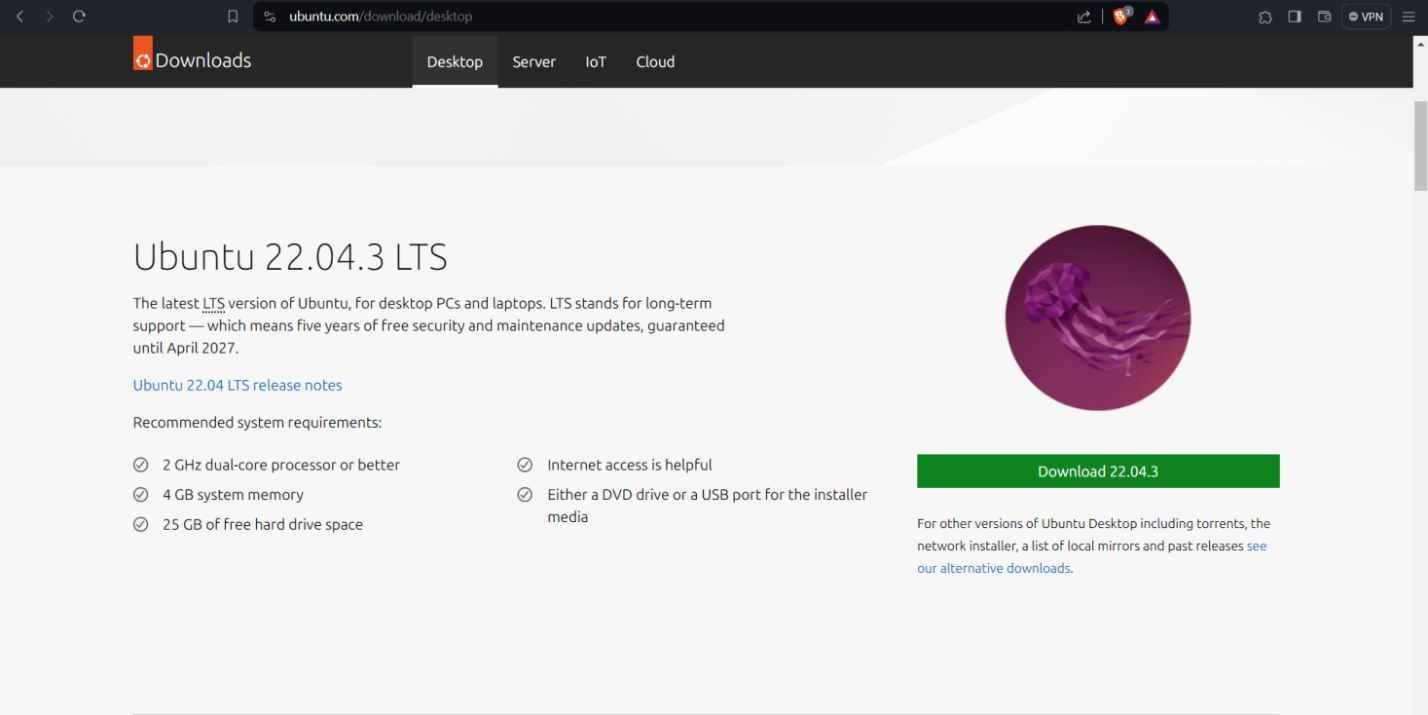
**Step 1: Download VirtualBox (VM)**

Go to <https://www.virtualbox.org/> and download VM for Windows.

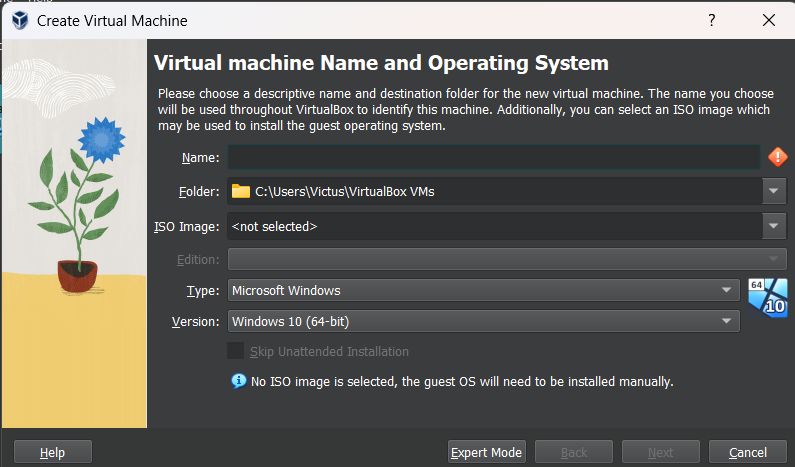
****

**Step 2: Download Ubuntu Linux ISO File.**

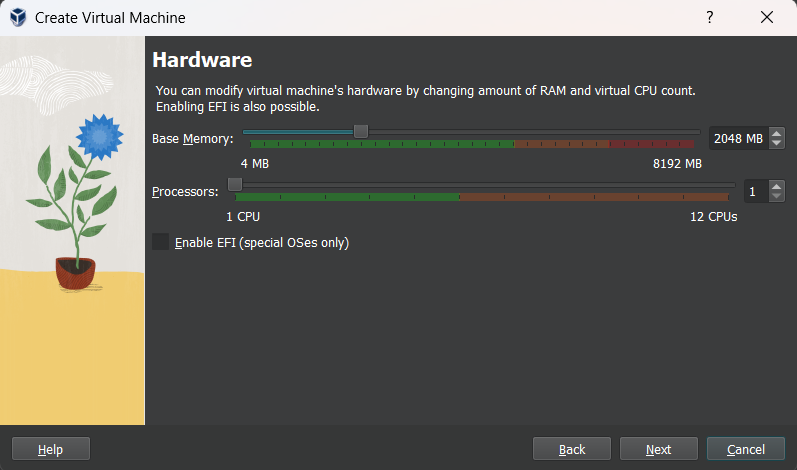
Go to <https://ubuntu.com/download> and download ISO file for Linux**.**

****

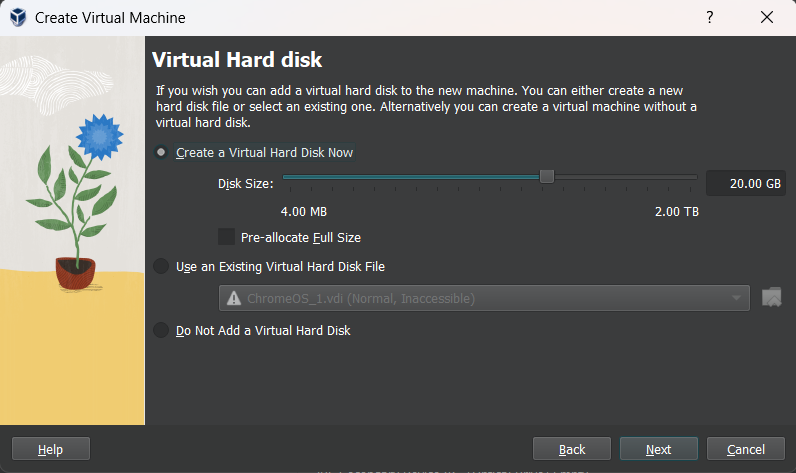
**Step 3: Create a new Virtual Machine.**

****

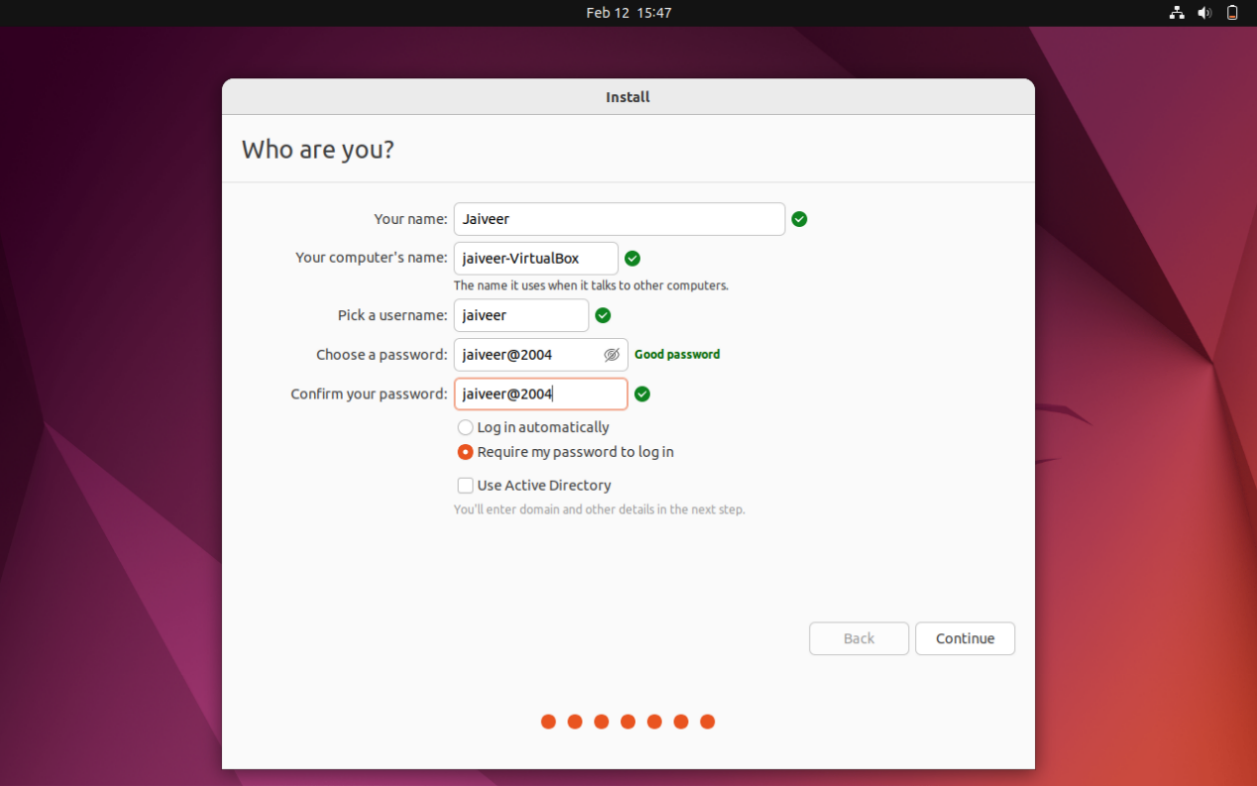
**Step 4: Allocate RAM Memory and No of Processors.**

****

**Step 5: Create a Virtual Hard Disk.**

****

**Step 6: Install Linux.**

****

Linux is Now Installed on your PC using VirtualBox VM.

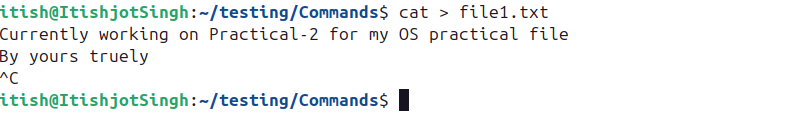
****

**Experiment 2**

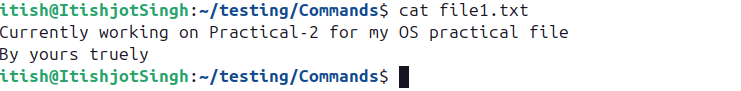
Implement the commands that are used for creating and manipulating files: cat, cp, mv, rm, ls and its options, touch and its options, which is, where is, what is.

**CAT COMMANDS**

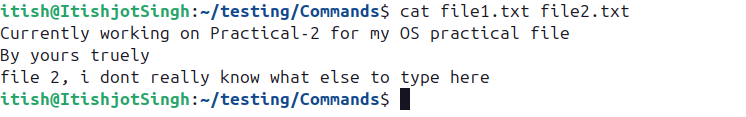
* **Cat > <file1>:** This command gives us the privilege to create the file and write it just after creating the file in the next line. To exit after writing into the file, press enter and press CTRL + C.

****

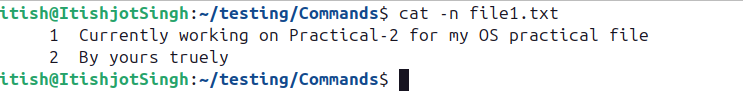
* **Cat <file1>:** This command is used to view the file contents.



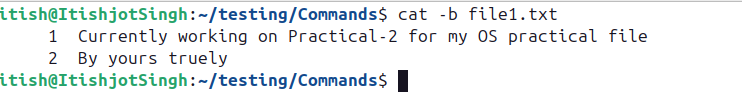
* **Cat <file1> <file2>:** This command is used to view both the file contents at the same time.



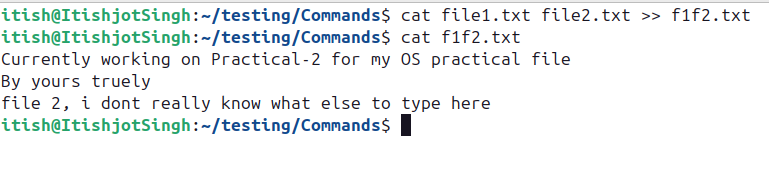
* **Cat -n <file1>:** This command provides numbering to the lines in the file



* **Cat -b <file1>:** This command will not give numbering to the blank lines.

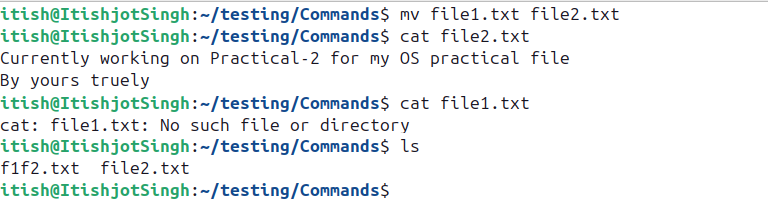


* **Cat <file1> <file2> >> <file3>:** This command combines the contents of both files into one file

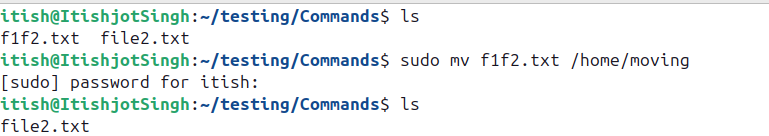


**MOVE COMMANDS**

* **mv file1.txt file2.txt:** This command moves the file1’s contents into file2 and deletes the file1. It will replace the contents of file2 if present.

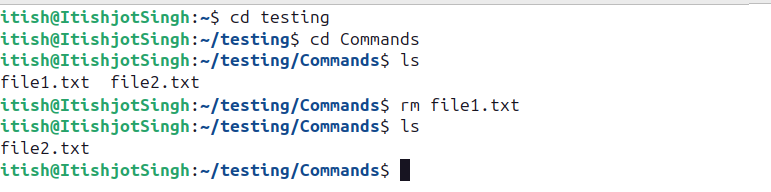


* **mv file\_name path\_of\_directory:** This command moves the file to another directory.

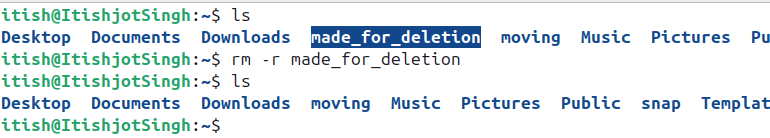


**REMOVE COMMANDS**

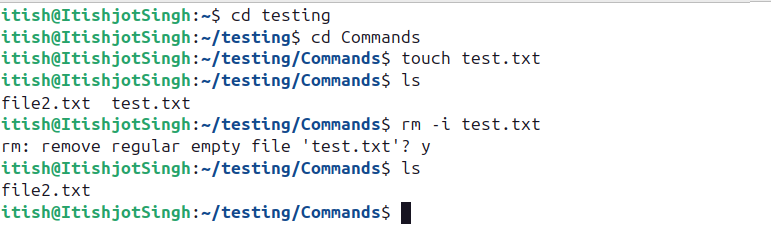
* **rm file\_name:** This command removes (deletes) the file\_name from the directory.



* **rm -r directory\_path:** This command deletes the directory.

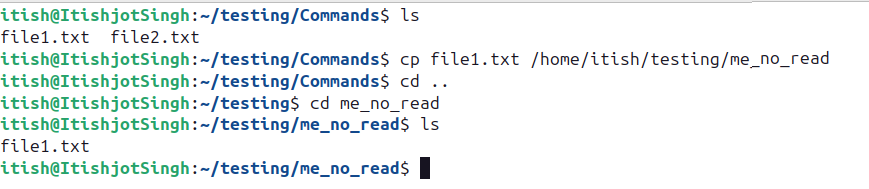


* **rm -i <file>.extension:** This command asks for user’s permission before deleting the file.



**COPY COMMANDS**

* **cp file1.txt directory\_path:** This command copies the file to another directory.



* **cp -r <directory1\_path> <directory2\_path>:** This command copies the whole directory to another directory

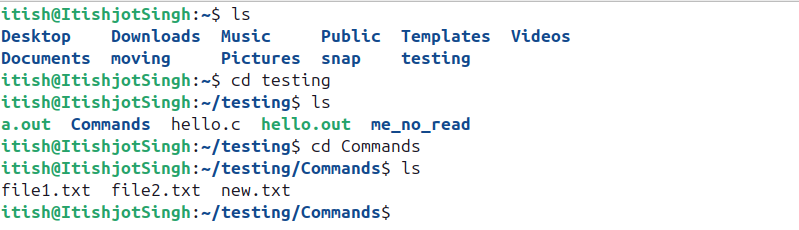


**Practical 3**

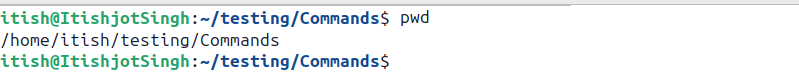
Implement Directory oriented commands: cd, pwd, mkdir, rmdir, Comparing Files using diff, cmp, comm.

**DIRECTORY ORIENTED COMMANDS**

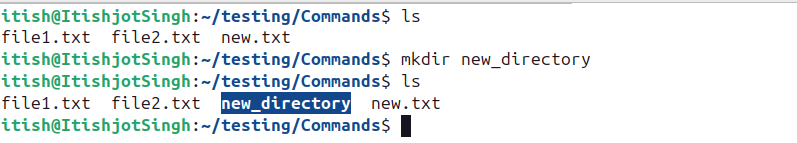
* **cd** (Change Directory) **<name\_of\_the\_directory>:** Used to navigate between different directories.

****

* **pwd** (Print Working Directory)**:** Displays the absolute path of the current directory.

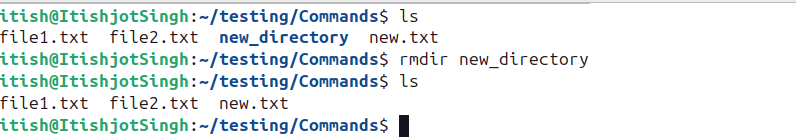


* **mkdir** (Make Directory) **<directory\_name>:** Creates a new directory.

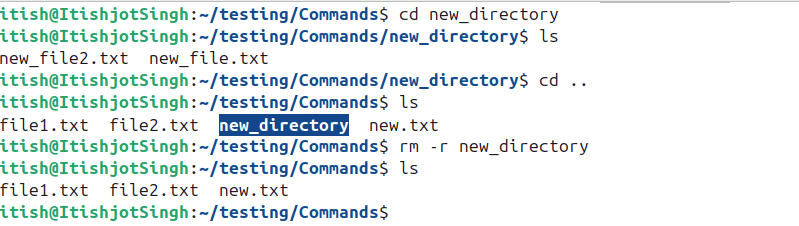


* **rmdir** (Remove Directory) **<directory\_name>:**

1. Removes an empty directory.

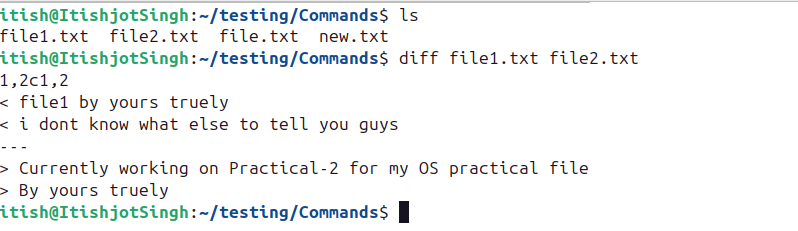
****

1. Forcefully removes a directory even if it contains files, using “rm -r <directory\_name>” instead of “rmdir <directory\_name>”.

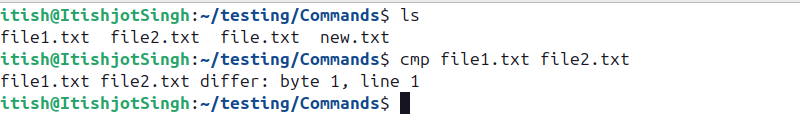
****

**COMPARING ORIENTED COMMANDS**

* **diff <file1> <file2>:** Compares two text files line by line and shows the differences.

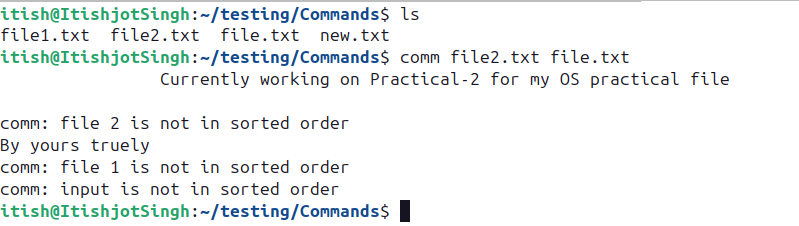


* **cmp <file1> <file2>:** This function compares files byte by byte and stops at the first difference, if the files are identical, it gives output along “No output (silent)”

****

* **comm <file1> <file2>:** Compares two sorted files and shows:

1. **1st column:** Common lines in both the lines.
2. **2nd column:** Lines only in <file1>.
3. **3rd column:** Lines only in <file2>.

****

It shows the common line at first, in the center, while the remaining lines after the common line.

**Program 4**

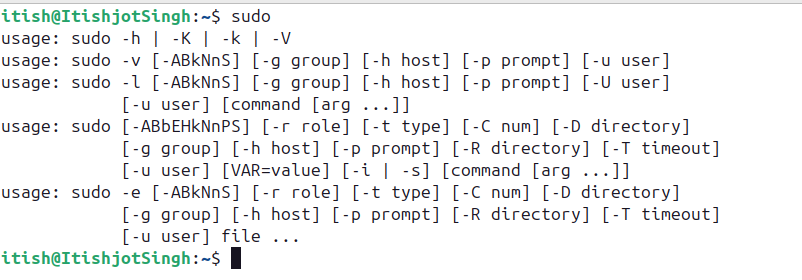
Implement the basic and user status commands like: su, sudo, man, help, history, who, whoami, id, uname, uptime, free, tty, cal, date, hostname, reboot, clear.

**Solution**

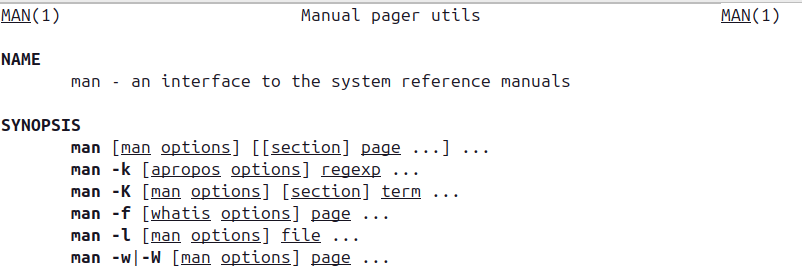
1. **su** (Switch User): Allows a user to switch to another user account, if they have the required permissions.



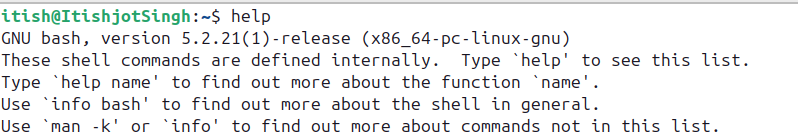
2. **sudo** (Superuser Do): Enables a permitted user to execute a command with the security privileges of another user (commonly the superuser or root).



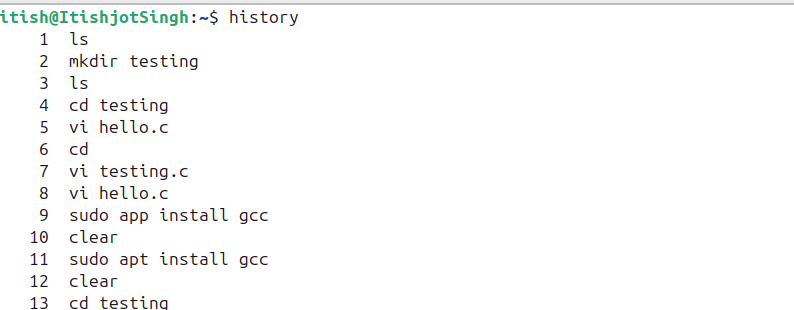
3. **man** (Manual Pages): Displays the manual pages for a specified command, providing detailed information on its usage, options, and examples.



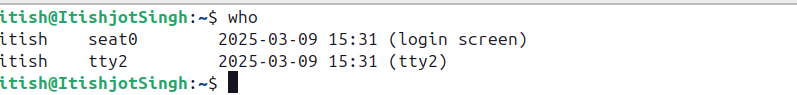
4. **help**: Displays help information about built-in shell commands, typically specific to the shell environment.



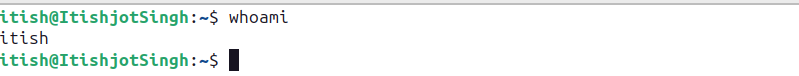
5. **history**: Displays a list of previously executed commands, allowing users to view and rerun commands from their history.



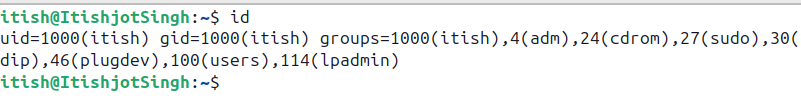
6. **who**: Displays information about users currently logged in, including their username, terminal, and login time.



7. **whoami**: Outputs the username of the current effective user.



8. **id**: Displays user identity information, including user and group IDs.

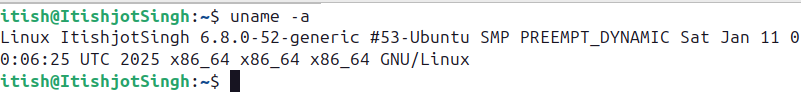


9. **uname**: There are several sub commands of uname:

* **uname**: Displays Operating System name.



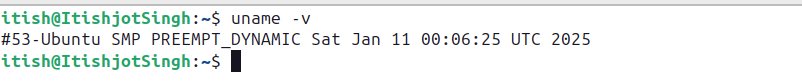
* **uname -a**: Provides basic system information such as system name, kernel version, and architecture.



* **uname -s**: Displays Kernel Name.



* **uname -v**: Displays version of the OS.



* **uname -r**: Displays Kernel version.



* **uname -p**: Displays the processor type or architecture of the system.

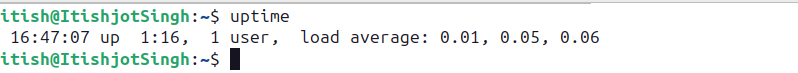


* **uname -o**: Displays the operating system name.



10. **uptime**: There are several sub commands of uptime:

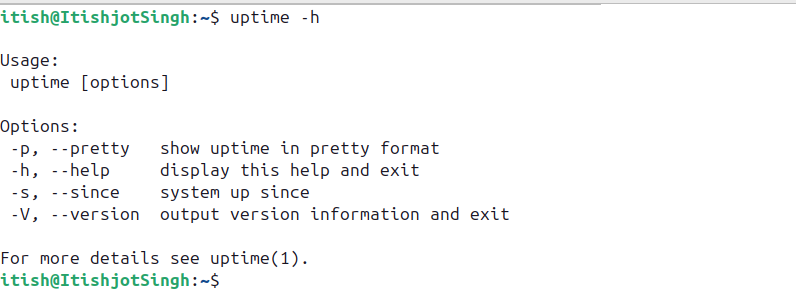
* **uptime** :Displays how long the system has been running, as well as load averages.



* **uptime -p**: Displays the duration since the system was last booted.



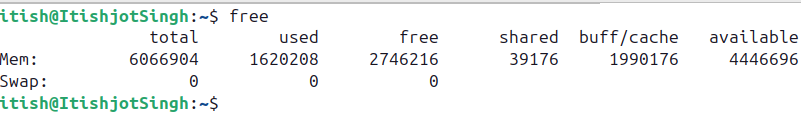
* **uptime -h**: Displays help box of uptime.



* **uptime -s** : Displays System up since.



11. **free**: Shows the amount of free and used memory (RAM) in the system.

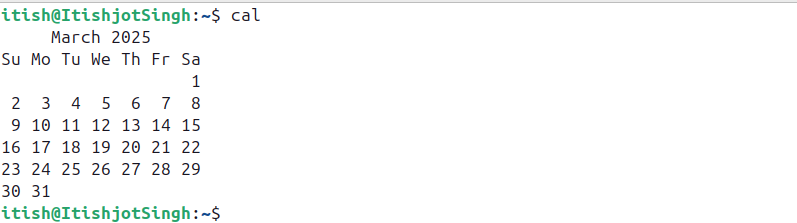


12. **tty**: Prints the file name of the terminal connected to standard input.

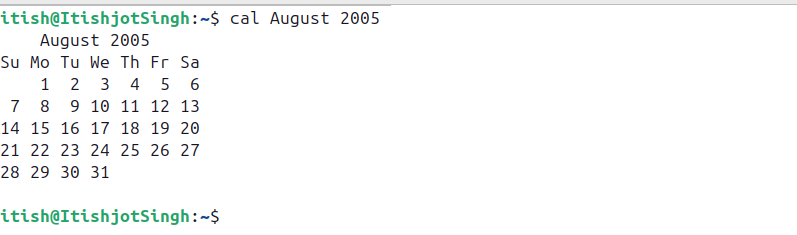


13. **cal** (Calendar): There are several sub commands associated with cal command:

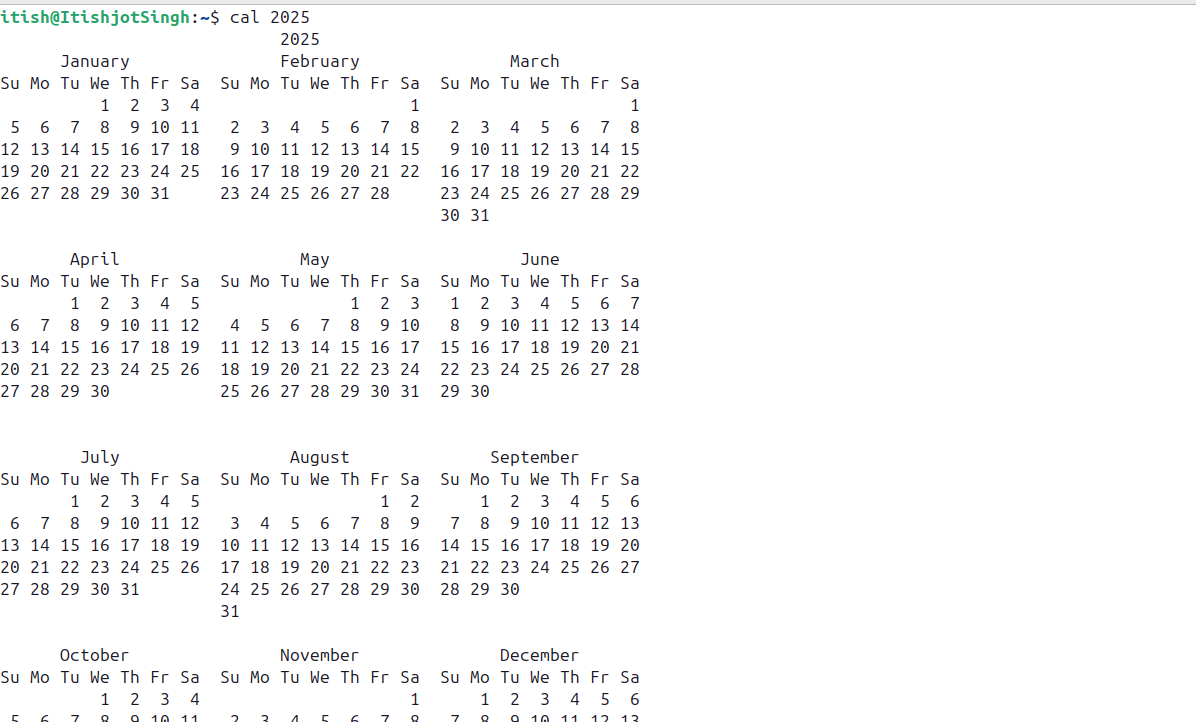
* **cal**: Displays a calendar for the current month or any specified month and year.



* **cal<month><year>**: Displays a calendar for the specified year of the specified month.



* **cal<year>**: Displays a calendar for the specified year.



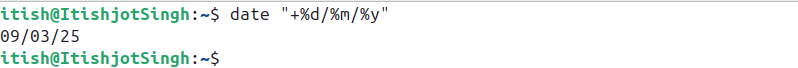
14. **date**:

There are several sub commands associated with date command.

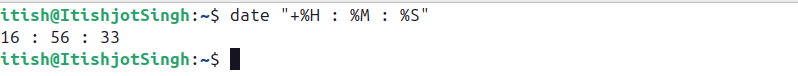
* **date**: Prints the current date and time according to the system's clock settings.



* **date “+%d/%m/%y”**: Prints current system date in format “day/month/year”.



* **date “+%H : %M : %S”**: Prints current system time in format “Hour/Minute/Second”.



* **date “+%y”**: Prints current system year.



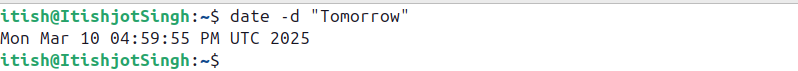
* **date “+%B”**: Prints current system month.



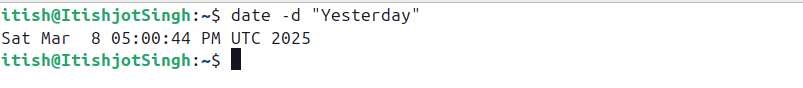
* **date “+%A”**: Prints current system Day.



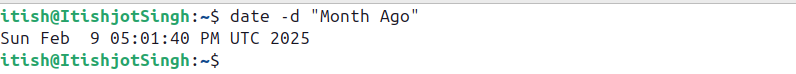
* **date -d “Tomorrow”**: Prints date, time of Tomorrow.



* **date -d “Yesterday”**: Prints date, time of Yesterday.



* **date -d “Month Ago”**: Pronts date, time of a month ago.



15. **hostname**: Displays the system's hostname (computer name).



16. **reboot**: Initiates a system reboot, restarting the operating system.



17. **clear**: Clears the terminal screen, providing a clean workspace for new commands and output.



**Program 5**

Write a program to implement process concepts using C language by printing process Id.

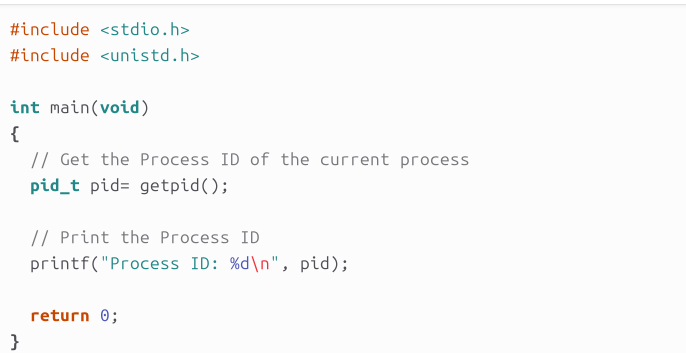
**Solution**

To demonstrate process concepts using C language and print the process ID (PID), we can create a simple program that prints the PID of the current process. We'll use the getpid() function provided by the unistd.h header file to retrieve the PID.

**Step 1: Entering the desired directory using** cd**.**

****

**Step 2: Write the basic code of C to demonstrate the pid function.**

****

**Code:**

#include <stdio.h>

#include <unistd.h>

int main(void)

{

// Get the process ID (PID) of the current process

pid\_t pid = getpid();

// Print the PID

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

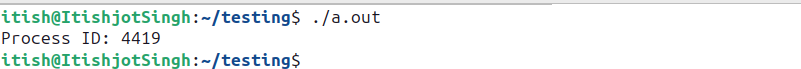
return 0;

}

**Step 3: Using GCC Compiler, compile the file and create an executable file.**

****

**Step 4: Execute the executable file.**

****