Experiment 2: Linux File System Permissions and Essential Commands

Aim

The aim of this experiment is to study Linux file system permissions and to practice some essential Linux commands. By doing this, we learn how to manage files, directories, and their access rights using command line.

Requirements

1. A system with Linux operating system installed (Ubuntu, Fedora, etc.).
2. Access to a terminal (command line interface).
3. Basic knowledge of Linux directory structure.

COMMANDS WITH THEIR EXPLANATION

Command	Purpose	Example
pwd	Show current directory	pwd
ls	List files	ls -la
cd	Change directory	cd /home
mkdir	Create directory	mkdir newfolder
rmdir	Remove empty directory	rmdir oldfolder
touch	Create empty file	touch newfile.txt

Command	Purpose	Example
cp	Copy files	cp file1.txt file2.txt
mv	Move/rename files	mv old.txt new.txt
rm	Delete files	rm -i file.txt
cat	Display file content	cat file.txt
less	View file with pagination	less largefile.txt
head	Show file beginning	head -n 5 file.txt
tail	Show file end	tail -n 5 file.txt
nano	Simple text editor	nano file.txt
chmod	Change permissions	chmod 755 script.sh
chown	Change ownership	sudo chown user:group
whoami	Current user	whoami
sudo	Run as administrator	sudo command
history	Command history	h

Procedure

1. Open the terminal in Linux.
2. Use pwd to check the present working directory.
3. Use ls to list files and directories.
4. Create a directory using mkdir testdir.
5. Change directory with cd testdir.
6. Create an empty file using touch file1.txt.
7. Use ls -l to check file permissions (read, write, execute).
8. Modify permissions with chmod (for example, chmod 755 file1.txt).
9. Change file ownership with chown if needed.
10. Display file content with cat file1.txt.
11. Finally, remove file using rm file1.txt.

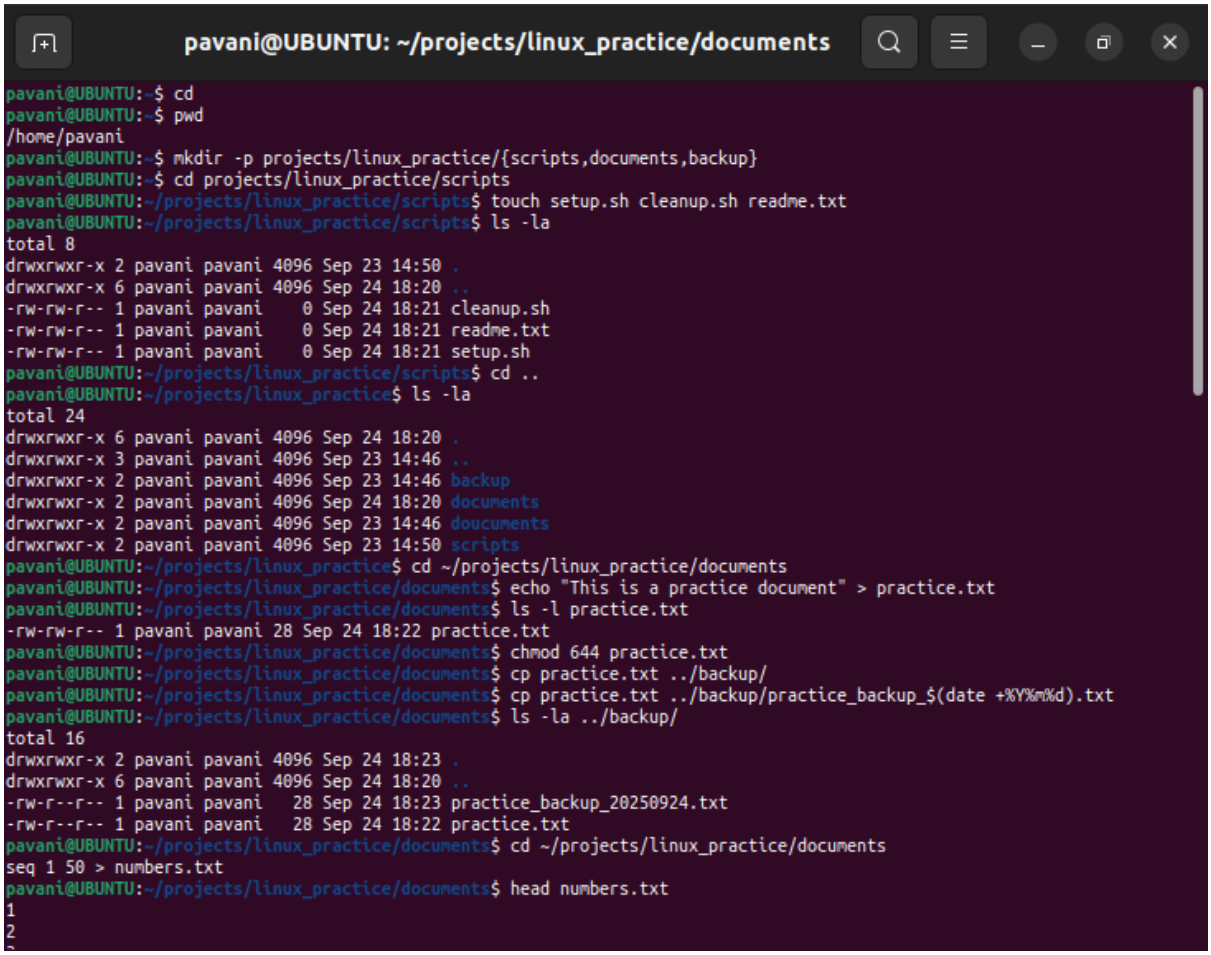
LAB EXERCISE

EXERCISE 1: FILE SYSTEM NAVIGATION

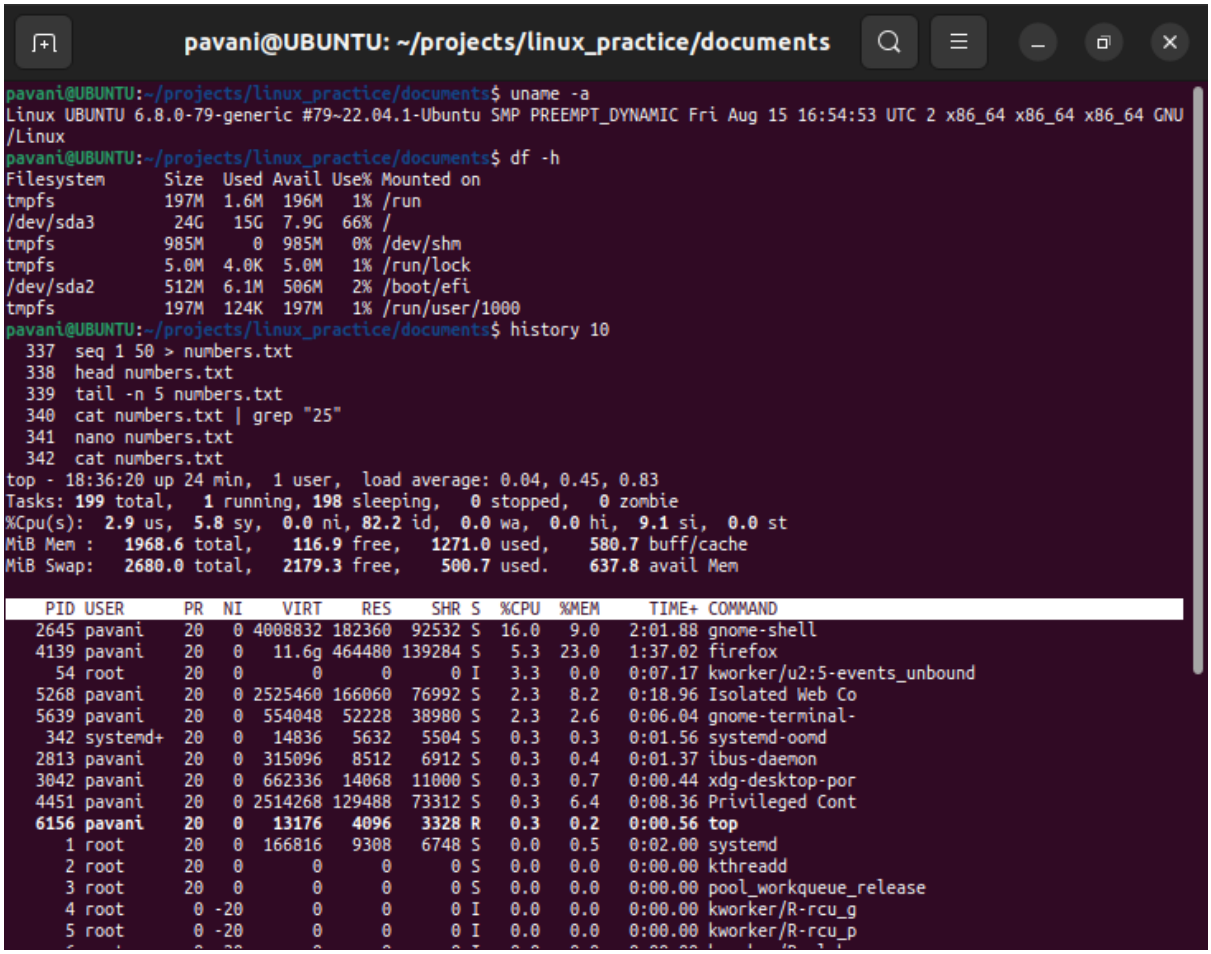
EXERCISE 2:FILE OPERATIONS AND PERMISSIONS

EXERCISE 3: TEXT EDITING AND VIEWING

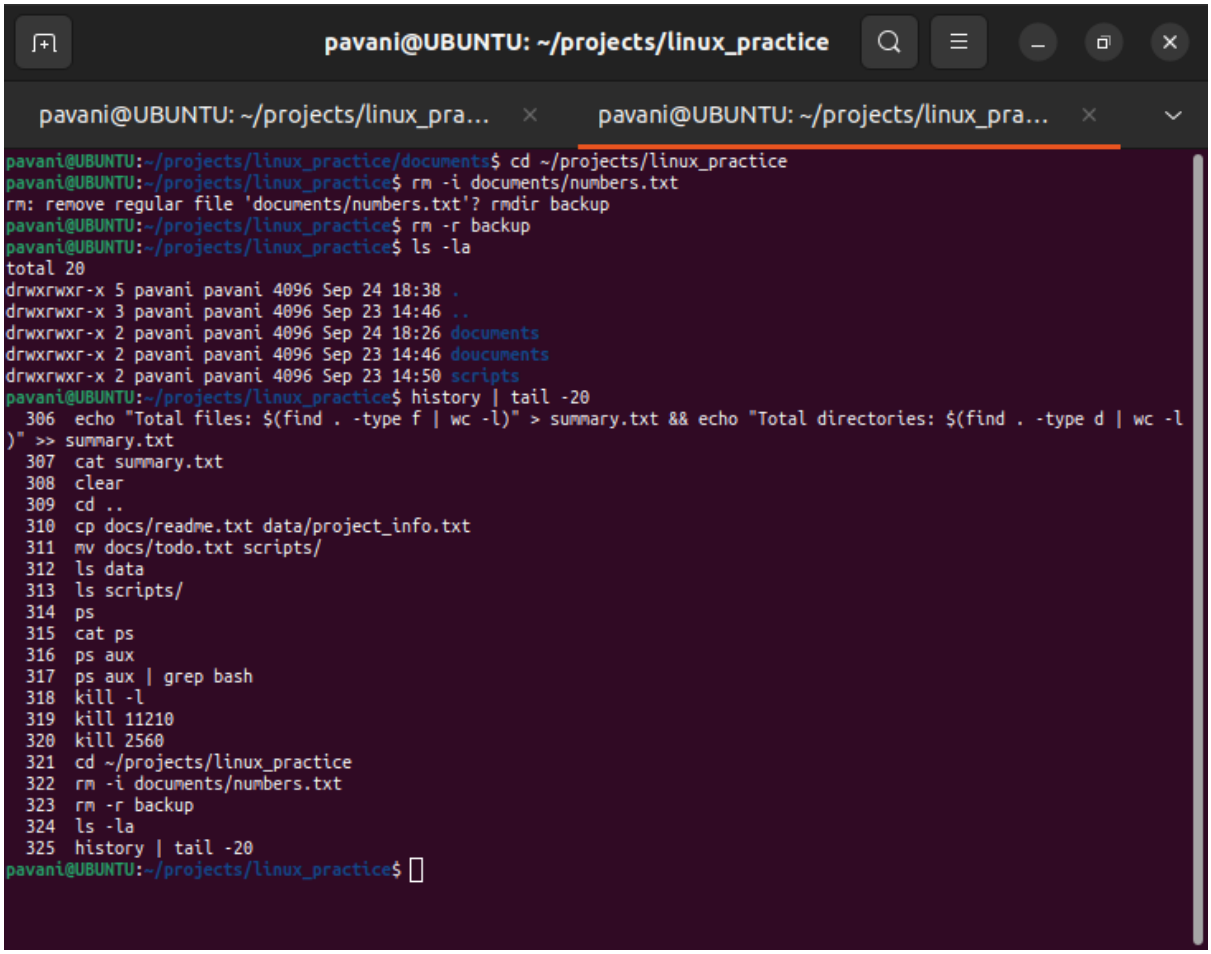
COMMANDS AND OUTPUT



EXPERIMENT 4: SYSTEM EXPLORATION



EXPERIMENT 5-CLEANUP

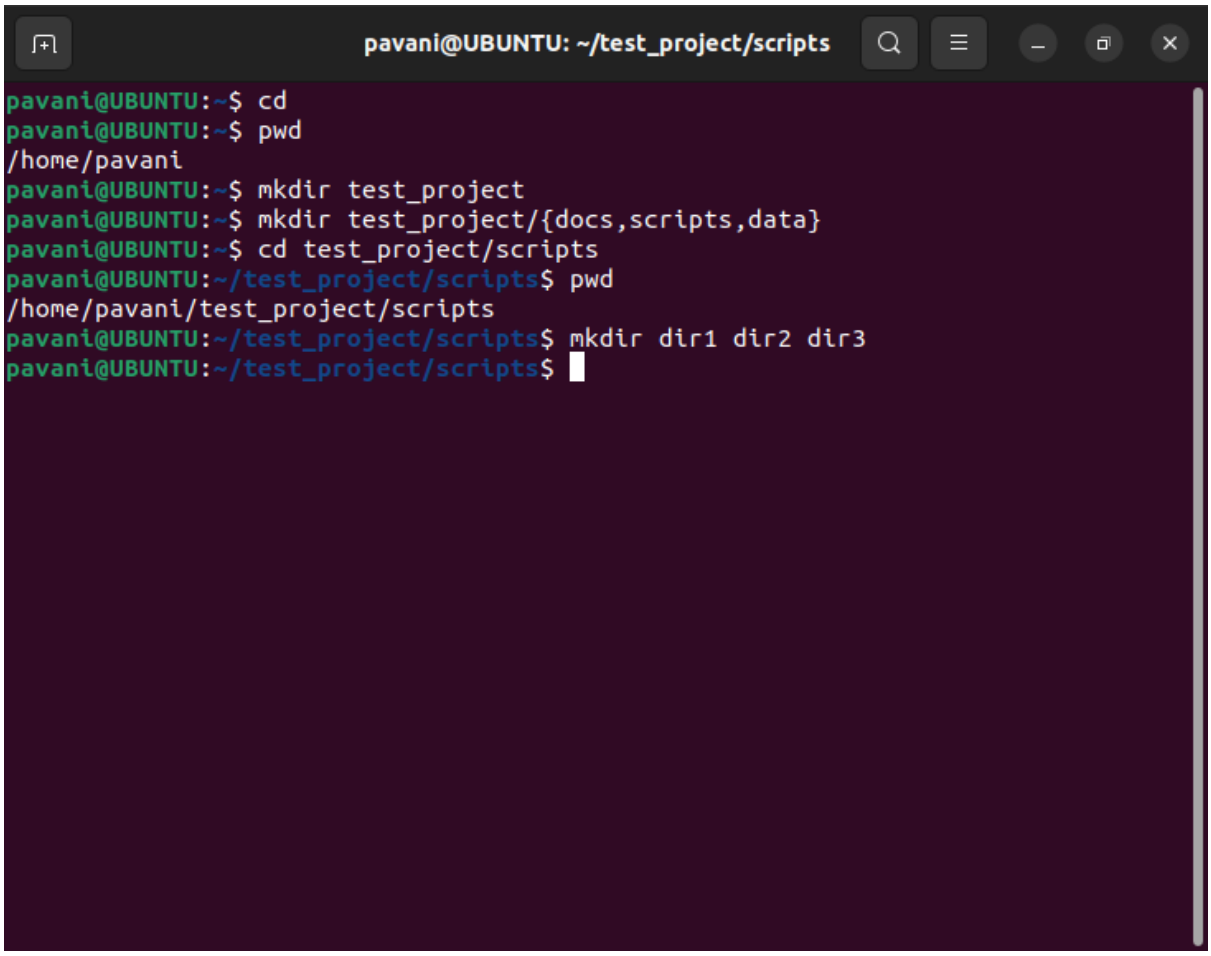


LAB TASKS

TASK 1- DIRECTORY NAVIGATION

Create a directory called test_project in your home directory, then create subdirectories docs, scripts, and data inside it. Navigate to the scripts directory and display your current path.

EXPLANATION-Directory navigation means moving around between different folders and files in a computer system. Every file or folder has a specific location (called a *path*), and we use navigation commands to reach them.



Conclusion:

Directory navigation helps us organize and easily find files in the system. With simple commands, we can move from one folder to another without confusion.

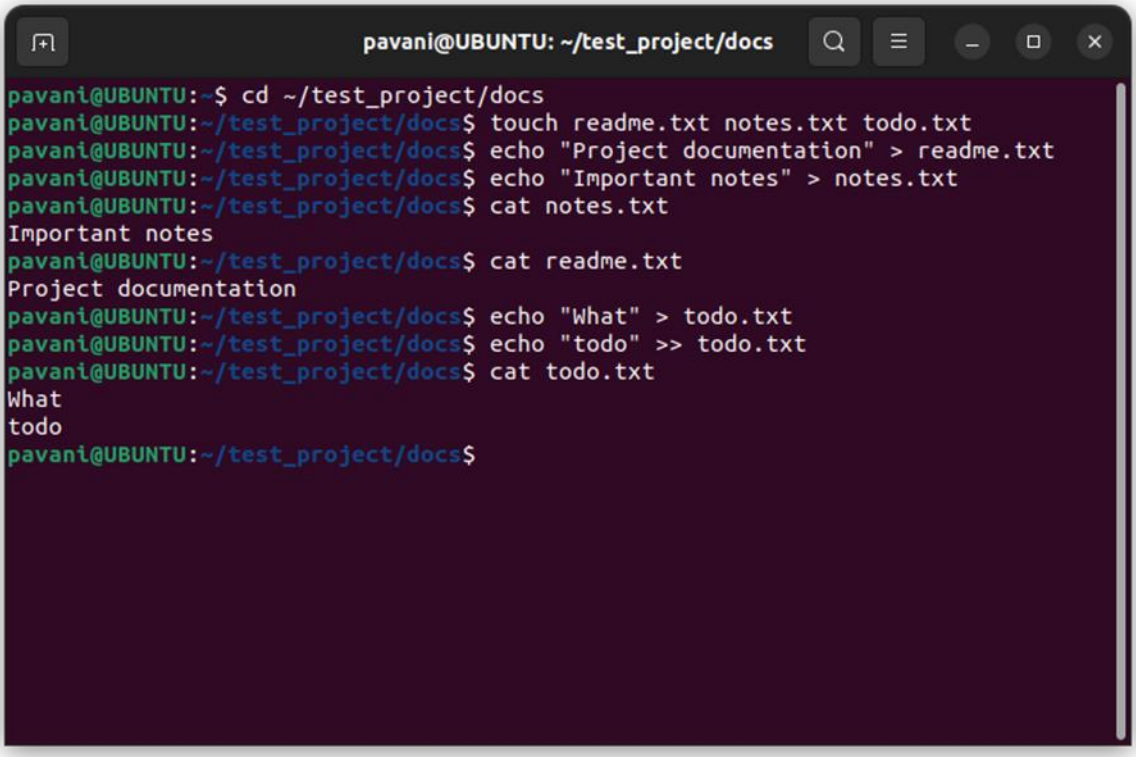
TASK 2 – FILE CREATION AND CONTENT

Create three files in
the docs directory: readme.txt, notes.txt, and todo.txt. Add the text "Project documentation" to readme.txt and "Important notes" to notes.txt. Display the contents of both files.

EXPLANATION

- First, we navigated to the **docs** directory.
- Then, we created three files: readme.txt, notes.txt, and todo.txt using the touch command.
- We added the text **"Project documentation"** to readme.txt and **"Important notes"** to notes.txt using the echo command with redirection (>).

- Finally, we displayed the contents of both files using the cat command.



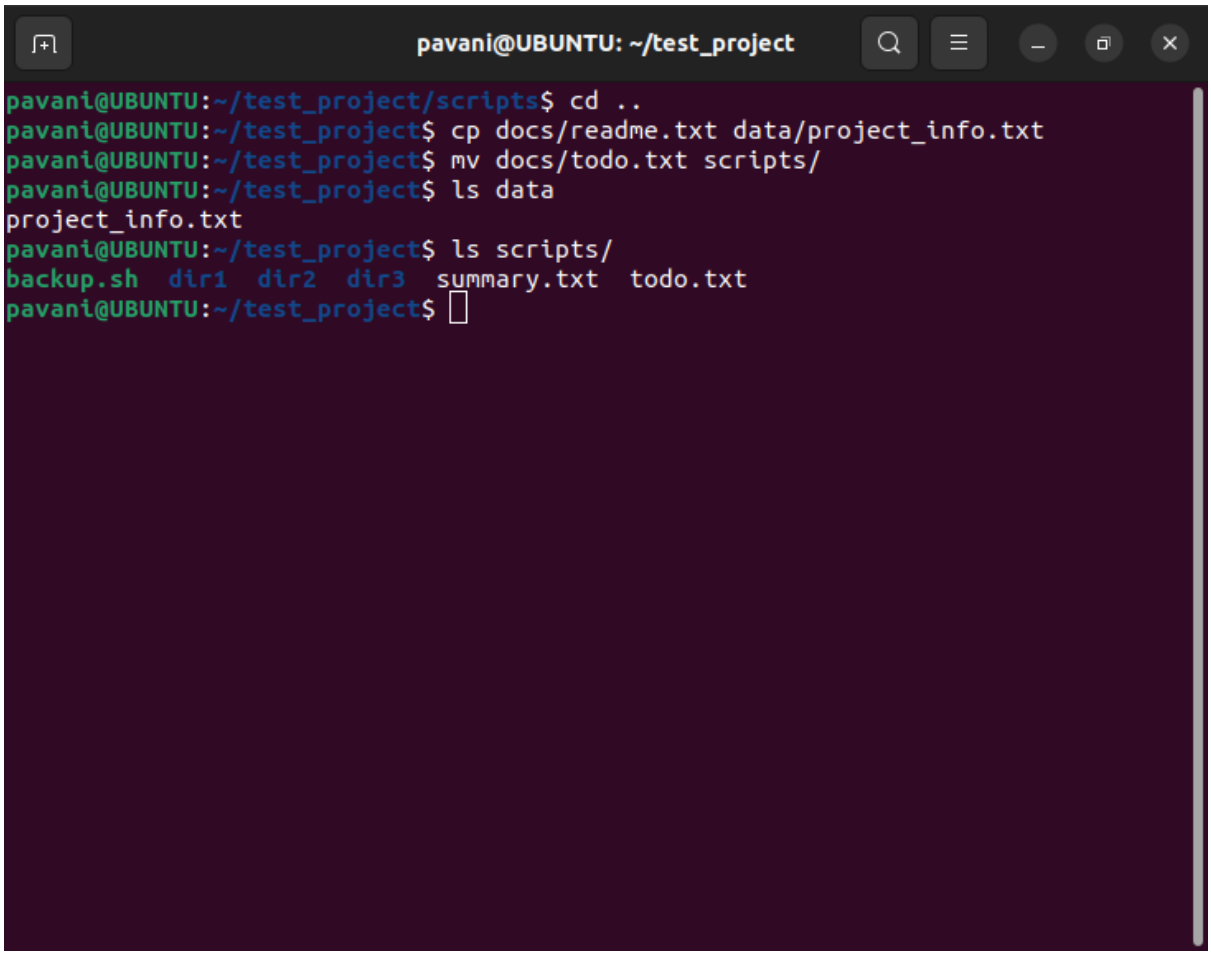
Conclusion:
We successfully created files, added content, and viewed their contents in the terminal.

TASK 3 – FILE OPERATIONS

Copy readme.txt to the data directory and rename the copy to project_info.txt. Then
move todo.txt from docs to scripts directory.

EXPLANATION

- We copied the file **readme.txt** from the *docs* directory to the *data* directory and renamed it as **project_info.txt** using the cp command.
- Next, we moved the file **todo.txt** from the *docs* directory to the *scripts* directory using the mv command.



Conclusion:

We learned how to copy a file to another directory with a new name and how to move a file from one directory to another.

TASK 4 – FILE PERMISSION

Create a shell script file called backup.sh in the scripts directory. Add the content #!/bin/bash and echo "Backup complete" to it. Make the file executable only for the owner.

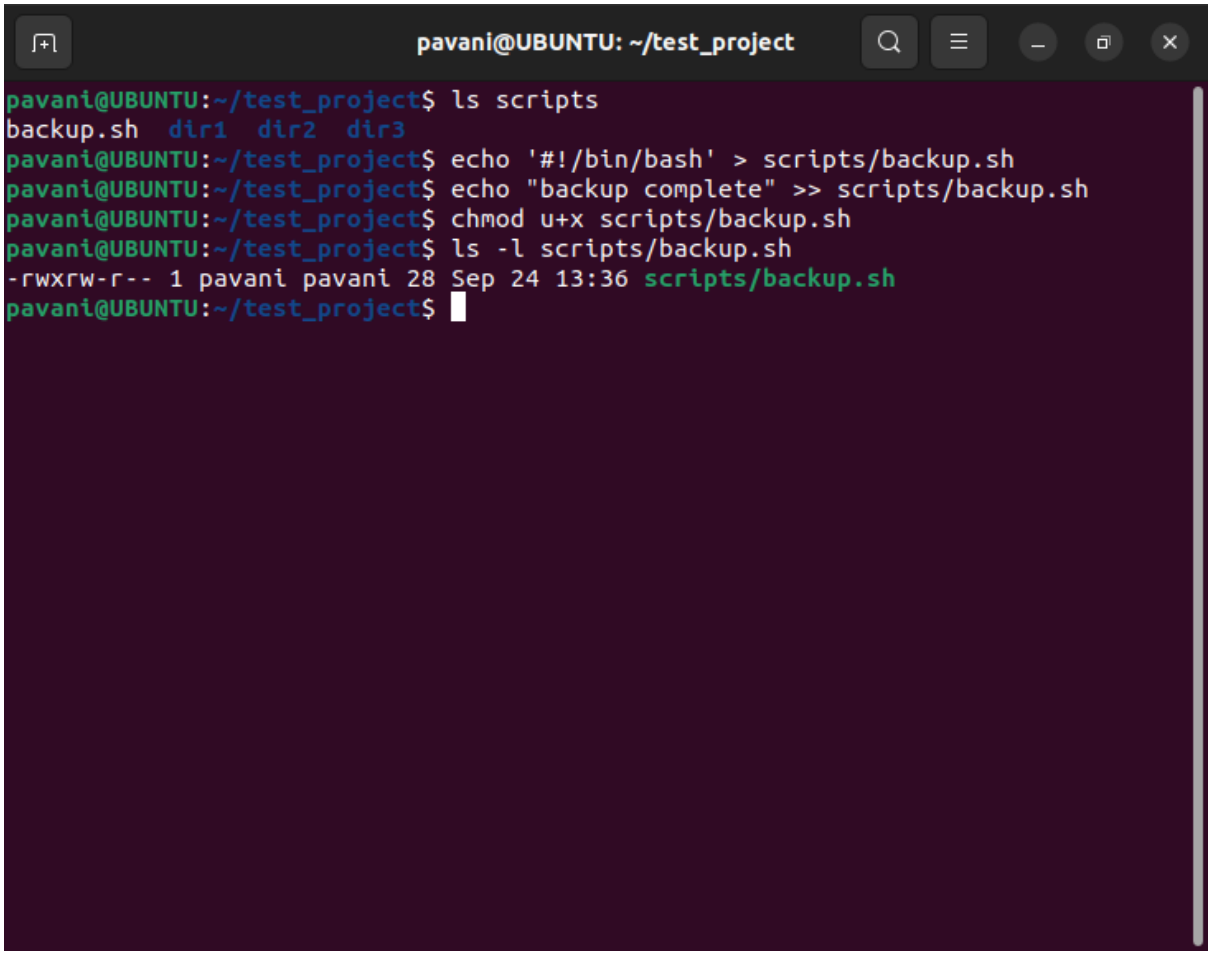
EXPLANATION

- We created a shell script file named **backup.sh** inside the *scripts* directory using the touch command.
- Added the content:
- #!/bin/bash
- echo "Backup complete"

using a text editor or redirection.

-

Then, we made the script executable **only for the owner** by setting permissions with the command



Conclusion:

We successfully created a shell script, added commands to it, and restricted execution rights to the owner only.

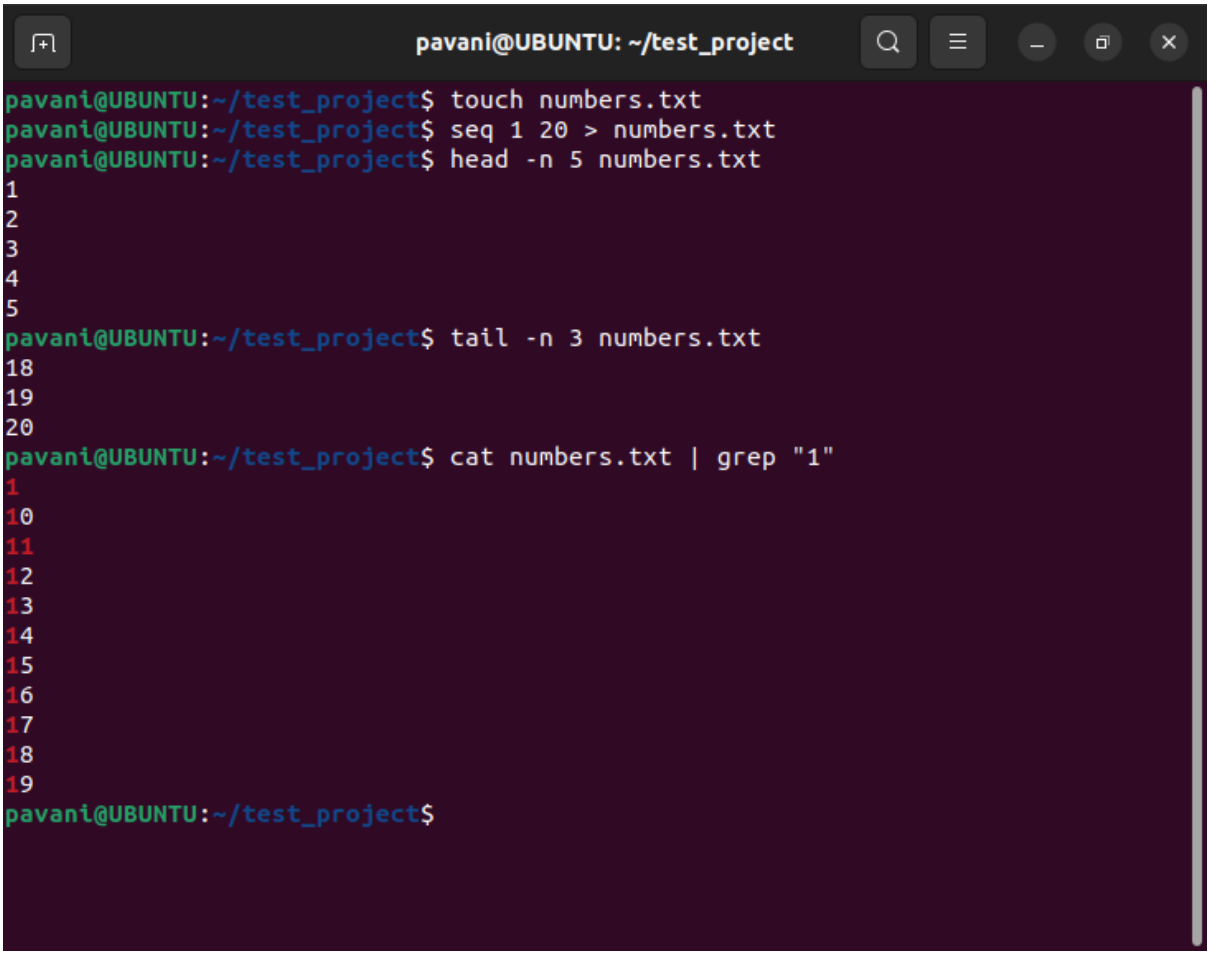
TASK 5 – FILE VIEWING

Create a file called numbers.txt with numbers 1 to 20 (each on a new line). Display only the first 5 lines, then

only the last 3 lines, then search for lines containing the number "1".

EXPLANATION

- A file named **numbers.txt** was created containing numbers from 1 to 20, each written on a new line.
- The **first 5 lines** of the file were displayed.
- The **last 3 lines** of the file were displayed.
- Then, all lines containing the number **“1”** were searched and shown.



Conclusion:

This task helped us understand how to create a file with

data, view only selected parts of it, and search for specific patterns inside a file.

TASK 6 – TEXT EDITING

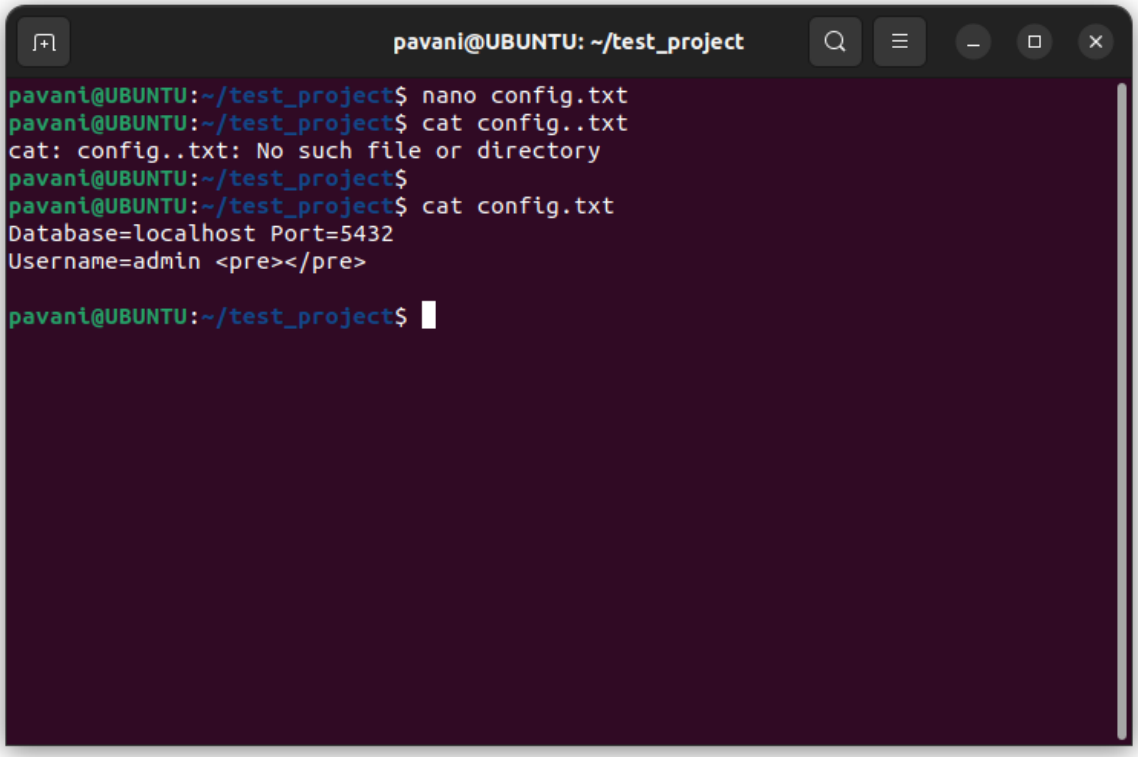
Using nano, create a file called config.txt with the following content:

Database=localhost Port=5432 Username=admin <pre></pre> Save the file and then display its contents.

EXPLANATION

- A file named **config.txt** was created using the nano text editor.
- The following content was written inside the file:
    - Database = localhost
    - Port = 5432
    - Username = admin

- The file was saved and then its contents were displayed.



Conclusion:
We learned how to create a text file using a text editor, add configuration details to it, save the file, and view its contents.

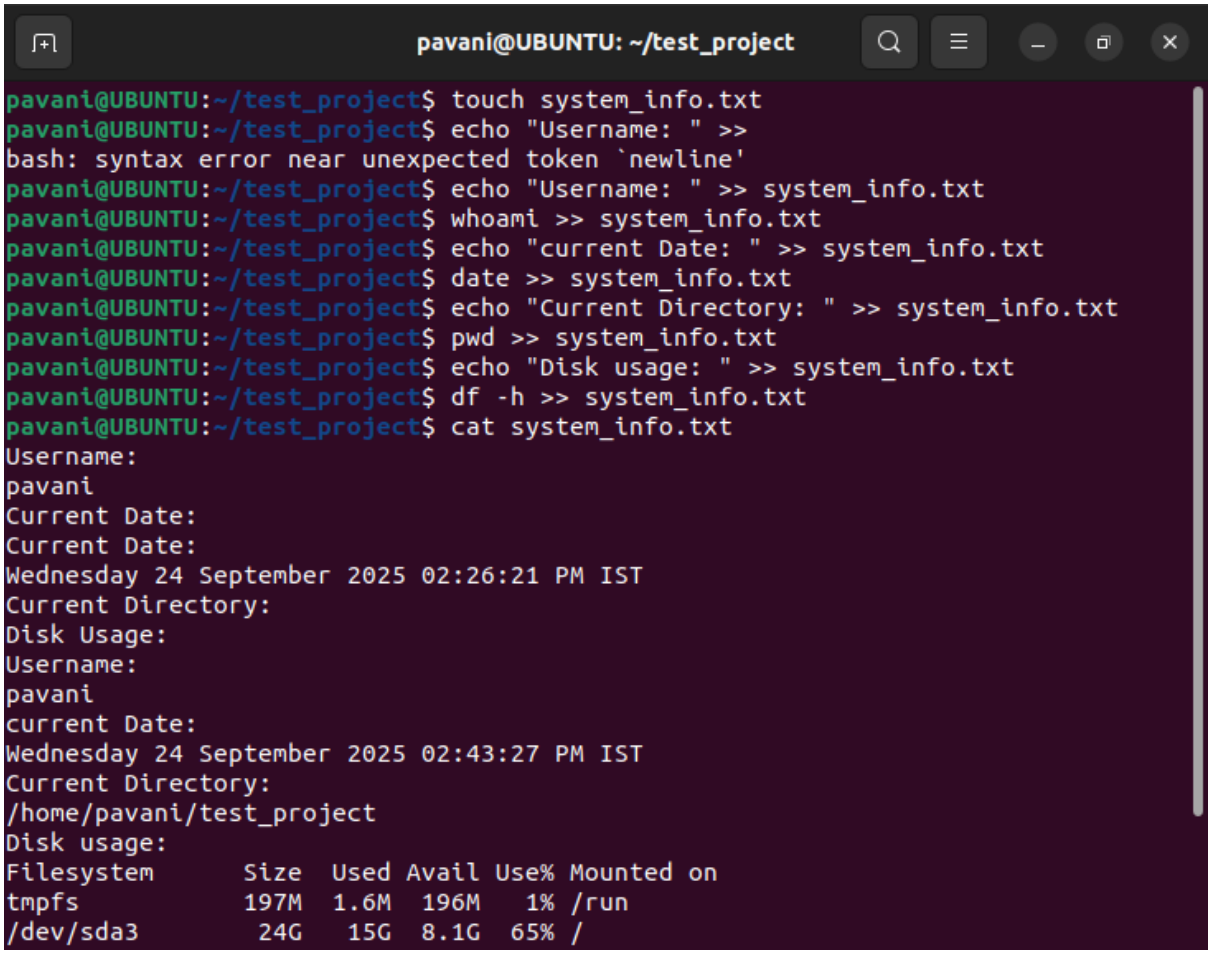
Task 7: System Information

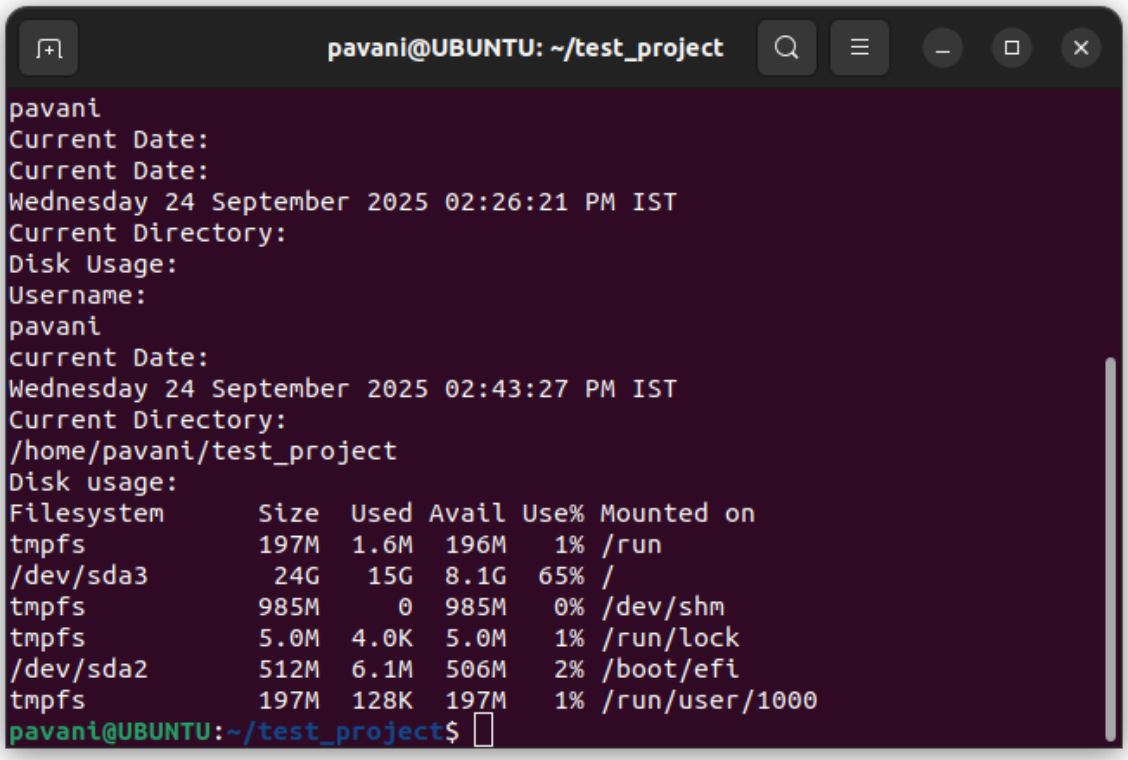
Create a file called system_info.txt that contains: your username, current date, your current directory, and disk usage information in human-readable format.

EXPLANATION

- A file named **system_info.txt** was created.

- The file contains the following information:
    - The current username.
    - The current date.
    - The present working directory.
    - Disk usage details in a human-readable format.
- All this information was saved inside the file and then viewed.





Conclusion:

We learned how to gather system-related details and store them in a text file for reference.

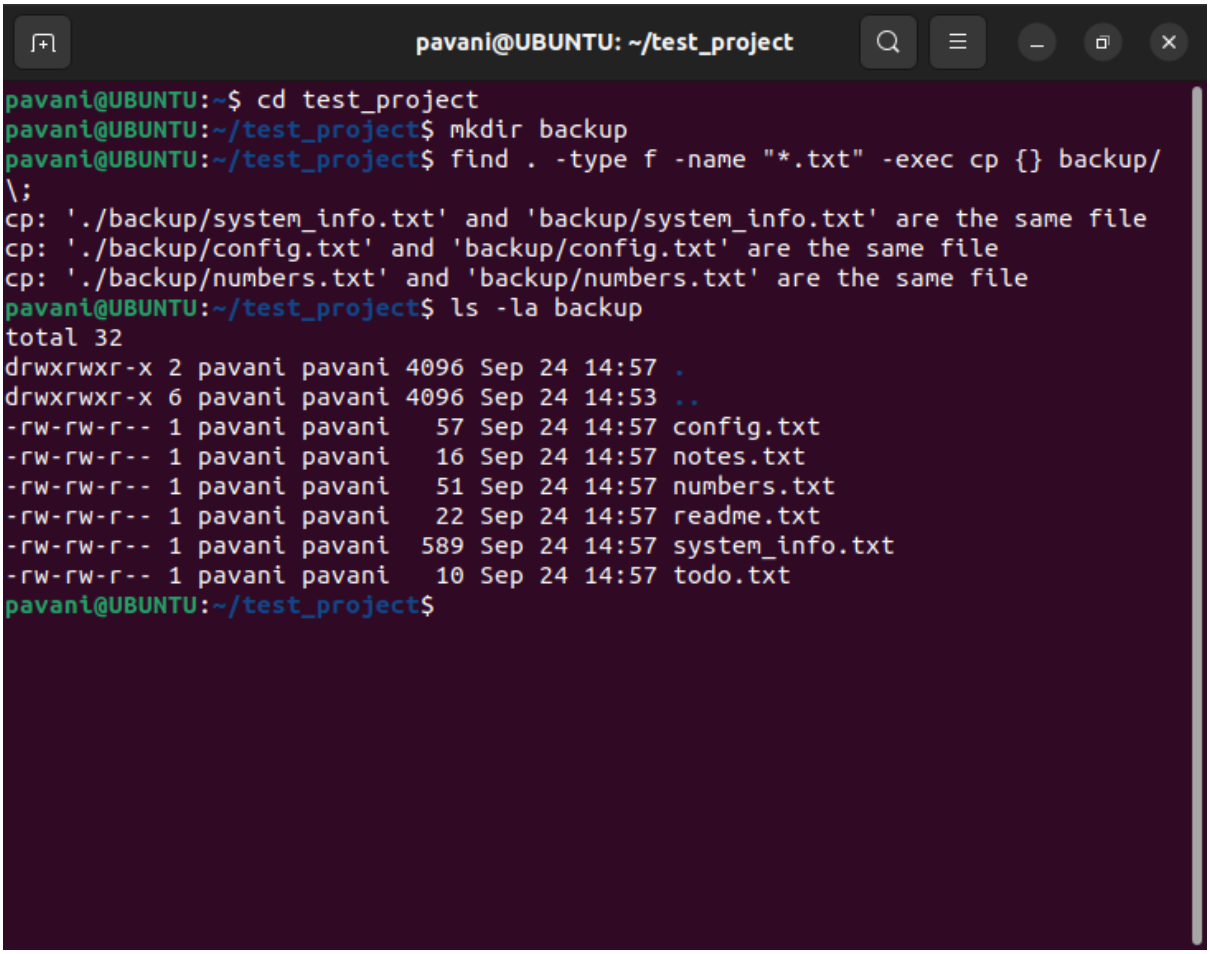
Task 8: File Organization

In your test_project directory, create a backup folder. Copy all .txt files from all subdirectories into this backup folder.

Then list all files in the backup folder with detailed information.

EXPLANATION

- In the **test_project** directory, a folder named **backup** was created.
- All text files (with the extension .txt) from every subdirectory were copied into the backup folder.
- After copying, the contents of the backup folder were listed with detailed information such as file size, permissions, and modification date.



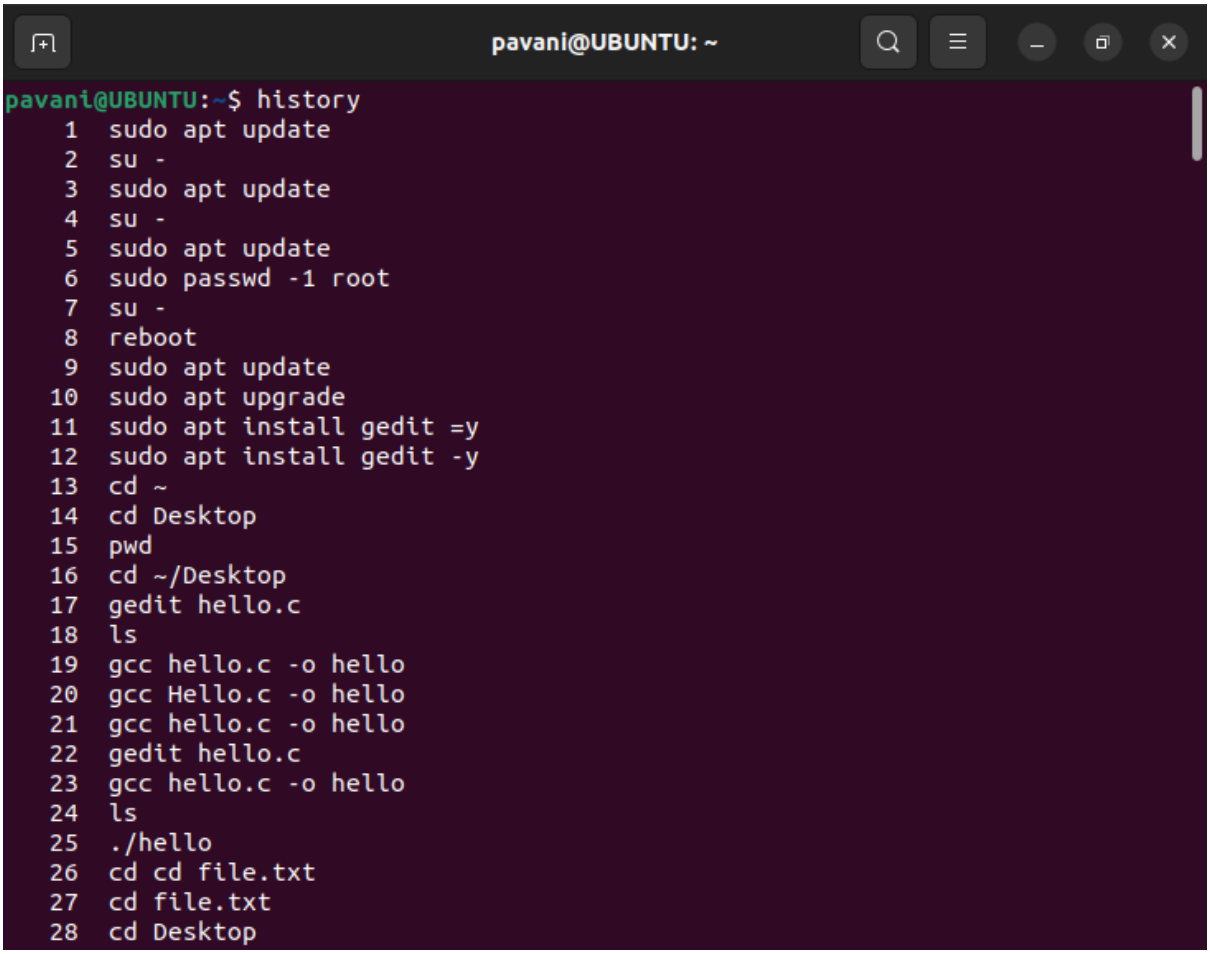
Conclusion:
We learned how to organize files by collecting them from different locations into a single folder and how to view detailed file information.

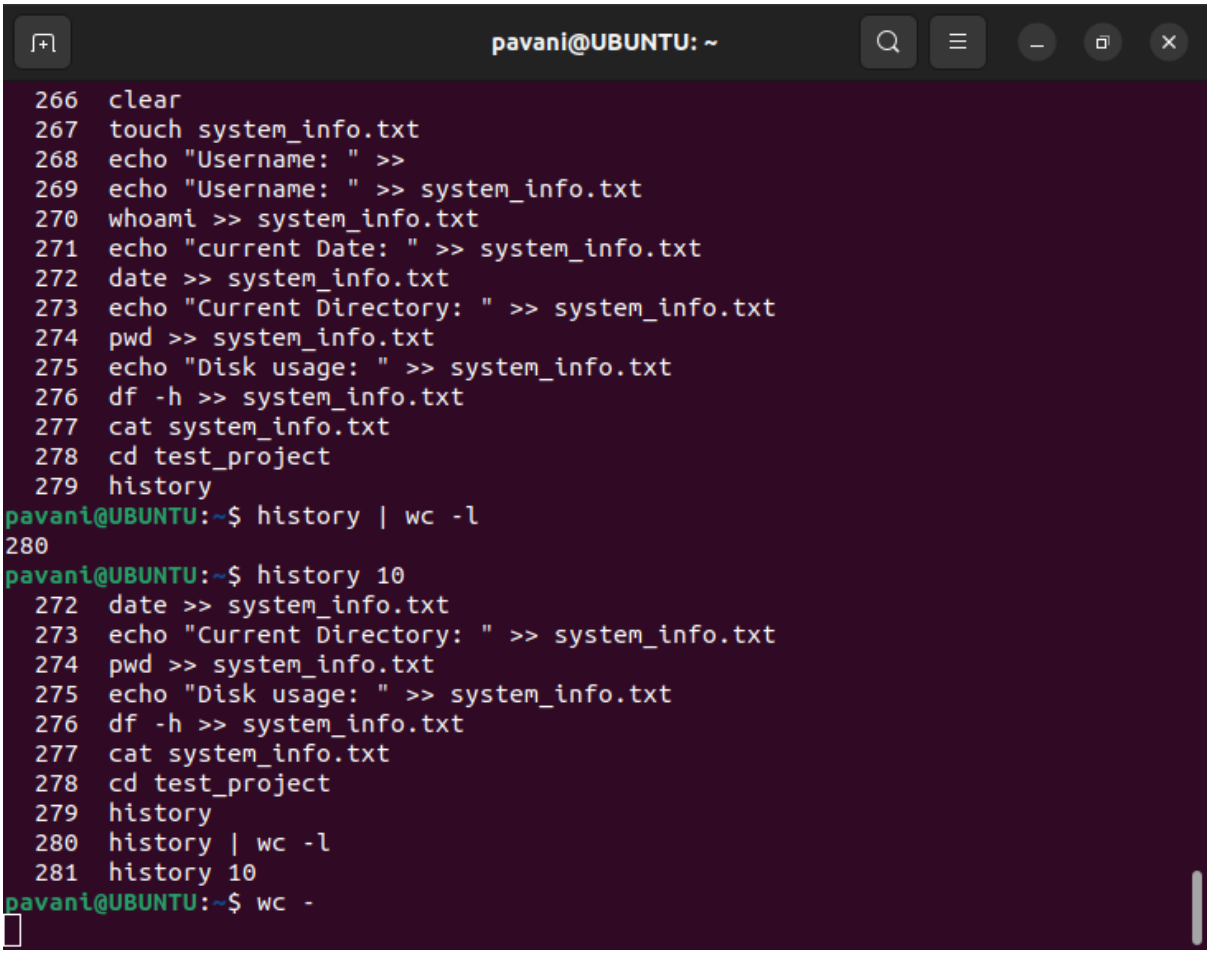
Task 9: Process and History

Display your command history and count how many commands you've executed. Then show the top 10 most recent commands.

EXPLANATION

- The full command history of the session was displayed.
- The total number of commands executed so far was counted.
- Finally, the most recent 10 commands were shown from the history.


pavani@UBUNTU: ~
pavani@UBUNTU:~$ history
1 sudo apt update
2 su -
3 sudo apt update
4 su -
5 sudo apt update
6 sudo passwd -1 root
7 su -
8 reboot
9 sudo apt update
10 sudo apt upgrade
11 sudo apt install gedit =y
12 sudo apt install gedit -y
13 cd ~
14 cd Desktop
15 pwd
16 cd ~/Desktop
17 gedit hello.c
18 ls
19 gcc hello.c -o hello
20 gcc Hello.c -o hello
21 gcc hello.c -o hello
22 gedit hello.c
23 gcc hello.c -o hello
24 ls
25 ./hello
26 cd cd file.txt
27 cd file.txt
28 cd Desktop



Conclusion:

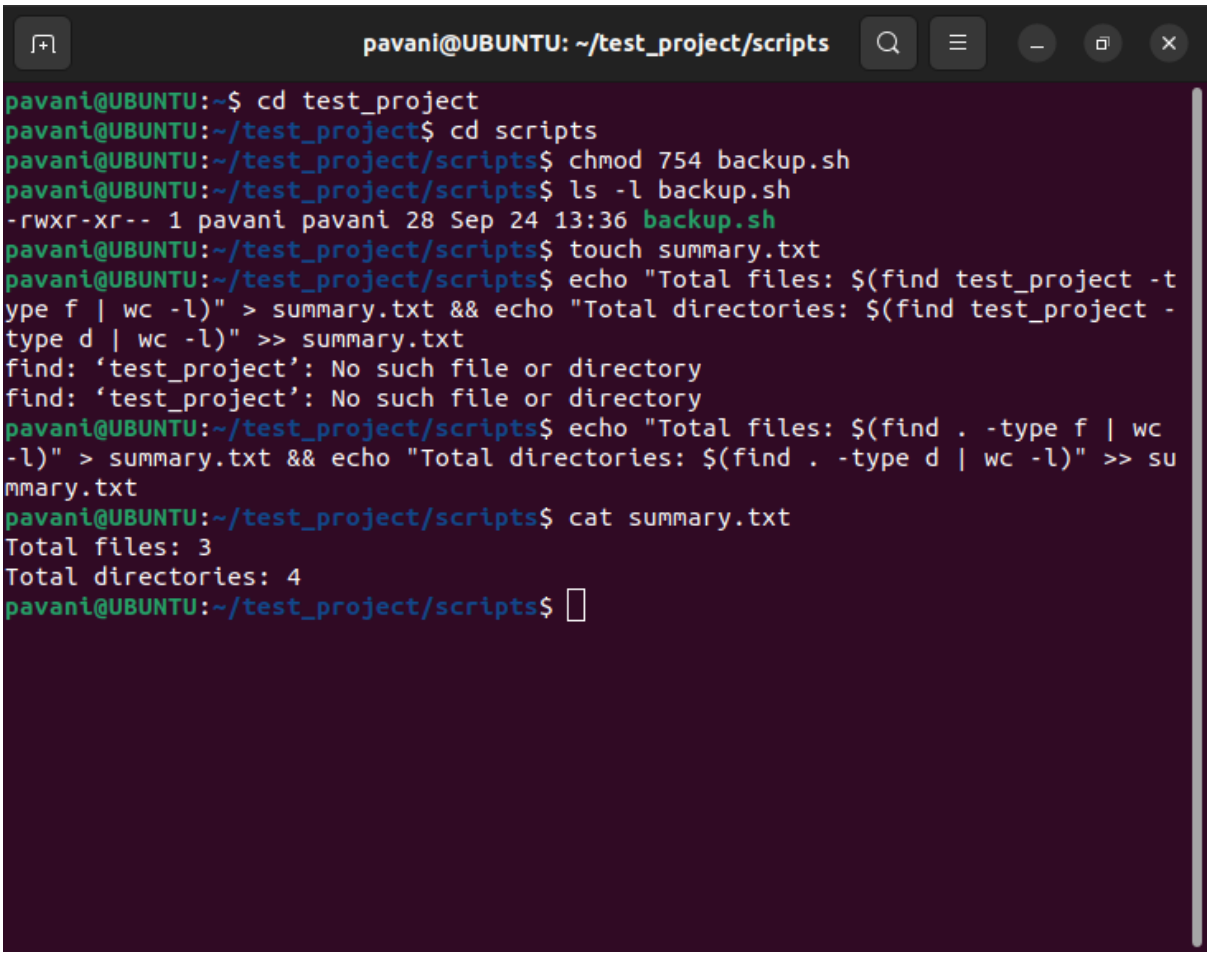
This task helped us understand how to view the command history, count executed commands, and check the latest commands used in the terminal.

Task 10: Comprehensive Cleanup

Set the permissions of your backup.sh script to be readable, writable, and executable by owner, readable and executable by group, and readable by others. Then create a summary file that lists the total number of files and directories in your entire test_project.

EXPLANATION

- The script **backup.sh** was updated so that the **owner** has read, write, and execute permissions, the **group** has read and execute permissions, and **others** have read permission.
- A **summary file** was created that lists the total number of files and directories in the entire **test_project** directory.



Conclusion:

This task demonstrated how to manage file permissions for different users and how to create a summary report of a project directory’s structure.

Quick Verification Commands

After each task, verify your work:

```
# Check current location
pwd

# List files with details
ls -la

# Check file contents
cat filename.txt

# Check permissions
ls -l filename

# Check directory structure
ls -R

# Check command history
history 5
```

Common Commands You Should Know After This

Task	Key Commands Used
1	mkdir, cd, pwd
2	touch, echo >, cat
3	cp, mv
4	chmod, ls -l
5	head, tail, grep
6	nano
7	whoami, date, df -h
8	find, wildcard *
9	history, wc -l
10	chmod with numbers, find with type

Observation

Every file and directory in Linux has three types of permissions: read (r), write (w), and execute (x).

These permissions are assigned to user (owner), group, and others.Commands like chmod and chown help in modifying permissions and ownershipThe terminal shows permissions in a 10-character string (e.g., -rwxr-xr--).

Conclusion

From this experiment, we conclude that Linux provides a powerful permission system to control access to files and directories. By using essential commands like ls, pwd, mkdir, touch, chmod, and chown, we can create, manage, and secure files. Understanding file system permissions is important for system security and effective usage of Linux.