

## Experiment [1]: [Linux OS Environment Setup]

Name: Lakshay dhanda , Roll No.: 590029328, Date: 30-10-25

AIM:

- To install and configure different Linux operating system environments on a Windows machine. We will use two distinct technologies: **Windows Subsystem for Linux (WSL)** for a lightweight command-line environment and **Oracle VirtualBox** for a full graphical virtual machine.

Requirements:

- A Windows 10/11 PC.
- Administrator access and **hardware virtualization enabled in the BIOS/UEFI**.
- An internet connection.

Theory:

- This experiment is designed to provide hands-on experience with two primary methods of running Linux on a Windows host. This is ideal for developers and system administrators who require a Linux command-line without the overhead of a full virtual machine.
- **Oracle VirtualBox**, on the other hand, is a traditional Type 2 hypervisor. It creates a complete, virtualized computer system on which a guest operating system (like Ubuntu or Linux Mint) can be installed. This method provides a fully isolated environment, complete with a graphical user interface (GUI).

---

## Procedure & Observations

Part 1: Installing and Configuring WSL (Ubuntu)

### **Exercise 1: [Installing WSL on Windows]**

- **Task Statement:** Enable the required Windows features and install the Ubuntu Linux distribution using a single command.
- **Explanation:** This demonstrates how the modern `wsl --install` command simplifies the entire setup process, automating what previously required multiple manual steps.
- **Command(s):**

```
wsl --install -d ubuntu
```

- **Observation:** The command automatically enabled the "Virtual Machine Platform" and "Windows Subsystem for Linux" optional components. It then proceeded to download the Ubuntu distribution. The system requested a reboot to complete the final stage of the installation.

### **Exercise 2: [Configuring the Ubuntu Distribution]**

- **Task Statement:** After the initial installation and reboot, configure the Ubuntu environment by creating a new user account.
- **Explanation:** This step is crucial for security and user management within the Linux environment. The new UNIX username and password created are separate from the Windows user account.
- **Observation:** Upon reboot, a terminal window opened automatically. It prompted for a "New UNIX username" and a password. After entering the credentials, the setup was complete and the command-line interface became available.

### Exercise 3: [Verifying WSL Installation]

- **Task Statement:** Confirm that the WSL installation is successful and the Ubuntu distribution is ready for use.
- **Explanation:** This command provides a simple way to list all installed WSL distributions, showing their names, versions, and current state.
- **Command(s):**

```
wsl -l -v
```

- **Output:**

NAME	STATE	VERSION
* Ubuntu	Running	2

- **Observation:** The output confirmed that Ubuntu was correctly installed and was currently in a "Running" state, with version 2 (indicating it is running on the WSL 2 architecture).

---

## Part 2: Installing VirtualBox and a Linux VM (Linux Mint)

### Exercise 4: [Installing Oracle VirtualBox]

- **Task Statement:** Download and install the Oracle VirtualBox hypervisor on the Windows host machine.
- **Procedure:** The VirtualBox installer was downloaded from the official website. Permission to install device software for network interfaces was granted.
- **Observation:** The VirtualBox application was successfully installed on the Windows system, along with the necessary drivers to support virtual machines.

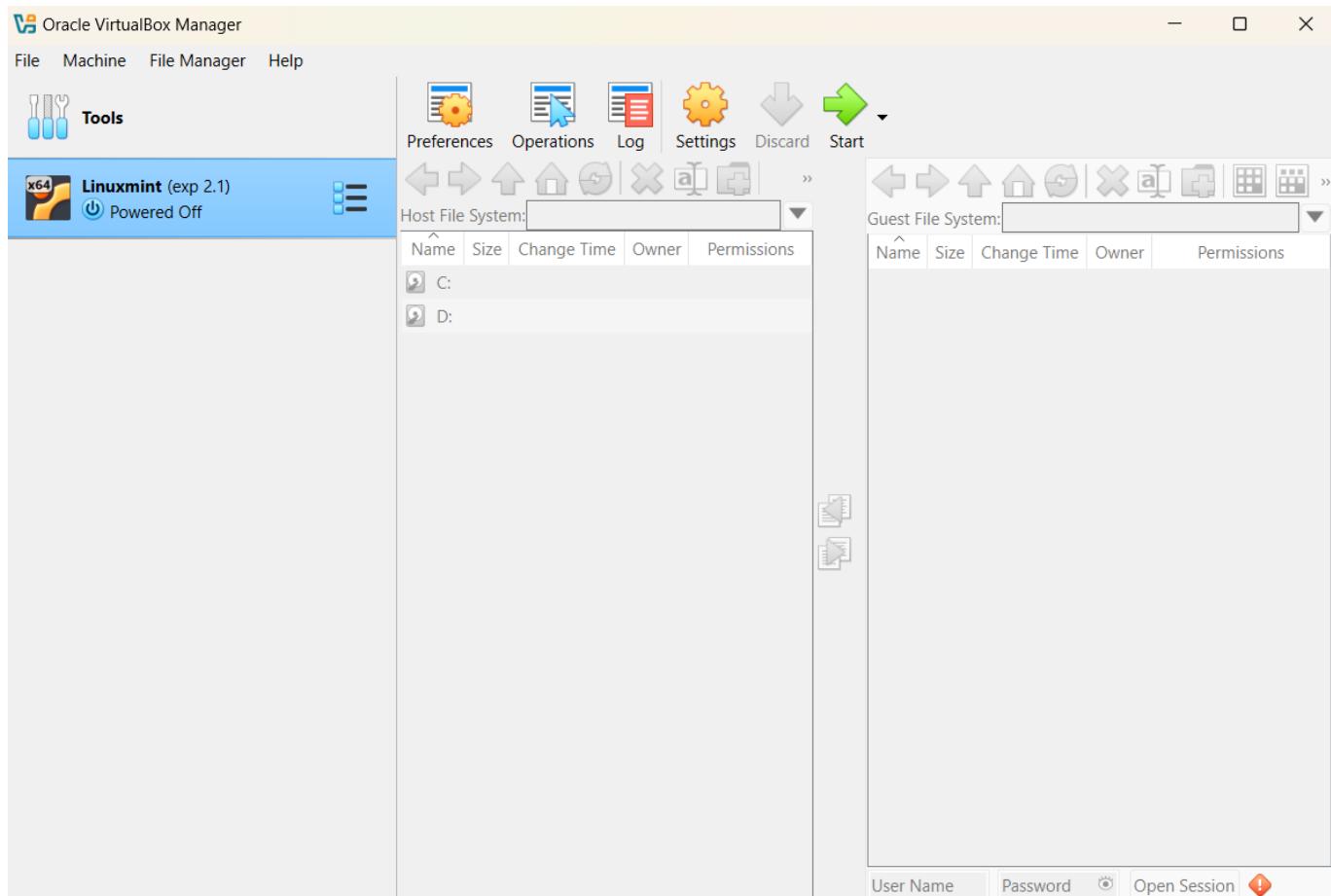
### Exercise 5: [Creating a Virtual Machine]

- **Task Statement:** Create a new virtual machine to host the Linux Mint operating system.
- **Procedure:** In the VirtualBox Manager, a new VM was created. The name was set to "Linux Mint", and a downloaded .iso file was selected as the installation medium. Hardware resources were configured with **4096 MB RAM** and **2 CPUs**. A new dynamically allocated virtual hard disk of **25 GB** was created.

- **Observation:** The VM was configured with the specified resources, creating a virtualized hardware environment ready to receive an operating system.

### Exercise 6: [Installing Linux Mint]

- **Task Statement:** Install the Linux Mint OS on the virtual machine.
- **Procedure:** The newly created VM was started, which booted directly from the Linux Mint .iso.
- **Observation:** The installation proceeded without issues, partitioning the virtual disk and copying the OS files. The process was identical to a standard installation on a physical computer.



---

## Result

- The experiment was successfully completed by setting up two distinct Linux environments. **Windows Subsystem for Linux** and a complete **virtual machine** with Linux Mint. This project demonstrated proficiency in using both a compatibility layer and a full hypervisor to meet different virtualization needs.
- 



## Experiment [2]: [Linux file systems permissions and essential commands]

Name: lakshay dhanada Roll.: 590029328 Date: 30-10-2025

AIM:

- [To Learn linux file systems permissions and essential commands]

Requirements:

- [Any Linux Distro, any kind of text editor (vs code, vim, notepad, nano, etc,)]

Theory:

- [Basic Linux file systems permissions and essential commands]

## Procedure & Observations

### TASK 1: [Directory Navigation]

Task Statement:

- [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.]

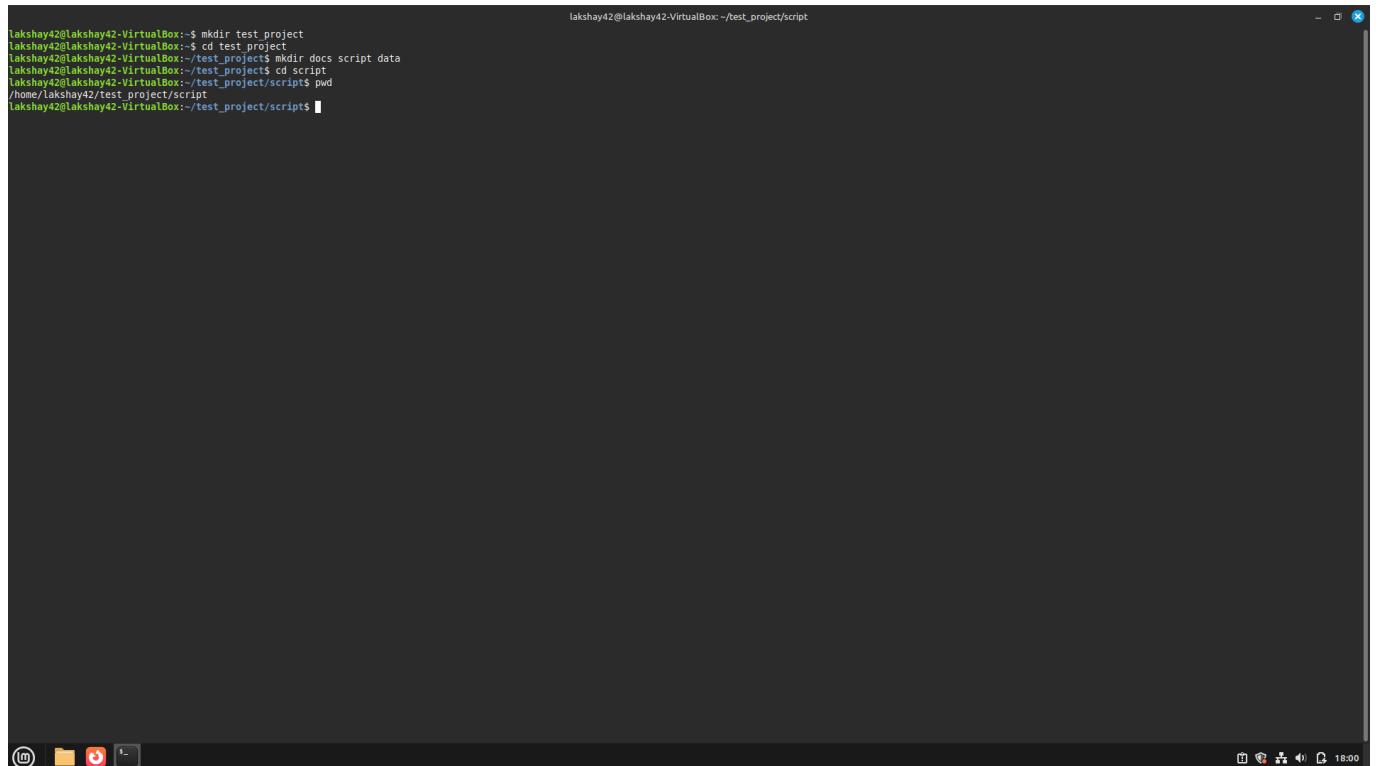
Explanation:

- [ Use mkdir to create the wanted directory we can use cd to navigate and use pwd to show current path ]

Command(s):

```
''' mkdir test_project cd test_project mkdir docs scripts data cd scripts pwd '''
```

Output:



```

lakshay42@lakshay42-VirtualBox:~$ mkdir test_project
lakshay42@lakshay42-VirtualBox:~$ cd test_project
lakshay42@lakshay42-VirtualBox:~/test_project$ mkdir docs script data
lakshay42@lakshay42-VirtualBox:~/test_project$ cd script
lakshay42@lakshay42-VirtualBox:~/test_project/script$ pwd
/home/lakshay42/test_project/script
lakshay42@lakshay42-VirtualBox:~/test_project/script$ 

```

The screenshot shows a terminal window with a dark background. The command history at the top shows the user navigating into a 'script' directory within a 'test\_project' folder. The bottom of the window shows the system tray with icons for battery, signal, and volume, and the time '18:00'.

## TASK 2: [File Creation and Content]

### Task Statement:

- [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:

- [We can use touch to create empty files and using echo "text" > file.txt to add content to a file and using cat to display file contents]

### Command(s):

```

cd docs
touch readme.txt notes.txt todo.txt
echo "Project documentation" > readme.txt
echo "Important notes" > notes.txt
cat notes.txt
cat readme.txt

```

### Output:

```

lakshay42@lakshay42-VirtualBox:~$ mkdir docs
lakshay42@lakshay42-VirtualBox:~$ cd docs
lakshay42@lakshay42-VirtualBox:~/docs$ echo "project documentation" > readme.txt
lakshay42@lakshay42-VirtualBox:~/docs$ echo "important notes" > notes.txt
lakshay42@lakshay42-VirtualBox:~/docs$ cat notes.txt
important notes
lakshay42@lakshay42-VirtualBox:~/docs$ cat readme.txt
project documentation
lakshay42@lakshay42-VirtualBox:~/docs$
```

## TASK 3: [File Operations]

### Task Statement:

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

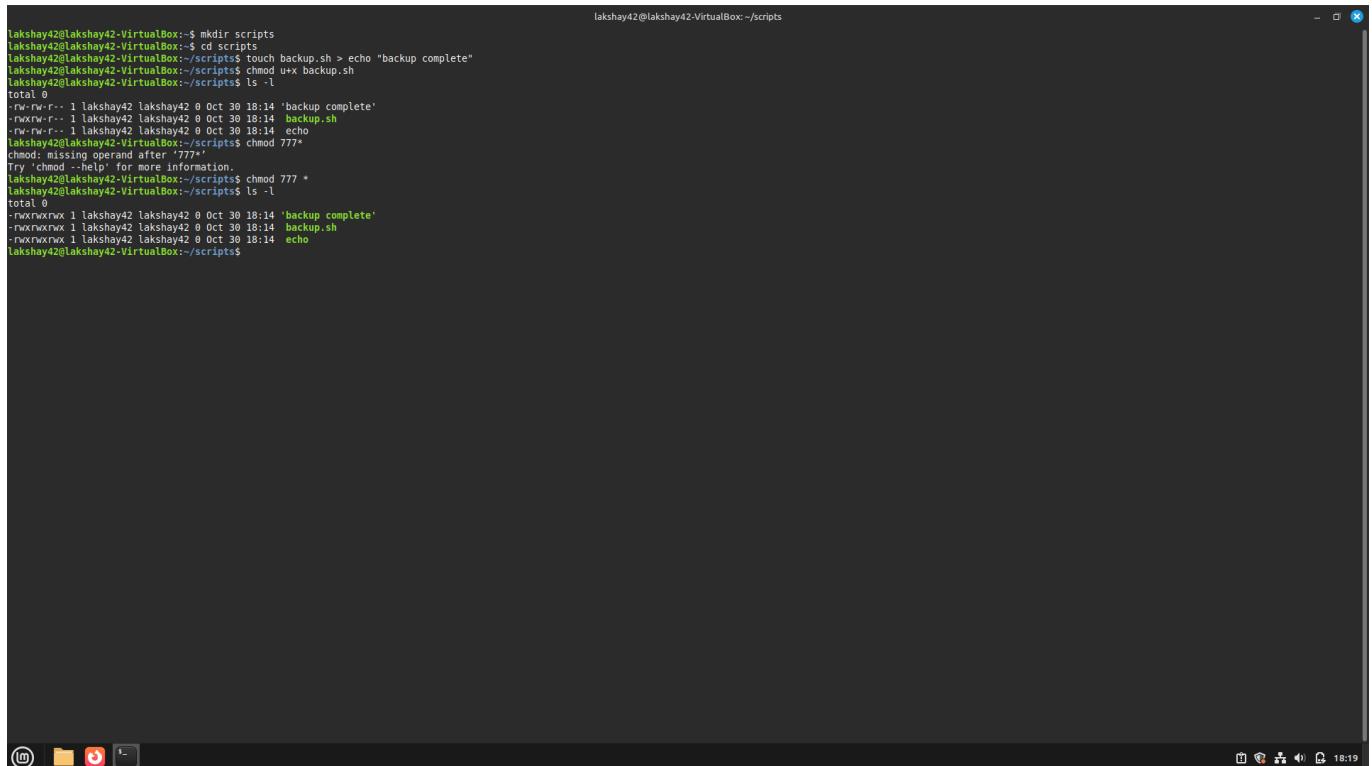
### Explanation:

- [- We can use the `cp` source destination to copy files and using the `mv oldname newname` to rename files also using the same command `mv` file directory/ to move files to another directory we can also combine copy and rename: `cp file.txt newdir/newname.txt`]

### Command(s):

```
cp readme.txt data/project_info.txt
```

### Output:



```

lakshay42@lakshay42-VirtualBox:~$ mkdir scripts
lakshay42@lakshay42-VirtualBox:~$ cd scripts
lakshay42@lakshay42-VirtualBox:~/scripts$ touch backup.sh > echo "Backup complete"
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod u+x backup.sh
lakshay42@lakshay42-VirtualBox:~/scripts$ ls -l
total 0
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 'backup complete'
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 backup.sh
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 echo
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod 777*
chmod: missing operand after `777*'
Try `chmod -help' for more information.
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod 777 *
lakshay42@lakshay42-VirtualBox:~/scripts$ ls -l
total 0
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 'backup complete'
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 backup.sh
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 echo
lakshay42@lakshay42-VirtualBox:~/scripts$
```

## TASK 4: [File Permissions]

### Task Statement:

- [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:

- [Using chmod u+x filename we can make the file executable for user only using ls -l to check for permissions also script files typically need executable permission to run]

### Command(s):

```

cd scripts
touch backup.sh > echo "Backup complete"
chmod u+x backup.sh

```

### Output:

```

lakshay42@lakshay42-VirtualBox:~$ mkdir scripts
lakshay42@lakshay42-VirtualBox:~$ cd scripts
lakshay42@lakshay42-VirtualBox:~/scripts$ touch backup.sh > echo "backup complete"
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod u+x backup.sh
lakshay42@lakshay42-VirtualBox:~/scripts$ ls -l
total 0
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 'backup complete'
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 backup.sh
-rw-rw-r-- 1 lakshay42 lakshay42 0 Oct 30 18:14 echo
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod 777
chmod: missing operand after `777'
Try `chmod --help' for more information.
lakshay42@lakshay42-VirtualBox:~/scripts$ chmod 777 *
lakshay42@lakshay42-VirtualBox:~/scripts$ ls -l
total 0
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 'backup complete'
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 backup.sh
-rwxrwxrwx 1 lakshay42 lakshay42 0 Oct 30 18:14 echo
lakshay42@lakshay42-VirtualBox:~/scripts$
```

## TASK 5: [File Viewing]

### Task Statement:

- [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:

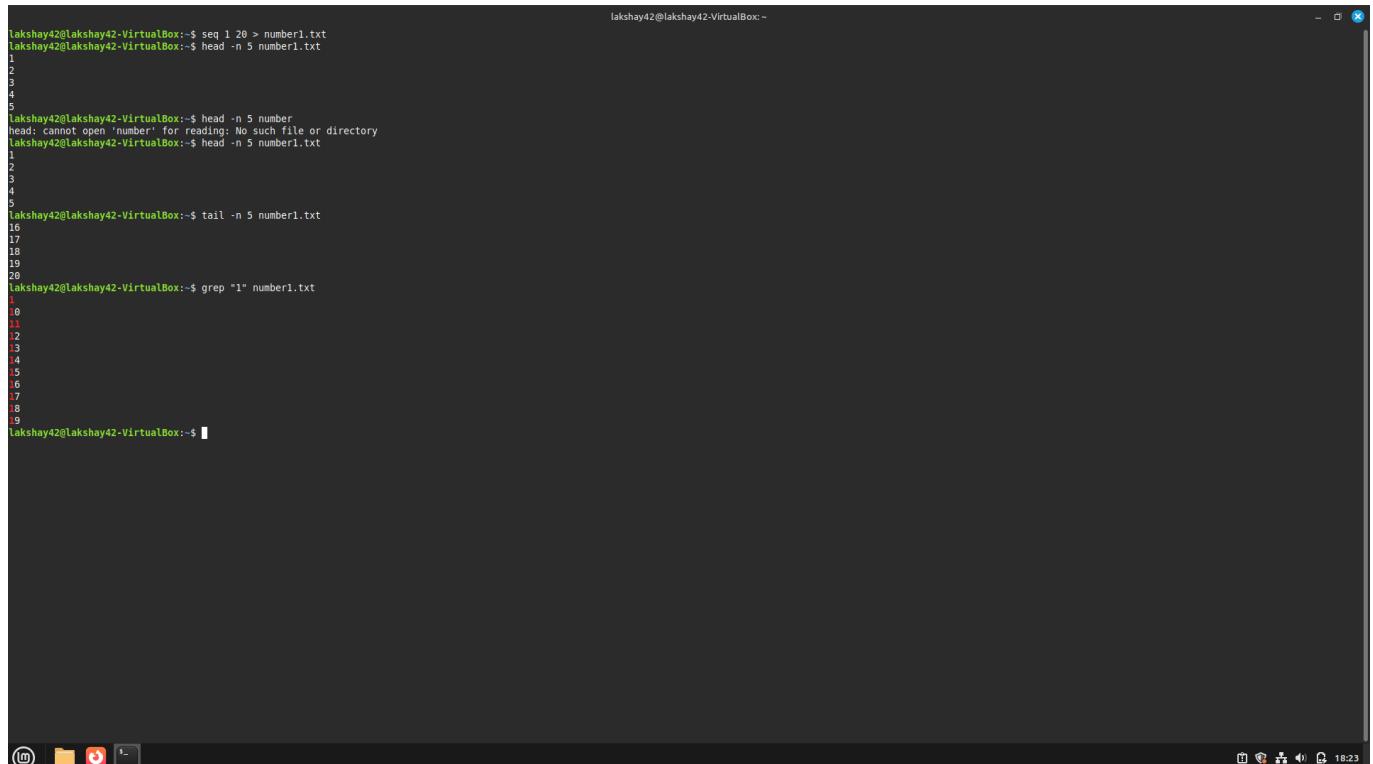
- [I can quickly generate a list of numbers by running seq 1 20 > numbers.txt. To check the first few numbers, I use head -n 5 to see the first 5 lines, and tail -n 3 to see the last 3 lines. If I want to find all numbers containing a "1", I can use grep "1". Alternatively, I could create the list manually by using multiple echo commands.]

### Command(s):

```

seq 1 20 > numbers.txt
head -n 5
tail -n 3
grep "1"
```

### Output:



```

lakshay42@lakshay42-VirtualBox:~$ seq 1 20 > number1.txt
lakshay42@lakshay42-VirtualBox:~$ head -n 5 number1.txt
1
2
3
4
5
lakshay42@lakshay42-VirtualBox:~$ head -n 5 number1.txt
head: cannot open 'number' for reading: No such file or directory
lakshay42@lakshay42-VirtualBox:~$ head -n 5 number1.txt
1
2
3
4
5
lakshay42@lakshay42-VirtualBox:~$ tail -n 5 number1.txt
16
17
18
19
20
lakshay42@lakshay42-VirtualBox:~$ grep "1" number1.txt
10
11
12
13
14
15
16
17
18
19
lakshay42@lakshay42-VirtualBox:~$ cat number1.txt
Database=localhost Port=5432 Username=admin

```

## TASK 6: [Text Editing]

### Task Statement:

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

Database=localhost Port=5432 Username=admin

Save the file and then display its contents.]

### Explanation:

- [I open a file in Nano using nano filename.txt and type my content normally. Once I'm done, I press Ctrl+O to save the file and Ctrl+X to exit Nano. After that, I use cat to check the contents and make sure everything was saved correctly.]

### Command(s):

```

vim config.txt
cat config.txt

```

### Alternatively

```
nano config.txt
cat config.txt
```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ touch config.txt
lakshay42@lakshay42-VirtualBox:~$ nano config.txt
lakshay42@lakshay42-VirtualBox:~$ cat config.txt
DHANDA
lakshay42@lakshay42-VirtualBox:~$
```

## TASK 7: [System Information]

### Task Statement:

- [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:

- [I can use whoami to check my username, date to see the current date, and pwd to know my current directory. To check disk usage, I use df -h. I can save the output of any command to a file by using redirection like command >> filename.txt. If I want to add labels, I use echo like this: echo "Username:" >> file.txt.]

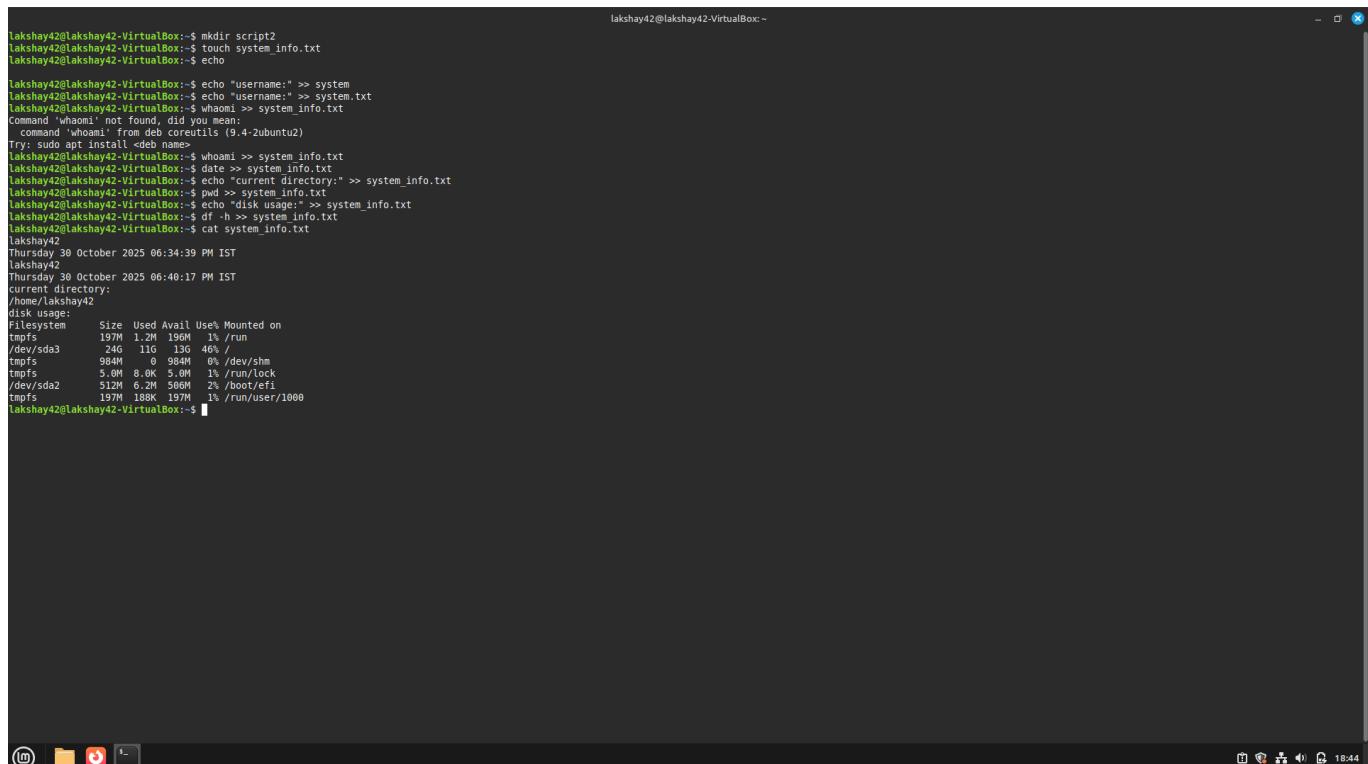
### Command(s):

```

cd scripts
touch system_info.txt
echo "Username:" >> system_info.txt
whoami >> system_info.txt
echo "Date:" >> system_info.txt
date >> system_info.txt
```

```
echo "Current Directory:" >> system_info.txt
pwd >> system_info.txt
echo "Disk Usage:" >> system_info.txt
df -h >> system_info.txt
```

## Output:



```

lakshay42@lakshay42-VirtualBox:~$ mkdir script2
lakshay42@lakshay42-VirtualBox:~$ touch system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo
lakshay42@lakshay42-VirtualBox:~$ echo "username:" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo "username" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo "password" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ command 'whomai' >> system_info.txt
Command 'whomai' not found, did you mean:
  command 'whomai' from deb coreutils 9.4-zubuntu2
Try: sudo apt install <deb name>
lakshay42@lakshay42-VirtualBox:~$ whomai >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ date >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo "Computer Model:" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo "System Info:" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ echo "disk usage:" >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ df -h >> system_info.txt
lakshay42@lakshay42-VirtualBox:~$ cat system_info.txt
lakshay42
Thursday 30 October 2025 06:34:39 PM IST
lakshay42
Thursday 30 October 2025 06:40:17 PM IST
current directory:
/home/lakshay42
disk usage:
Filesystem      Size   Used Avail Use% Mounted on
tmpfs          197M   1.2M  196M  1% /run
/dev/sda3       404G   10G  394G  3% /home
tmpfs          984M     0  984M  0% /dev/shm
tmpfs          5.0M   8.0K  5.0M  1% /run/lock
/dev/sda2       512M   6.2M  506M  2% /boot/efi
tmpfs          197M  188K  197M  1% /run/user/1000
lakshay42@lakshay42-VirtualBox:~$
```

## TASK 8: [File Organisation]

### Task Statement:

- [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:

- [I can use find . -name "\*.txt" to locate all .txt files. Alternatively, I can navigate to each directory and copy files manually. To copy multiple files at once, I use cp file1.txt file2.txt destination/. If I want detailed information about the files, I use ls -la. The wildcard \*.txt helps me match all files that end with .txt.]

### Command(s):

```
cp test_project/data/project_info.txt    test_project/docs/notes.txt
test_project/docs/readme.txt    test_project/docs/todo.txt
test_project/scripts/config.txt    test_project/scripts/numbers.txt
test_project/scripts/system_info.txt    test_project/scripts/todo.txt    backup/
```

## Output:

```

lakshay42@lakshay42-VirtualBox:~$ cp -r readme.txt -r todo.txt scripts/
lakshay42@lakshay42-VirtualBox:~$ ls -la
total 220
drwxr-xr-x 24 lakshay42 lakshay42 4096 Oct 30 18:53 .
drwxr-xr-x  3 root   root   4096 Aug 26 22:48 ..
-rwxr--r--x  1 lakshay42 lakshay42 16064 Sep 11 11:27 a.out
-rw-r----- 1 lakshay42 lakshay42 1211 Oct 30 18:08 .bash_history
-rw-r--r--  1 lakshay42 lakshay42 228 Aug 26 22:48 .bash_logout
-rw-r--r--  1 lakshay42 lakshay42 3771 Aug 26 22:48 .bashrc
drwxr-xr-x 12 lakshay42 lakshay42 4096 Sep 30 17:14 .cache
drwxr-xr-x 10 lakshay42 lakshay42 4096 Sep 23 18:28 .config
drwxr-xr-x  2 lakshay42 lakshay42 25 Aug 26 22:48 .config.txt
drwxr-xr-x  2 lakshay42 lakshay42 4096 Aug 29 16:48 Desktop
-rw-r--r--  1 lakshay42 lakshay42 27 Aug 26 23:09 .dmrc
drwxrwxr-x  2 lakshay42 lakshay42 4096 Oct 30 18:10 docs
drwxr-xr-x  2 lakshay42 lakshay42 4096 Sep 23 17:03 Documents
drwxr-xr-x  2 lakshay42 lakshay42 4096 Aug 26 23:09 Downloads
drwxrwxr-x  1 lakshay42 lakshay42 4096 Sep 23 16:39 .empty
drwxr-xr-x  1 lakshay42 lakshay42 25 Aug 26 22:48 .gtkrc-2.0
-rw-r--r--  1 lakshay42 lakshay42 516 Aug 26 22:48 .gtkrc-xfce
drwxrwxr-x  2 lakshay42 lakshay42 4096 Sep 12 16:22 ;linux command 3'
-rw-r----- 1 lakshay42 lakshay42 28 Sep 5 16:38 .lessht
drwxrwxr-x  2 lakshay42 lakshay42 4096 Sep 12 16:23 ;linux command 1'
drwxrwxr-x  3 lakshay42 lakshay42 4096 Sep 12 16:23 ;linux command 2'
drwxrwxr-x  4 lakshay42 lakshay42 4096 Aug 26 23:09 .local
drwxr-xr-x  4 lakshay42 lakshay42 4096 Sep 12 16:48 .mozilla
drwxrwxr-x  2 lakshay42 lakshay42 4096 Aug 26 23:09 Music
-rw-rw-r--  1 lakshay42 lakshay42 51 Oct 30 18:08 number1.txt
drwxr-xr-x  2 lakshay42 lakshay42 4096 Sep 30 17:12 Pictures
-rw-r--r--  1 lakshay42 lakshay42 807 Aug 26 22:48 .profile
drwxrwxr-x  1 lakshay42 lakshay42 4096 Aug 26 23:09 Public
drwxrwxr-x  1 lakshay42 lakshay42 4096 Aug 26 23:09 .public
drwxrwxr-x  2 lakshay42 lakshay42 4096 Oct 30 18:37 scripts
drwxrwxr-x  2 lakshay42 lakshay42 4096 Oct 30 18:55 scripts
drwxrwxr-x  2 lakshay42 lakshay42 4096 Oct 30 18:29 scripts2
-rw-rw-r--  1 lakshay42 lakshay42 159 Sep 30 17:23 script.sh
-rw-r--r--  1 lakshay42 lakshay42 6 Sep 5 17:08 .sudo_as_admin_successful
drwxrwxr-x  1 lakshay42 lakshay42 4096 Oct 30 18:53 system
drwxrwxr-x  1 lakshay42 lakshay42 477 Oct 30 18:43 .system.info.txt
-rw-rw-r--  1 lakshay42 lakshay42 28 Oct 30 18:38 system.txt
drwxrwxr-x  2 lakshay42 lakshay42 4096 Aug 26 23:09 Templates
-rw-rw-r--  1 lakshay42 lakshay42 143 Sep 11 11:22 test.c
drwxrwxr-x  5 lakshay42 lakshay42 4096 Oct 30 17:50 test_project
-rw-rw-r--  1 lakshay42 lakshay42 28 Oct 30 18:53 todo.txt
-rw-rw-r--  1 (lakshay42) 0 Aug 29 11:48 .txt
drwxrwxr-x  2 lakshay42 lakshay42 4096 Oct 30 18:09 .vims
drwxrwxr-x  1 lakshay42 lakshay42 11075 Sep 30 17:23 .viminfo
-rw-rw-r--  1 lakshay42 lakshay42 1933 Sep 5 17:09 .wsl-config
-rw-r----- 1 lakshay42 lakshay42 65 Oct 29 11:27 .xauthority
-rw-r----- 1 lakshay42 lakshay42 8904 Oct 29 11:41 .xsession-errors
-rw-r----- 1 lakshay42 lakshay42 8715 Oct 2 20:19 .xsession-errors.old
-rw-rw-r--  1 (lakshay42) 0 Aug 29 16:54 xyz
-rw-rw-r--  1 (lakshay42) 0 Sep 2 22:11 xyz.txt
lakshay42@lakshay42-VirtualBox:~$ 

```

(Minimize, Maximize, Close, Taskbar icons, 18:56)

## TASK 9: [Process and History]

### Task Statement:

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

### Explanation:

- [I can use history to see all the commands I've typed. To count the total number of commands, I use history | wc -l. If I want to view just the last 10 commands, I can use history 10 or history | tail -10. The wc -l command simply counts the number of lines in the output.]

### Command(s):

```
history 10
```

## Output:

```

lakshay42@lakshay42-VirtualBox:~$ history 20
172 echo "disk usage:" >> system_info.txt
173 df -h >> system_info.txt
174 cat system_info.txt
175 cd ~
176 clear
177 cd -> README.txt -r todo.txt script/
178 cd -> README.txt -r todo.txt script/
179 cd ~
180 clear
181 echo "todo list goes here" > todo.txt
182 echo "todo list goes here" > todo.txt
183 cd -> README.txt -r todo.txt script/
184 cp -r README.txt -r todo.txt script/
185 cp -r README.txt -r todo.txt script/
186 cp -r README.txt -r todo.txt scripts/
187 ls -la
188 cd ~
189 clear
190 history 20
lakshay42@lakshay42-VirtualBox:~$ 
```

## TASK 10: [Comprehensive Cleanup]

### Task Statement:

- [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:

- [I can set permissions for backup.sh using chmod 754 backup.sh to give rwxr-xr-- permissions. Alternatively, I can use chmod u=rwx,g=rx,o=r backup.sh. To count all files, I use find . -type f | wc -l, and to count directories, I use find . -type d | wc -l. If I want to see the full directory structure recursively, I use ls -R. I can also combine multiple commands with && or save the outputs to a summary file for later reference.]

### Command(s):

```

chmod 754 backup.sh

echo "Total files:" > summary.txt
find . -type f | wc -l >> summary.txt
echo "Total directories:" >> summary.txt
find . -type d | wc -l >> summary.txt 
```

### Output:

```
lakshay42@lakshay42-VirtualBox:~$ echo "total files:" > summary.txt
lakshay42@lakshay42-VirtualBox:~$ find . -type f | wc -l >> summary.txt
lakshay42@lakshay42-VirtualBox:~$ echo "total directoresies:" >> summary.txt
bash: .. Is a directory
lakshay42@lakshay42-VirtualBox:~$ echo "total directoresies:" >> summary.txt
lakshay42@lakshay42-VirtualBox:~$ find . -type d | wc -l >> summary.txt
lakshay42@lakshay42-VirtualBox:~$ cat summary.txt
total files:
1733
total directoresies:
466
lakshay42@lakshay42-VirtualBox:~$
```

# Experiment 3: Linux File Manipulation and System Manipulation I

Name: lakshay dhanda Roll No.: 590029328 Date: 30-10-2025

Aim:

- To practice Linux file manipulation commands like `touch`, `cp`, `mv`, `rm`, `cat`, `less`, `head`, `tail`.
- To explore file permissions and ownership with `ls -l`, `chmod`, `chown`, and `chgrp`.
- To search and filter files using `find` and `grep`.
- To understand archiving and compression with `tar`, `gzip`, and `gunzip`.
- To create and manage links (`ln`) for both hard and symbolic links.

## Requirements

- A Linux machine with bash shell (Ubuntu/Fedora/other).
- User privileges to create, modify, and delete files and directories.
- Access to system utilities like `tar`, `gzip`, `grep`, and `find`.

## Theory

Linux file management involves creating, copying, moving, removing, and viewing files. File permissions and ownership ensure secure access control. Searching and filtering tools like `grep` and `find` help locate information efficiently. Archiving with `tar` and compression with `gzip` reduce storage usage and simplify file transfer. Links (`ln`) allow multiple references to the same file data (hard links) or path references (symbolic links).

## Procedure & Observations

### Exercise 1: Creating and Managing Files

Task Statement:

Create files and manage timestamps using `touch`.

Command(s):

```
touch newfile.txt
touch file1.txt file2.txt file3.txt
touch -t 202401151430 dated_file.txt
```

Output:



### Exercise 2: Copying, Moving, and Deleting Files

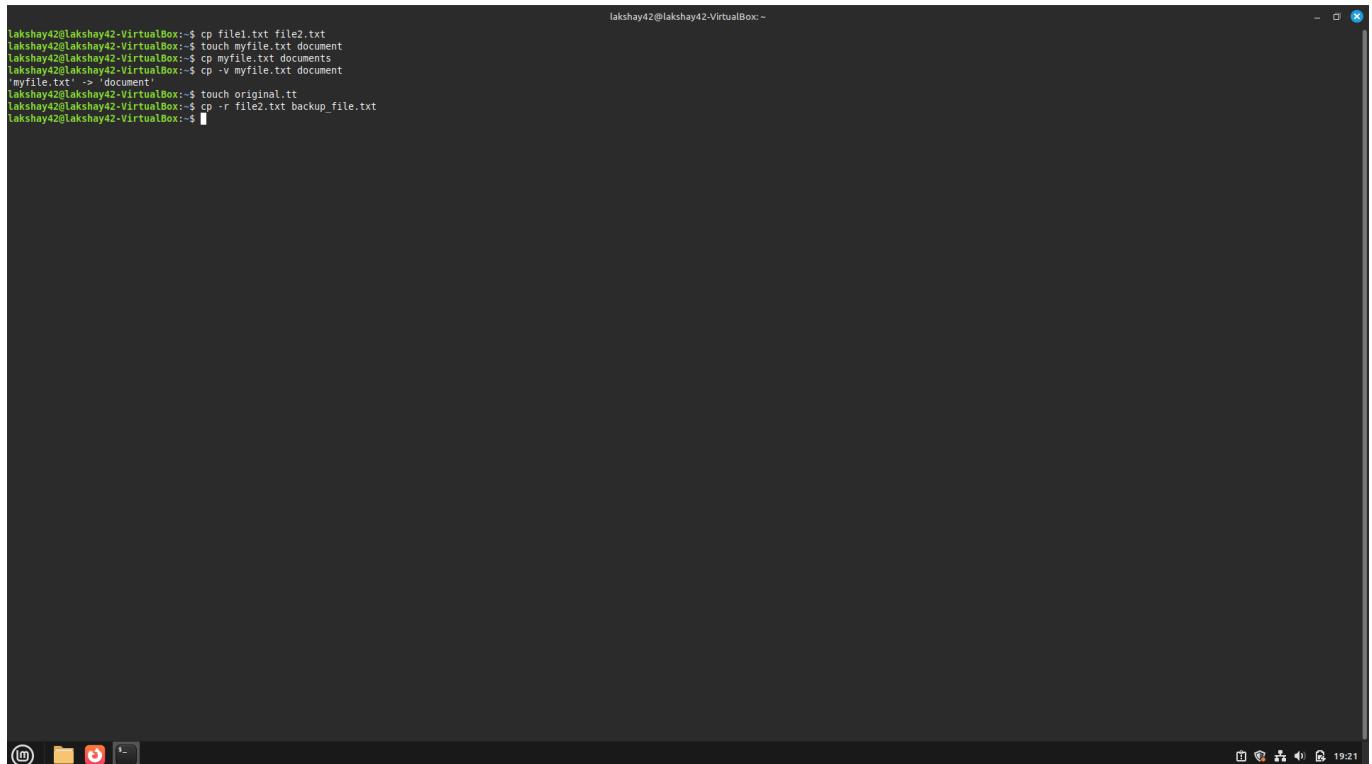
Task Statement:

Use `cp`, `mv`, and `rm` to copy, rename, move, and delete files and directories.

Command(s):

```
cp document.txt backup_document.txt
mv oldname.txt newname.txt
rm unwanted_file.txt
rm -r old_directory/
```

Output:



```

lakshay42@lakshay42-VirtualBox:~$ cp file1.txt file2.txt
lakshay42@lakshay42-VirtualBox:~$ touch myfile.txt document
lakshay42@lakshay42-VirtualBox:~$ cp myfile.txt documents
lakshay42@lakshay42-VirtualBox:~$ cp -v myfile.txt document
myfile.txt -> 'document'
lakshay42@lakshay42-VirtualBox:~$ touch original_tt
lakshay42@lakshay42-VirtualBox:~$ cp -r file2.txt backup_file.txt
lakshay42@lakshay42-VirtualBox:~$
```

## Exercise 3: Viewing File Contents

Task Statement:

Display file contents using `cat`, `less`, `head`, and `tail`.

Command(s):

```
cat filename.txt
less /var/log/syslog
head -n 5 filename.txt
tail -n 20 filename.txt
tail -f /var/log/syslog
```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ rm myfile.txt
\\lakshay42@lakshay42-VirtualBox:~$ ls
a.out      backup original.txt  dated file.txt  document  Downloads  file2.txt      'Linux command'  Music    original..tt  Pictures   script2  script.sh  system.info.txt  test.c  Videos
\.\out     backup config.txt   Desktop   documents 'exp 8'    file3.txt  'Linux command 1' newfile.txt  original.tt  Public    scripts  summary.txt  system.txt  test_project  xyz
backup file.t  dated          docs       Documents  file1.txt  'linux command 3'  number1.txt  original.txt  readme.txt  scripts2 system    Templates  todo.txt   xyz.txt
lakshay42@lakshay42-VirtualBox:~$
```

## Exercise 4: File Permissions and Ownership

Task Statement:

Explore file permissions and ownership with `ls -l`, `chmod`, `chown`, and `chgrp`.

Command(s):

```

ls -l
chmod 755 script.sh
chmod u+x script.sh
sudo chown newuser:newgroup file.txt
chgrp developers project.txt

```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ mkdir dir1
lakshay42@lakshay42-VirtualBox:~$ touch dir1/file3.txt
lakshay42@lakshay42-VirtualBox:~$ mv file3 dir1
mv: cannot stat 'file3': No such file or directory
lakshay42@lakshay42-VirtualBox:~$ ls dir1
file3.txt
lakshay42@lakshay42-VirtualBox:~$ mkdir file4
lakshay42@lakshay42-VirtualBox:~$ ls dir2
ls: cannot access 'dir2': No such file or directory
lakshay42@lakshay42-VirtualBox:~$ mkdir dir2
lakshay42@lakshay42-VirtualBox:~$ ls dir2
lakshay42@lakshay42-VirtualBox:~$ touch file.txt
lakshay42@lakshay42-VirtualBox:~$ mkdir file1
lakshay42@lakshay42-VirtualBox:~$ mv file1 touch.txt
lakshay42@lakshay42-VirtualBox:~$ ls
a.out      config.txt  dir1      documents  file1.txt  file.txt  'linux command 2'  original..tt  Public    scripts2   system.info.txt  test_project  xyz
backup_file.t  dated     dir2      Documents  file2.txt  ':linux command 3'  Music      original.tt  readme.txt  script.sh  system.txt  todo.txt   xyz.txt
backup_file.txt  dated     docs      Downloads  file3.txt  ':linux command'  newfile.txt  original.txt  script2   summary.txt  Templates   touch.txt
backup_originaltxt  Desktop  document  'exp 8'   file4     'linux command 1'  number1.txt  Pictures   scripts   system     test.c    Videos
lakshay42@lakshay42-VirtualBox:~$ mv dir1 dir_1
lakshay42@lakshay42-VirtualBox:~$ ls
dir_1      documents  file1.txt  file.txt  'linux command 2'  original..tt  Public    scripts2   system.info.txt  test_project  xyz
back_up_file.t  dated     dir_1    Documents  file2.txt  ':linux command 3'  Music      original.tt  readme.txt  script.sh  system.txt  todo.txt   xyz.txt
back_up_file.txt  dated     docs      Downloads  file3.txt  ':linux command'  newfile.txt  original.txt  script2   summary.txt  Templates   touch.txt
backup_originaltxt  Desktop  document  'exp 8'   file4     'linux command 1'  number1.txt  Pictures   scripts   system     test.c    Videos
lakshay42@lakshay42-VirtualBox:~$ 

```

The terminal window shows a series of commands being run on a Linux system. It starts by creating a directory 'dir1' and a file 'file3.txt' in it. Then, it attempts to move 'file3' to 'dir1' but fails because 'file3' does not exist. Next, it creates a directory 'file4'. It then tries to list 'dir2' but fails because 'dir2' does not exist. It creates 'dir2' and then lists its contents, which include 'file.txt'. It then creates 'file1' and moves it to 'touch.txt'. Finally, it renames 'dir1' to 'dir\_1' and lists its contents again.

## Exercise 5: File Searching with `find`

Task Statement:

Search files by name, type, size, and permissions using `find`.

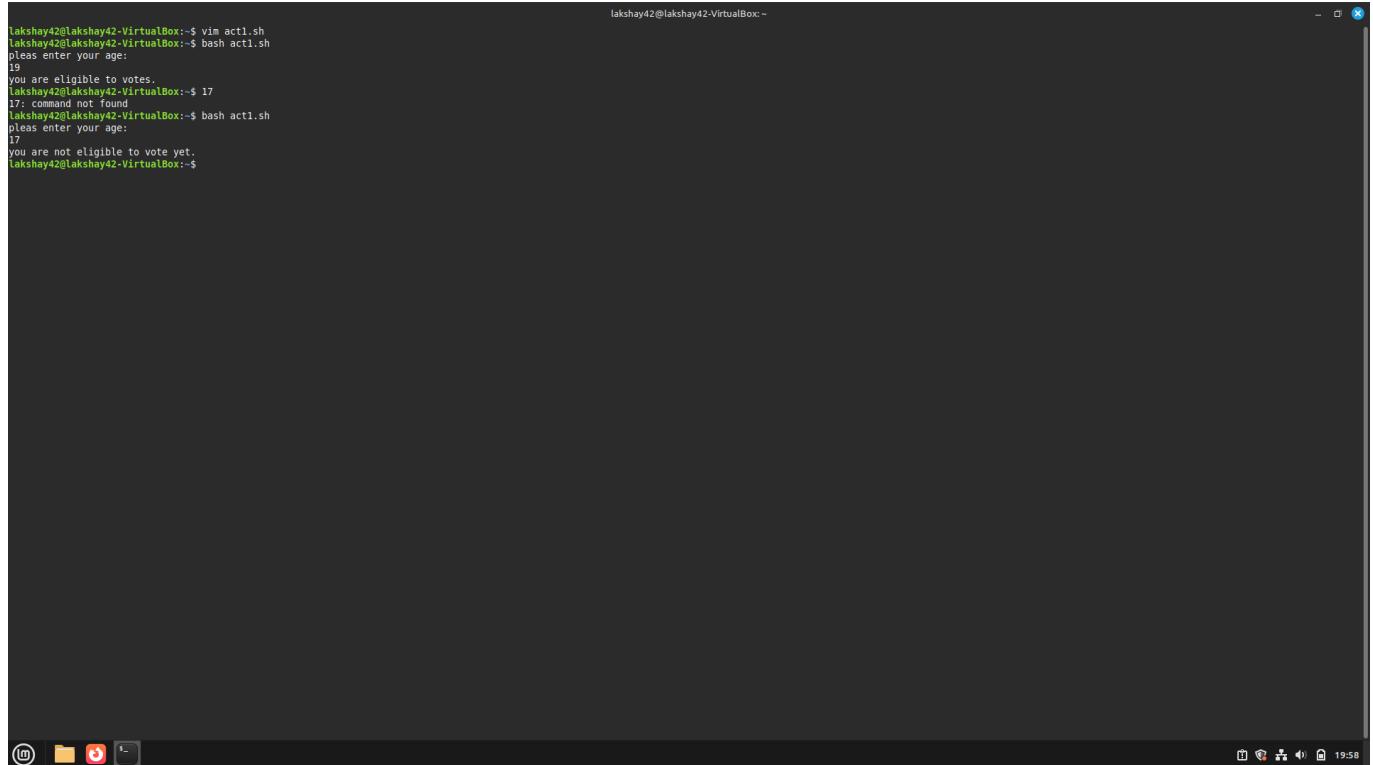
Command(s):

```

find /home -name "*.txt"
find /home -type f -size +100M
find /etc -name "*conf*"
find /tmp -type f -empty -delete

```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ vim act1.sh
lakshay42@lakshay42-VirtualBox:~$ bash act1.sh
please enter your age:
19
you are eligible to votes.
lakshay42@lakshay42-VirtualBox:~$ 17: command not found
lakshay42@lakshay42-VirtualBox:~$ bash act1.sh
please enter your age:
17
you are not eligible to vote yet.
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 6: Pattern Searching with grep

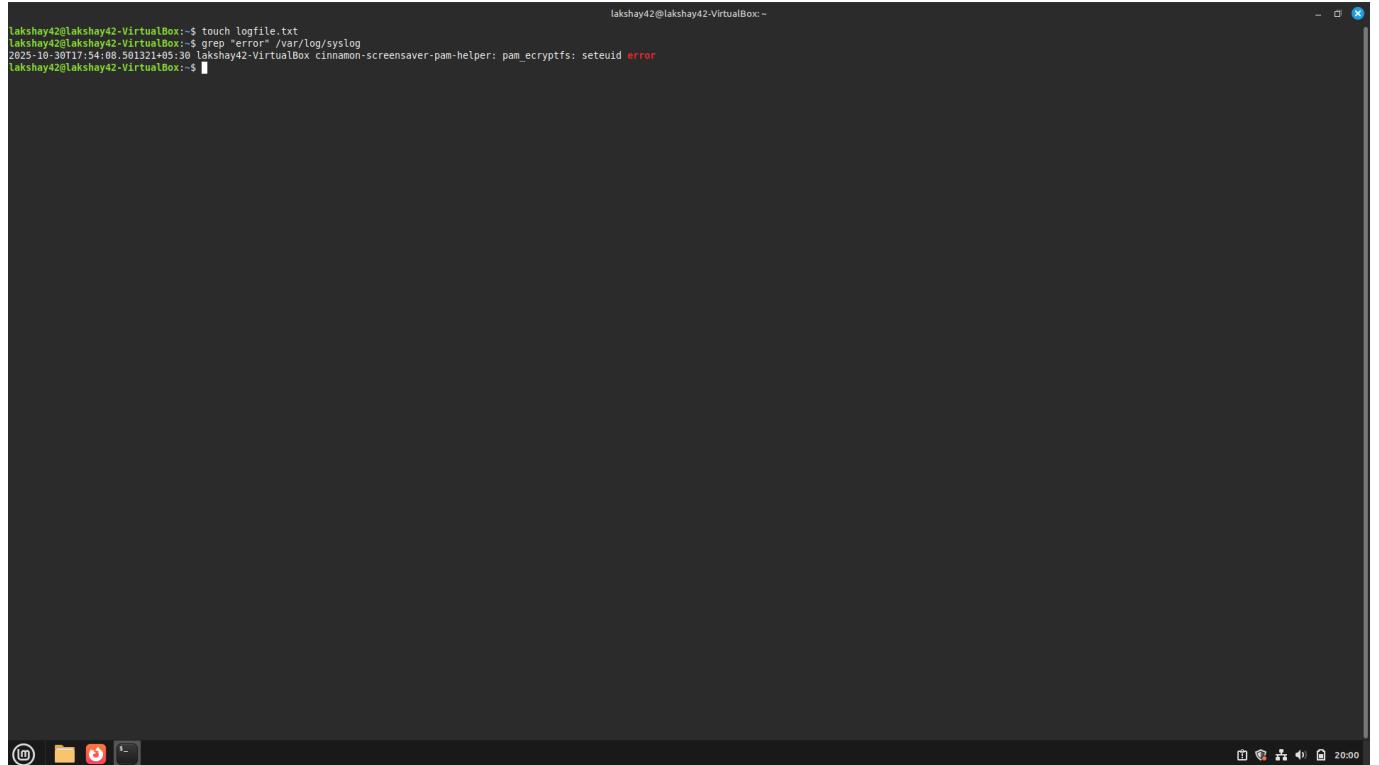
Task Statement:

Search for patterns in files using **grep**.

Command(s):

```
grep "error" /var/log/syslog
grep -i "Error" logfile.txt
grep -r "function" ~/code/
grep -n "TODO" *.txt
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ touch logfile.txt
lakshay42@lakshay42-VirtualBox:~$ grep "error" /var/log/syslog
2025-10-30T17:54:08.591321+05:30 lakshay42-VirtualBox cinnamon-screensaver-pam-helper: pam_ecryptfs: seteuid error
lakshay42@lakshay42-VirtualBox:~$
```

The screenshot shows a terminal window with a dark background. At the top, there's a header bar with the terminal title 'lakshay42@lakshay42-VirtualBox:~'. Below the header, the terminal displays a command-line session. The user runs 'touch logfile.txt' followed by 'grep "error" /var/log/syslog'. The output shows a single line from the syslog: '2025-10-30T17:54:08.591321+05:30 lakshay42-VirtualBox cinnamon-screensaver-pam-helper: pam\_ecryptfs: seteuid error'. The bottom of the window shows the standard Linux desktop interface with icons for file, folder, and system settings, along with a system tray showing battery level and time.

---

## Exercise 7: Archiving and Compression

Task Statement:

Create and extract archives using **tar**, compress and decompress with **gzip/gunzip**.

Command(s):

```
tar -czf backup.tar.gz /home/user/documents
tar -xzf backup.tar.gz -C /restore/
gzip largefile.txt
gunzip largefile.txt.gz
```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ gunzip hello.gz
gzip: hello.gz: No such file or directory
lakshay42@lakshay42-VirtualBox:~$ tar -czf archive.tar.gz *
lakshay42@lakshay42-VirtualBox:~$ ls
act1.sh      backup file.txt  dated file.txt  docs    Downloads  file3.txt      'Linux command'  Music   original.tt  readme.txt  script.sh  system.txt  todo.txt  xyz.txt
a.out        backup_originaltxt Desktop  document 'exp 8'   file4       'linux command 1'  newfile.txt  original.txt  script2    summary.txt  Templates  touch.txt
archive.tar.gz config.txt     dir_1      documents file1.txt  file.txt    'linux command 2'  number1.txt  Pictures   scripts   system     test.c    Videos
backup_file.t dated file.txt  dir2      Documents file2.txt  file3.txt  'linux command 3'  logfile.txt original.tt  Public    scripts2  system_info.txt test_project  xyz
lakshay42@lakshay42-VirtualBox:~$ ls
act1.sh      backup file.txt  dated file.txt  docs    Downloads  file3.txt      'Linux command'  Music   original.tt  readme.txt  script.sh  system.txt  todo.txt  xyz.txt
a.out        backup_originaltxt Desktop  document 'exp 8'   file4       'linux command 1'  newfile.txt  original.txt  script2    summary.txt  Templates  touch.txt
archive.tar.gz config.txt     dir_1      documents file1.txt  file.txt    'linux command 2'  number1.txt  Pictures   scripts   system     test.c    Videos
backup_file.t dated file.txt  dir2      Documents file2.txt  file3.txt  'linux command 3'  logfile.txt original.tt  Public    scripts2  system_info.txt test_project  xyz
lakshay42@lakshay42-VirtualBox:~$ 

```

The terminal window shows a series of commands being run. It starts with an attempt to gunzip a non-existent file 'hello.gz'. Then it creates a tar archive named 'archive.tar.gz' from the current directory. After creating the archive, it lists the contents of the directory again. The directory contains several files and sub-directories, including 'act1.sh', 'a.out', 'archive.tar.gz', 'backup\_file.t', and various configuration and backup files. The terminal window has a dark theme and includes standard Linux window controls at the top and bottom.

## Exercise 8: Creating Links

Task Statement:

Create and test hard and symbolic links using `ln`.

Command(s):

```

echo "Hello" > original.txt
ln original.txt hardlink.txt
ln -s original.txt symlink.txt
ls -li original.txt hardlink.txt symlink.txt

```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ ln file.txt hardlink_file
lakshay42@lakshay42-VirtualBox:~$ ls -l
total 212
1117222 -rw-rw-r-- 1 lakshay42 lakshay42 148 Oct 30 19:57 act1.sh
1049061 -rwxrwxr-x 1 lakshay42 lakshay42 16064 Sep 11 21:27 a.out
1117223 -rw-rw-r-- 1 lakshay42 lakshay42 68328 Oct 30 20:09 archive.tar.gz
1117219 -rw-rw-r-- 1 lakshay42 lakshay42 8 Oct 30 19:19 backup_file.txt
1117218 -rwxrwxr-x 1 lakshay42 lakshay42 0 Oct 30 19:16 backup_file.txt
1117217 -rwxrwxr-x 1 lakshay42 lakshay42 0 Oct 30 19:16 backup_originaltxt
1117208 -rwxrwxr-- 1 lakshay42 lakshay42 7 Oct 30 18:28 config.txt
1117210 -rwxrwxr-- 1 lakshay42 lakshay42 0 Jan 15 2024 dated
1117211 -rwxrwxr-- 1 lakshay42 lakshay42 0 Jan 15 2024 dated_file.txt
1194074 drwxr-xr-x 2 lakshay42 lakshay42 4096 Aug 29 16:48 Desktop
1368711 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 19:46 dir_1
1368712 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 19:46 dir_2
1368651 -rwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 18:10 docs
1117213 -rwxrwxr-x 1 lakshay42 lakshay42 8 Oct 30 19:20 document
1117214 -rwxrwxr-x 1 lakshay42 lakshay42 8 Oct 30 19:20 documents
1194078 drwxr-xr-x 2 lakshay42 lakshay42 4096 Sep 23 17:03 Documents
1194075 drwxr-xr-x 2 lakshay42 lakshay42 4096 Aug 26 23:09 Downloads
1317588 drwxrwxr-x 2 lakshay42 lakshay42 4096 Sep 23 16:34 exp_8
1117207 -rwxrwxr-x 2 lakshay42 lakshay42 0 Oct 30 19:20 file1.txt
1117286 -rwxrwxr-x 1 lakshay42 lakshay42 0 Oct 30 19:20 file2.txt
1117289 -rwxrwxr-- 1 lakshay42 lakshay42 0 Oct 30 19:40 file3.txt
1368711 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 19:47 file4
1117212 -rwxrwxr-- 2 lakshay42 lakshay42 0 Oct 30 19:49
1117212 -rwxrwxr-- 2 lakshay42 lakshay42 0 Oct 30 19:49 hardlink_file
1202675 drwxrwxr-x 2 lakshay42 lakshay42 4096 Sep 12 16:23 ;linux command 3'
1368706 drwxrwxr-x 2 lakshay42 lakshay42 4096 Sep 12 16:23 ;linux command 1'
1199748 drwxrwxr-x 2 lakshay42 lakshay42 4096 Sep 12 17:16 linux command 2'
1050560 -rwxrwxr-- 1 lakshay42 lakshay42 0 Oct 30 19:58 logfile.txt
1194079 drwxr-xr-x 2 lakshay42 lakshay42 4096 Aug 26 23:09 Music
1117206 -rwxrwxr-- 1 lakshay42 lakshay42 0 Oct 30 19:11 newfile.txt
1049062 -rwxrwxr-- 1 lakshay42 lakshay42 51 Oct 30 18:28 number1.txt
1117205 -rwxrwxr-x 1 lakshay42 lakshay42 0 Oct 30 19:21 original.txt
1117213 -rwxrwxr-- 1 lakshay42 lakshay42 0 Oct 30 19:16 original.txt
1117216 -rwxrwxr-x 2 lakshay42 lakshay42 4096 Sep 30 17:12 Pictures
1194077 drwxr-xr-x 2 lakshay42 lakshay42 4096 Aug 26 23:09 Public
1050558 -rwxrwxr-x 1 lakshay42 lakshay42 0 Sep 5 15:12 readme.txt
1368706 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 18:37 script2
1368694 drwxrwxr-x 1 lakshay42 lakshay42 4096 Oct 30 18:29 scripts
1368713 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 18:29 script2
1050554 -rwxrwxr-- 1 lakshay42 lakshay42 159 Sep 30 17:23 script.sh
1117205 -rwxrwxr-- 1 lakshay42 lakshay42 41 Oct 30 19:01 summary.txt
1117202 -rwxrwxr-- 1 lakshay42 lakshay42 28 Oct 30 18:37 system
1117201 -rwxrwxr-- 1 lakshay42 lakshay42 477 Oct 30 18:43 system.info.txt
1117203 -rwxrwxr-- 1 lakshay42 lakshay42 28 Oct 30 18:38 system.txt
1050536 -rwxrwxr-x 1 lakshay42 lakshay42 4096 Aug 26 23:09 Templates
1050562 -rwxrwxr-x 1 lakshay42 lakshay42 143 Oct 30 19:22 test
1368690 drwxrwxr-x 5 lakshay42 lakshay42 4096 Oct 30 17:58 test_project
1117204 -rwxrwxr-- 1 lakshay42 lakshay42 20 Oct 30 18:53 todo.txt
1368713 drwxrwxr-x 2 lakshay42 lakshay42 4096 Oct 30 19:49 touch.txt
1194082 drwxr-xr-x 2 lakshay42 lakshay42 4096 Aug 26 23:09 Videos
1050566 -rwxrwxr-- 1 lakshay42 lakshay42 0 Aug 29 16:54 xyz
1050562 -rwxrwxr-- 1 lakshay42 lakshay42 0 Sep 2 22:11 xyz.txt
lakshay42@lakshay42-VirtualBox:~$
```

## Result

- Successfully created, copied, moved, and deleted files.
- Practiced viewing file contents and monitoring logs.
- Explored file permissions and ownership management.
- Used `find` and `grep` to locate and filter data.
- Created archives and compressed files.
- Demonstrated both hard and symbolic links.

## Challenges Faced & Learning Outcomes

- Challenge 1: Accidentally deleted files with `rm` without `-i`. Learned to use `rm -i` for safety.
- Challenge 2: Remembering numeric vs symbolic permissions in `chmod`. Fixed through repeated practice.

### Learning:

- Gained practical skills with file manipulation and permission commands.
- Learned how to efficiently search files and patterns in Linux.
- Understood how to archive and compress files for better storage management.
- Understood differences between hard and symbolic links.

## Conclusion

This experiment provided hands-on experience with core Linux file management, permissions, searching, archiving, and linking. These are foundational skills for effective Linux system administration and daily usage.

## Experiment [4]: [Bash Scripting]

Name:lakshay dhanda, Roll No.: 590029328, Date: 30-10-2025

AIM:

- [To Learn Basics of Bash Scripting.]

Requirements:

- [Any Linux Distro, any kind of text editor (vs code, vim, notepad, nano, etc)]

Theory:

- [Learning the basics of bash scripting.]

## Procedure & Observations

### Exercise 1: [Hello World Script]

Task Statement:

- [Basic Usage of Shell Scripts]

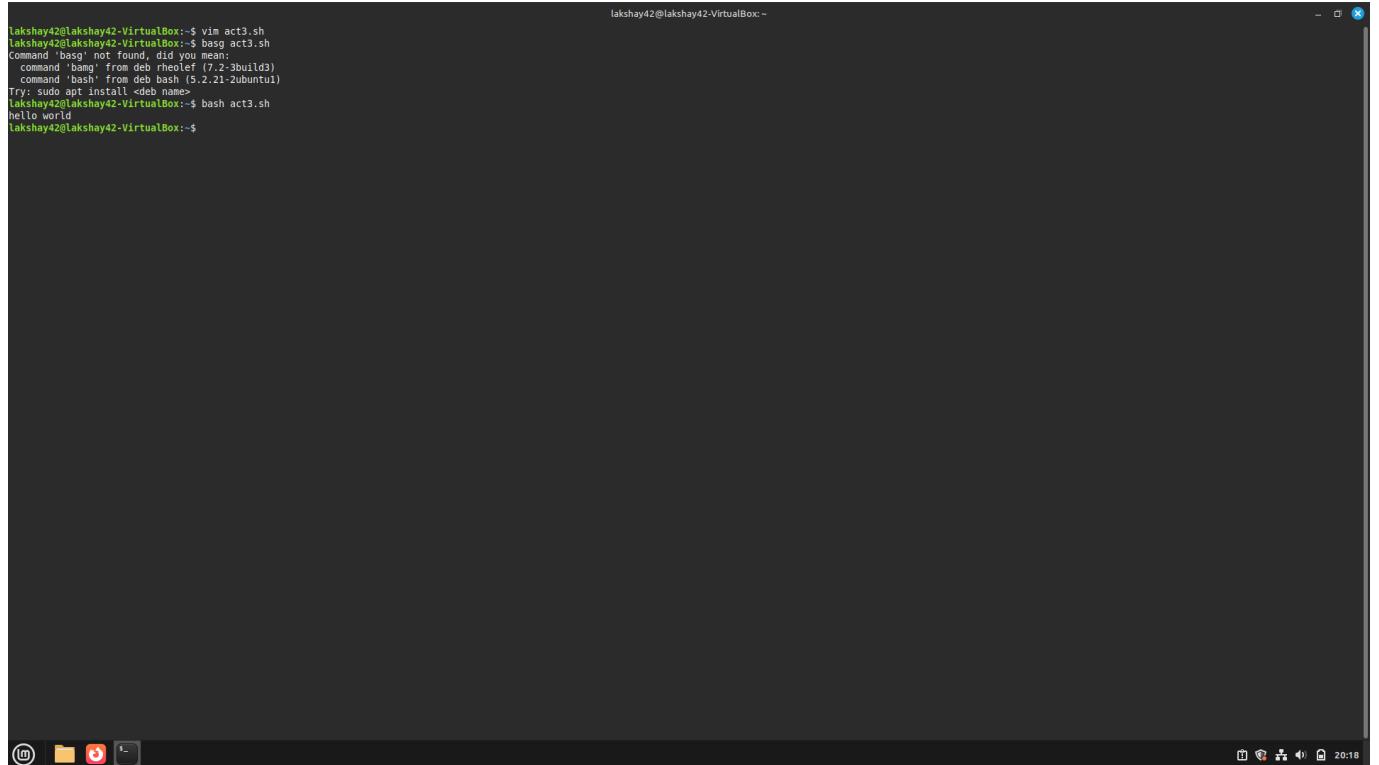
Explanation:

- [Writing Begginer level Shell Scripts]

Command(s):

```
#!/bin/bash
echo "Hello, World!"
```

Output:



A screenshot of a Linux terminal window titled "lakshay42@lakshay42-VirtualBox: ~". The terminal shows the following command and its output:

```

lakshay42@lakshay42-VirtualBox:~$ vim act3.sh
lakshay42@lakshay42-VirtualBox:~$ basg act3.sh
Command 'basg' not found, did you mean:
  command 'bamg' from deb rheolef (7.2-3build3)
  command 'bash' from deb bash (5.2.21-2ubuntu1)
Try: sudo apt install <deb name>
lakshay42@lakshay42-VirtualBox:~$ bash act3.sh
Hello World
lakshay42@lakshay42-VirtualBox:~$
```

The terminal window has a dark background and light-colored text. The bottom of the window shows the desktop environment's taskbar with icons for file, folder, and system status.

## Exercise 2: [Personalized Greeting Script]

### Task Statement:

- [Basic Shell Script to callout user defined function.]

### Explanation:

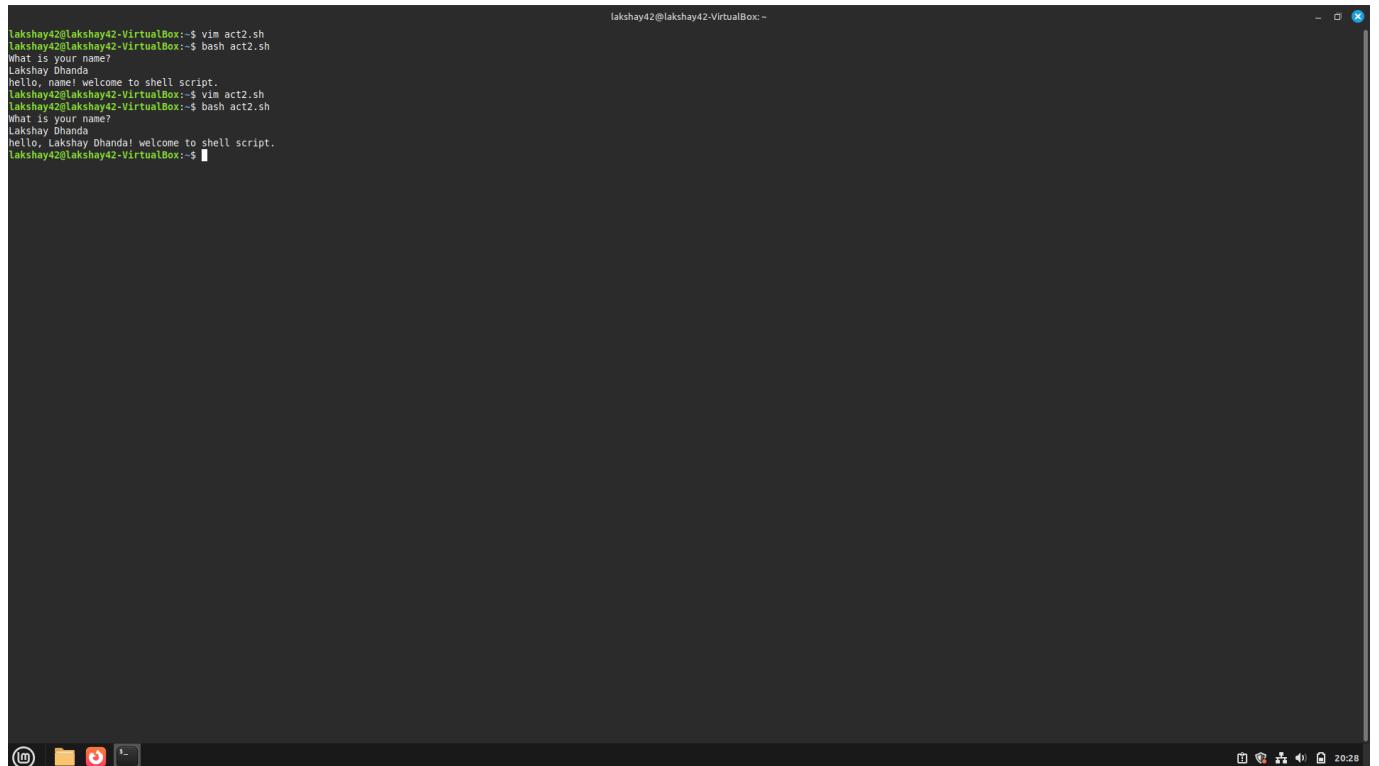
- [This Shell script will take input from user and store it in a variable and then call the variable which will output the stored value.]

### Command(s):

```

#!/bin/bash
echo "What is your name?"
read name
echo "Hello, $name! Welcome to Shell Scripting."
```

### Output:



A screenshot of a terminal window titled "lakshay42@lakshay42-VirtualBox:~". The window contains the following text:

```

lakshay42@lakshay42-VirtualBox:~$ vim act2.sh
lakshay42@lakshay42-VirtualBox:~$ bash act2.sh
What is your name?
Lakshay Dhanda
hello, name! welcome to shell script.
lakshay42@lakshay42-VirtualBox:~$ vim act2.sh
lakshay42@lakshay42-VirtualBox:~$ bash act2.sh
What is your name?
Lakshay Dhanda
hello, Lakshay Dhanda! welcome to shell script.
lakshay42@lakshay42-VirtualBox:~$ 

```

The terminal window has a dark background and light-colored text. The bottom right corner shows the system tray with icons for battery, signal, and volume, and the time "20:28".

## Exercise 3: [Arithmetic Operations in Shell Scripting]

### Task Statement:

- [Using Basic Arithmetic Operations in Shell Scripts]

### Command(s):

```

#!/bin/bash
echo "Enter first number: "
read num1
echo "Enter second number: "
read num2

echo "Addition: $((num1 + num2))"
echo "Subtraction: $((num1 - num2))"
echo "Multiplication: $((num1 * num2))"
echo "Division: $((num1 / num2))"

```

### Output:



## Exercise 4:

- [Voting Eligibility]

## Task Statement:

- [Using Conditionals in Shell script ]

## Command(s):

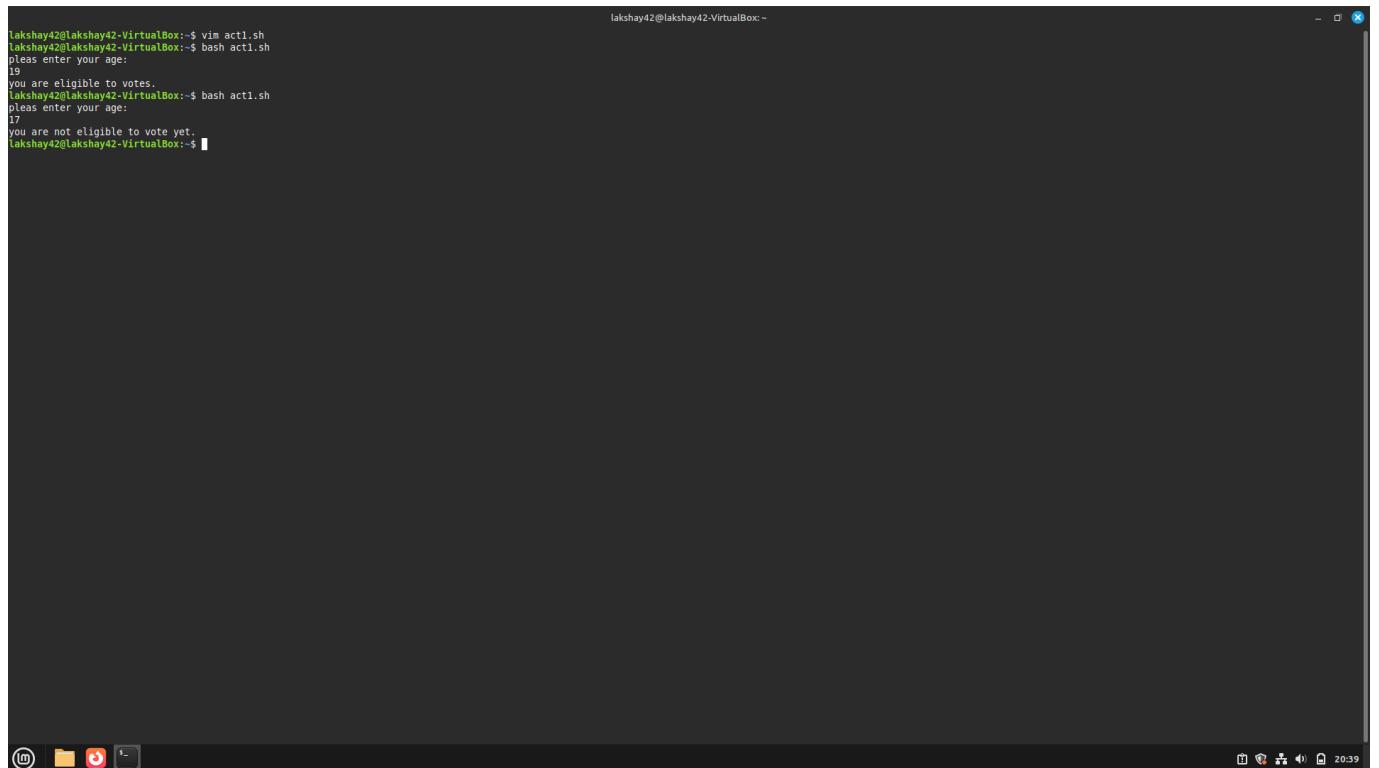
```
#!/bin/bash
echo "What is your age?"
read age
if [ $age -ge 18 ]; then

    echo "You are eligible to vote!"

else
    echo "You are not eligible to vote!"

fi
```

## Output:



The screenshot shows a terminal window with a dark background. At the top, it says 'lakshay42@lakshay42-VirtualBox:~'. The terminal displays the following session:

```
lakshay42@lakshay42-VirtualBox:~$ vim act1.sh
lakshay42@lakshay42-VirtualBox:~$ bash act1.sh
please enter your age:
19
you are eligible to votes.
lakshay42@lakshay42-VirtualBox:~$ bash act1.sh
please enter your age:
17
you are not eligible to vote yet.
lakshay42@lakshay42-VirtualBox:~$
```

At the bottom of the terminal, there is a toolbar with icons for file operations and a system tray showing the date and time '20:39'.

## Result

- The Exercises were successfully completed for Basic Shell Scripting

## Experiment [5]: [Shell Programming]

Name:lakshay dhanda Roll.290029328: Date:30-10-2025

AIM:

- [To Learn Basic Conditional Statements in Bash Scripting]

Requirements:

- [Any Linux Distro, any kind of text editor (vs code, vim, notepad, nano, etc)]

Theory:

- [Basic usage of conditions and arrays in bash scripting.]

## Procedure & Observations

### Exercise 1: [Prime Number Check]

Task Statement:

- [To check if the number given by the user is a prime number or not.]

Explanation:

- [using if else loop wap to check if the number is a prime number or not.]

Command(s):

```
#!/bin/bash
echo "Enter a number: "
read num
flag=0

for ((i=2; i<=num/2; i++))
do
    if [ $((num % i)) -eq 0 ]
    then
        flag=1
        break
    fi
done

if [ $flag -eq 0 ]
then
    echo "$num is a prime number."
else
    echo "$num is not a prime number."
fi
```

## Output:

```

lakshay42@lakshay42-VirtualBox:~$ vim exp5.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp5.sh
Enter a number:
42
42 is not a prime number.
lakshay42@lakshay42-VirtualBox:~$ bash exp5.sh
Enter a number:
7
7 is a prime number.
lakshay42@lakshay42-VirtualBox:~$
```

## Exercise 2: [Sum of Digits]

### Task Statement:

- [Take input from user and give the sum of two digits.]

### Explanation:

- [This script will take input from user and will give the following output.]

### Command(s):

```

#!/bin/bash
echo "Enter a number: "
read num
sum=0

while [ $num -gt 0 ]
do
    digit=$((num % 10))
    sum=$((sum + digit))
    num=$((num / 10))
done

echo "Sum of digits: $sum"
```

## Output:

```

lakshay42@lakshay42-VirtualBox:~$ vim exp5.1.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp5.1.sh
Enter a number:
42
Sum of digits: 6
lakshay42@lakshay42-VirtualBox:~$

```

The screenshot shows a terminal window with a dark background. At the top, there's a header bar with the terminal title and some icons. The main area contains a command-line session. The user runs 'vim exp5.1.sh' to edit a script, then 'bash exp5.1.sh' to execute it. The script prompts for a number ('Enter a number:'), receives '42', calculates the sum of digits ('Sum of digits: 6'), and then exits ('lakshay42@lakshay42-VirtualBox:~\$').

## Exercise 3: [Armstrong Numbers]

### Task Statement:

- [Take input user and give the sum of Armstrong number of n digits is a number equal to the sum of its digits raised to the power n. Example:  $153 = 1^3 + 5^3 + 3^3$  ]

### Explanation:

- [This script will tell if the number entered by the user is an armstrong number or not.]

### Command(s):

```

#!/bin/bash
echo "Enter a number: "
read num
temp=$num
n=${#num}      # number of digits
sum=0

while [ $temp -gt 0 ]
do
    digit=$((temp % 10))
    sum=$((sum + digit**n))
    temp=$((temp / 10))
done

if [ $sum -eq $num ]
then
    echo "$num is an Armstrong number."
else

```

```
echo "$num is not an Armstrong number."  
fi
```

## Output:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp5.3.sh  
lakshay42@lakshay42-VirtualBox:~$ bash exp 5.3.sh  
bash: exp: No such file or directory  
lakshay42@lakshay42-VirtualBox:~$ bash exp5.3.sh  
Enter a number:  
153  
153 is an Armstrong number.  
lakshay42@lakshay42-VirtualBox:~$
```

## Result:

- The Exercises were successfully completed for Basic Shell Scripting.

## Experiment 6: Shell Loops

Name:lakshay Dhanda Roll No.: 590029328 Date: 30-10-2025

Aim:

- To understand and implement shell loops (`for`, `while`, `until`) in Bash.
- To practice loop control constructs (`break`, `continue`) and loop-based file processing.

### Requirements

- A Linux system with bash shell.
- A text editor (nano, vim) and permission to create and execute shell scripts.

## Theory

Loops allow repeated execution of commands until a condition is met. Common loop constructs in Bash include `for` (iterate over items), `while` (repeat while condition true), and `until` (repeat until condition becomes true). Loop control statements like `break` and `continue` change the flow inside loops. Loops are essential for automating repetitive tasks such as processing multiple files, generating sequences, and collecting user input.

## Procedure & Observations

### Exercise 1: Simple `for` loop

Task Statement:

Write a `for` loop that prints numbers 1 to 5.

Command(s):

```
for i in 1 2 3 4 5; do
    echo "Number: $i"
done
```

Output:

A screenshot of a terminal window titled "lakshay42@lakshay42-VirtualBox:~". The window shows the command "vim exp6.sh" being run, followed by "Number: 1", "Number: 2", "Number: 3", "Number: 4", "Number: 5", and finally "lakshay42@lakshay42-VirtualBox:~\$". The terminal has a dark background and a light-colored text area. At the bottom, there is a toolbar with icons for file operations and a system tray with various icons.

---

## Exercise 2: **for** loop over files

Task Statement:

Process all **.txt** files in a directory and count lines in each.

Command(s):

```
for f in *.txt; do
    echo "File: $f - Lines: $(wc -l < "$f")"
done
```

Output:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.2.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.2.sh
File: backup_file.txt - Lines: 0
File: config.txt - Lines: 1
File: dated_file.txt - Lines: 0
File: file1.txt - Lines: 0
File: file2.txt - Lines: 0
File: file3.txt - Lines: 0
File: file4.txt - Lines: 0
File: logfile.txt - Lines: 0
File: newfile.txt - Lines: 0
File: number1.txt - Lines: 20
File: original.txt - Lines: 0
File: readme.txt - Lines: 0
File: standard_in - Lines: 4
File: system_info.txt - Lines: 14
File: systemTxt - Lines: 2
File: todo.txt - Lines: 1
wc: 'standard input': Is a directory
File: touch.txt - Lines: 0
File: xyz.txt - Lines: 0
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 3: C-style **for** loop

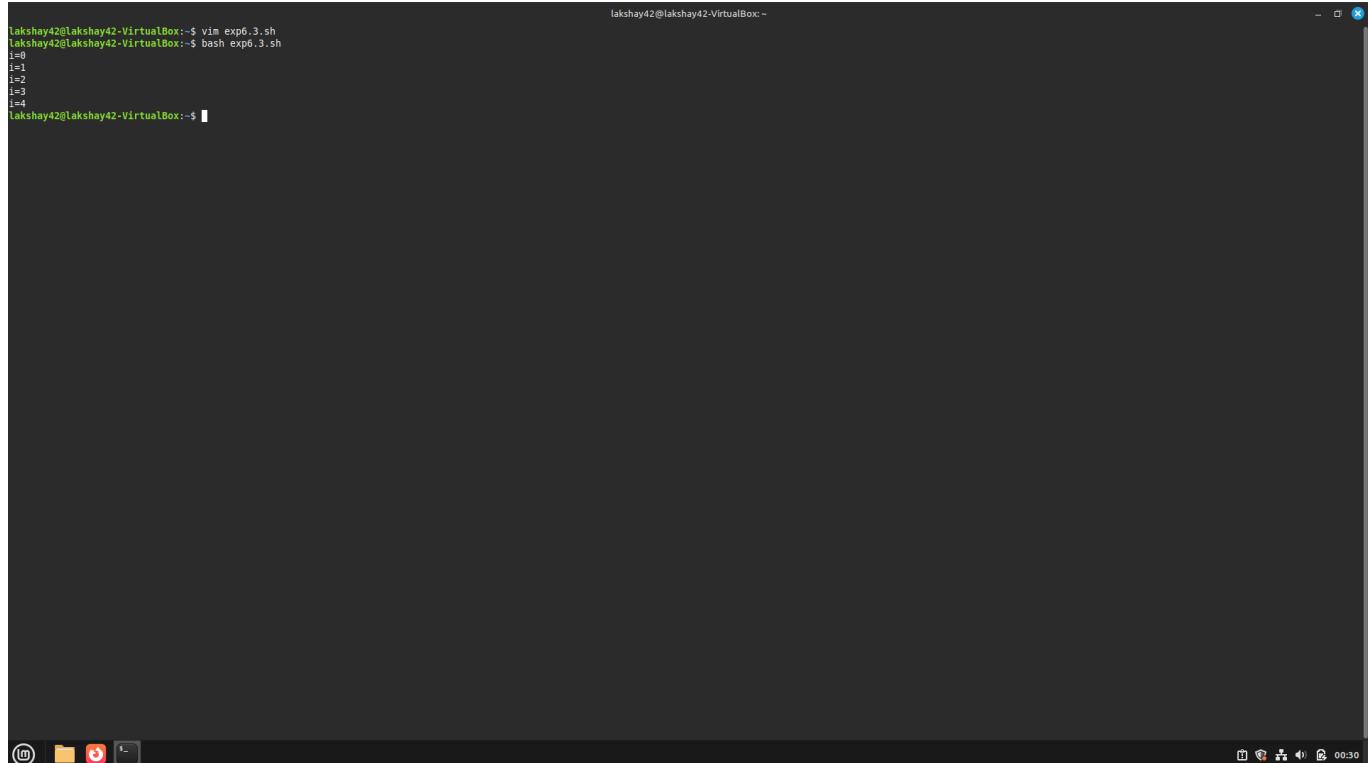
Task Statement:

Use arithmetic C-style loop for numeric iteration.

Command(s):

```
for ((i=0;i<5;i++)); do
    echo "i=$i"
done
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.3.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.3.sh
i=0
i=1
i=2
i=3
i=4
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 4: `while` loop and reading input

Task Statement:

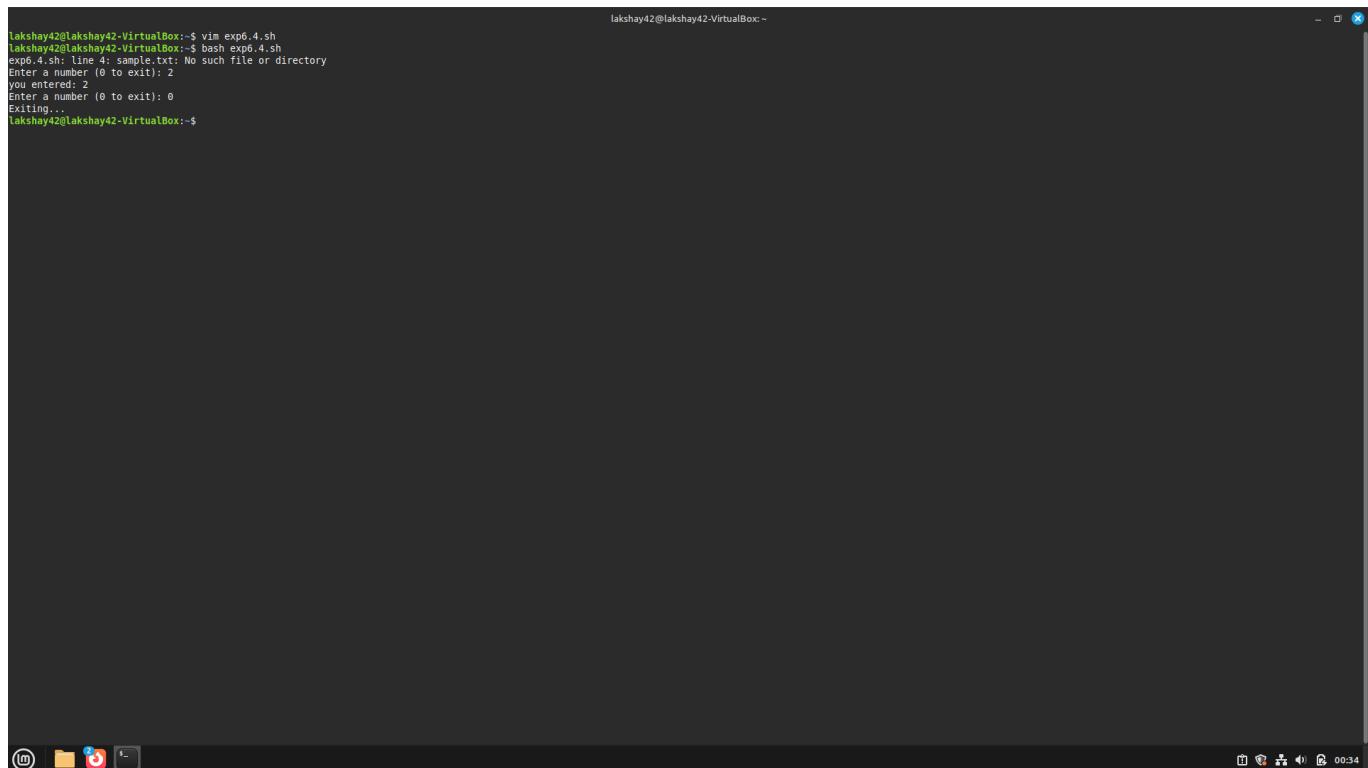
Write a `while` loop that reads lines from a file or from user input.

Command(s):

```
# Read from file
while read -r line; do
    echo "Line: $line"
done < sample.txt

# Read from user with exit condition
while true; do
    read -p "Enter a number (0 to exit): " n
    if [[ $n -eq 0 ]]; then
        echo "Exiting..."; break
    fi
    echo "You entered: $n"
done
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.4.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.4.sh
exp6.4.sh: line 4: sample.txt: No such file or directory
Enter a number (0 to exit): 2
you entered: 2
Enter a number (0 to exit): 0
Exiting...
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 5: until loop

Task Statement:

Use an `until` loop to run until a condition becomes true.

Command(s):

```
count=1
until [ $count -gt 5 ]; do
    echo "count=$count"
    ((count++))
done
```

Output:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.5.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.5.sh
count=1
count=2
count=3
count=4
count=5
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 6: **break** and **continue**

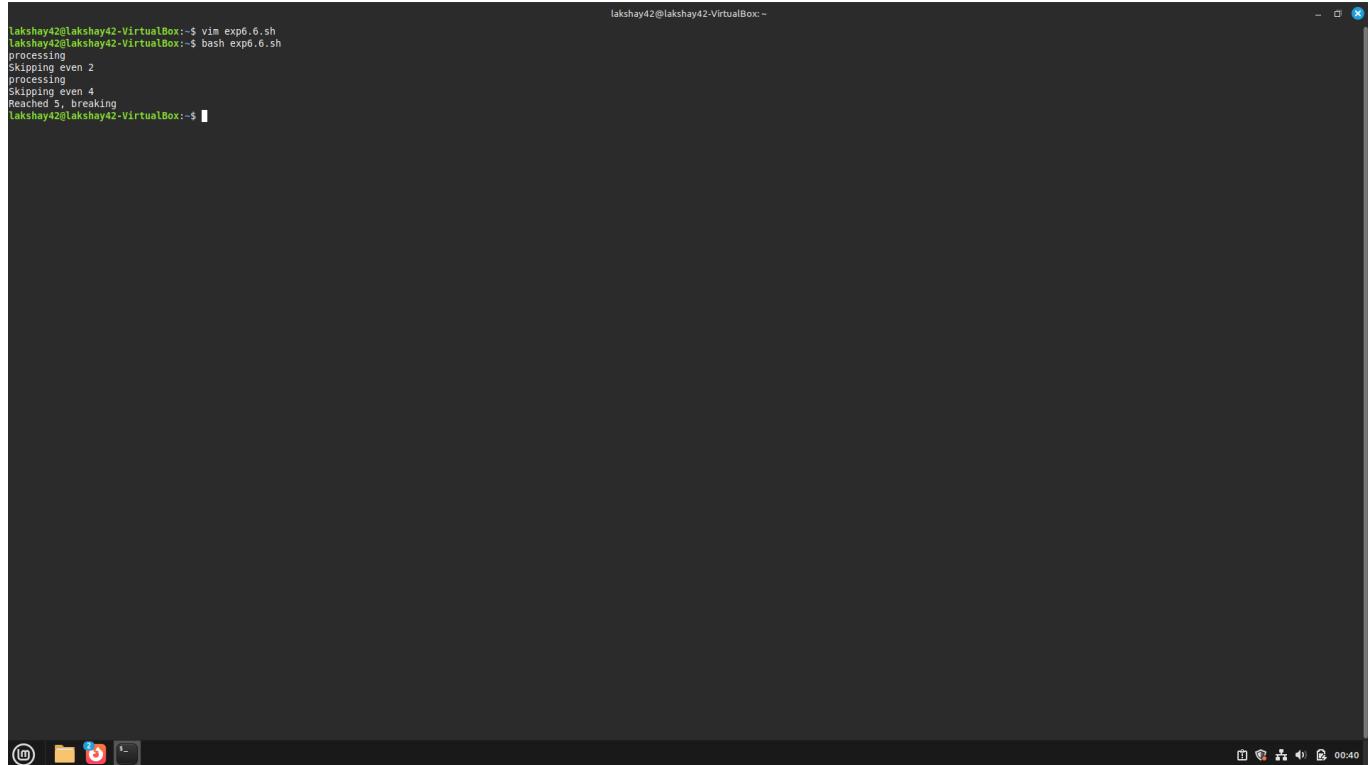
Task Statement:

Demonstrate **break** and **continue** inside a loop.

Command(s):

```
for i in {1..10}; do
    if [[ $i -eq 5 ]]; then
        echo "Reached 5, breaking"; break
    fi
    if (( i % 2 == 0 )); then
        echo "Skipping even $i"; continue
    fi
    echo "Processing $i"
done
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.6.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.6.sh
processing
Skipping even 2
processing
Skipping even 4
Reached 5, breaking
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 7: Nested loops

Task Statement:

Create nested loops to generate a multiplication table.

Command(s):

```
for i in {1..3}; do
    for j in {1..3}; do
        echo -n "$(i*j)) "
    done
    echo
done
```

Output:

The screenshot shows a terminal window titled 'lakshay42@lakshay42-VirtualBox:~'. The terminal output is as follows:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp6.7.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp6.7.sh
1 2 3
2 4 6
3 6 9
lakshay42@lakshay42-VirtualBox:~$
```

The terminal interface includes a title bar, a scrollable text area, and a dark-themed window frame. The bottom of the screen shows the desktop environment's taskbar with icons for file operations and system status.

---

## Result

- Implemented `for`, `while`, and `until` loops and used loop control statements.
- Practiced reading input, processing files, and nested iteration.

## Challenges Faced & Learning Outcomes

- Challenge 1: Handling spaces and special characters when iterating filenames — learned to use quotes and `read -r`.
- Challenge 2: Remembering arithmetic syntax in Bash — used `(( ))` and `expr` where needed.

## Learning:

- Loops are powerful for automation in shell scripting. Correct quoting and use of control constructs prevent common bugs.

## Conclusion

The lab demonstrated practical loop constructs in Bash for automating repetitive tasks and processing data efficiently.

# Experiment 7: Shell Programming, Process and Scheduling

Name: lakshay Dhanda Roll No.: 590029328 Date: 30-10-2025

Aim:

- To write shell scripts that demonstrate process management.
- To understand how to schedule processes using `cron` and `at`.
- To monitor running processes and practice job control commands.

## Requirements

- A Linux machine with bash shell.
- Access to process management commands (`ps`, `top`, `kill`, `jobs`, `fg`, `bg`).
- Access to scheduling utilities (`cron`, `at`).

## Theory

Every program running in Linux is a process identified by a unique process ID (PID). Shell programming allows automation of tasks including spawning and controlling processes. Process management commands like `ps`, `top`, `kill`, `jobs`, `bg`, and `fg` let users monitor and control execution. Scheduling utilities such as `cron` (repeated tasks) and `at` (one-time tasks) allow tasks to run automatically at defined times. Combining scripting with scheduling is a core system administration skill.

## Procedure & Observations

### Exercise 1: Writing a basic shell script

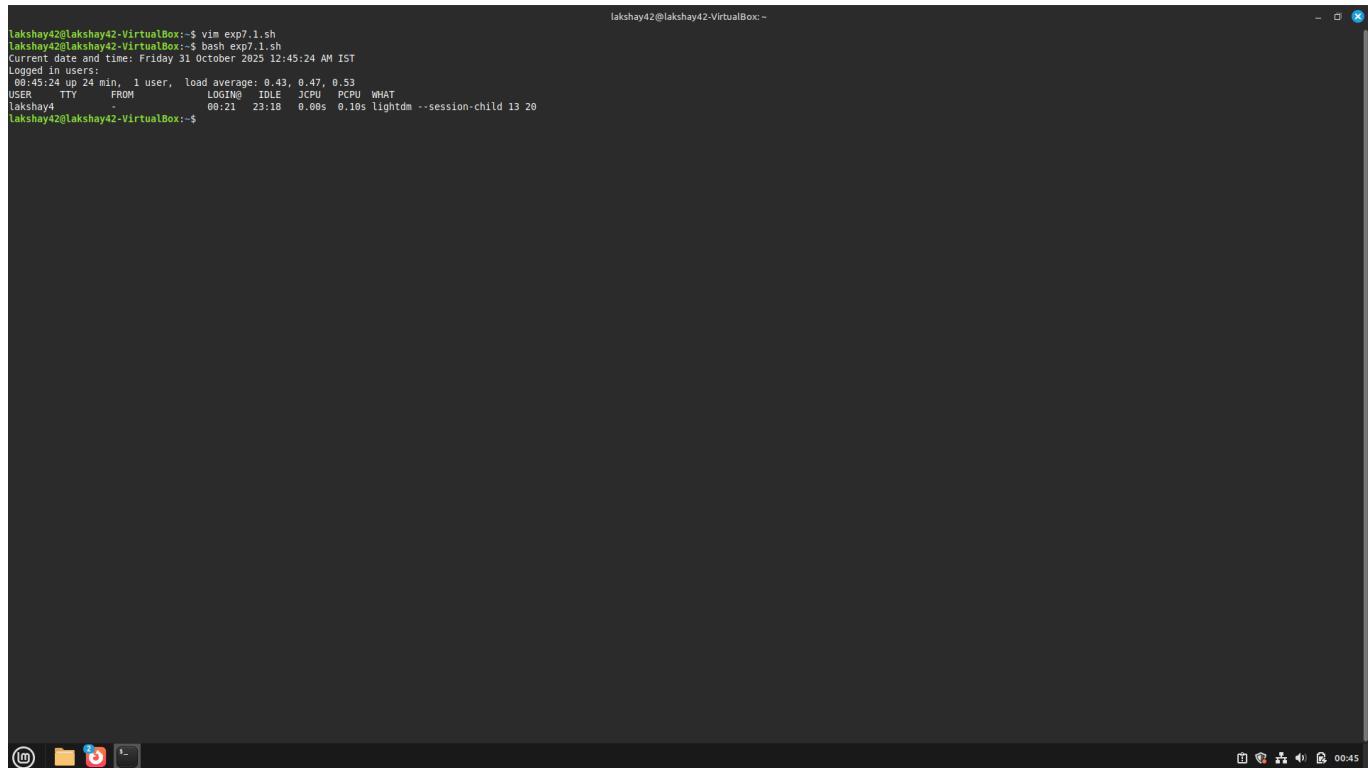
Task Statement:

Create a shell script that prints the current date, time, and the list of logged-in users.

Command(s):

```
#!/bin/bash
echo "Current date and time: $(date)"
echo "Logged in users:"
w
```

Output:



A screenshot of a terminal window titled "lakshay42@lakshay42-VirtualBox:~". The window displays system status information:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp7.1.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp7.1.sh
Current date and time: Friday 31 October 2025 12:45:24 AM IST
Logged in users:
 00:45:24 up 24 min, 1 user, load average: 0.43, 0.47, 0.53
USER   TTY      FROM             LOGIN@  IDLE   JCPU   PCPU WHAT
lakshay4        00:21   23:18   0.00s  0.10s lightdm --session-child 13 20
lakshay42@lakshay42-VirtualBox:~$
```

The terminal window has a dark background and a light-colored text area. The bottom of the window shows the desktop environment's taskbar with various icons.

---

## Exercise 2: Background and foreground processes

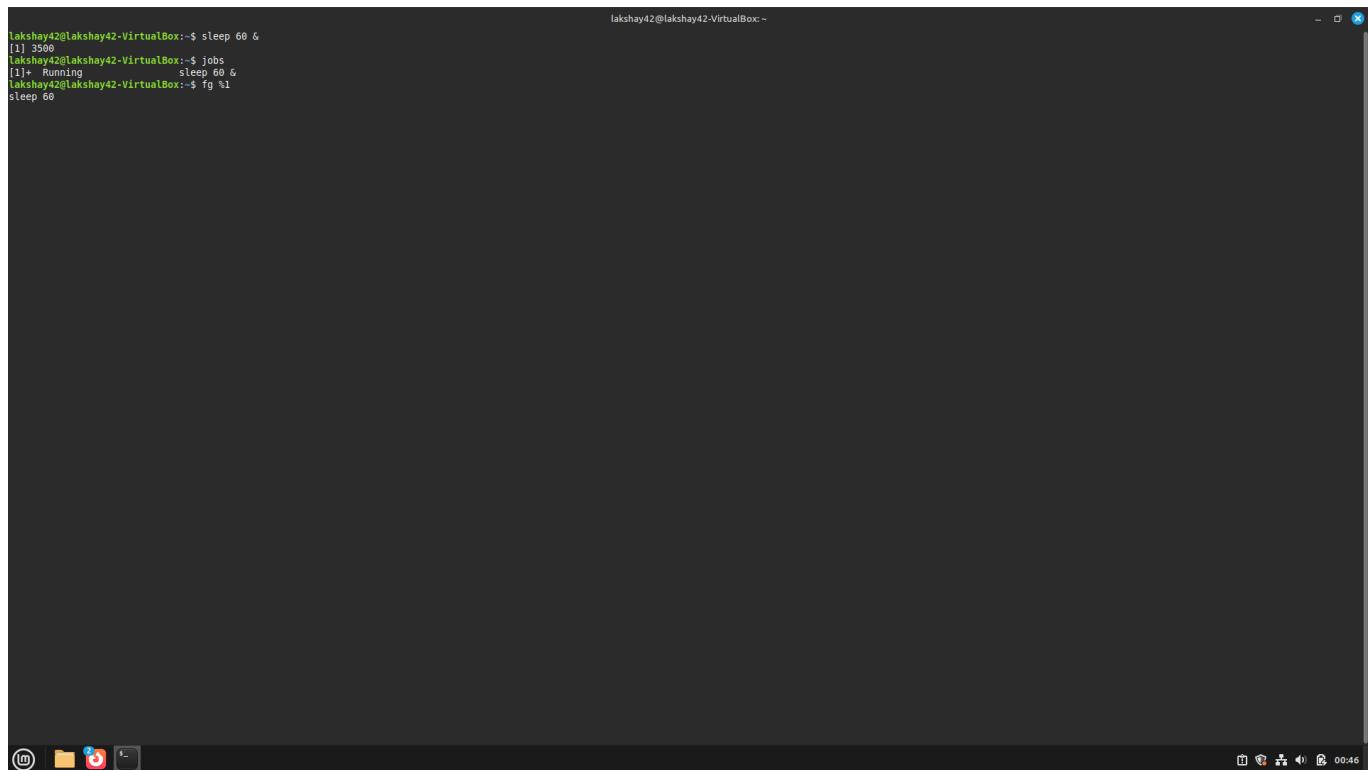
Task Statement:

Run a process in background and bring it to the foreground.

Command(s):

```
sleep 60 &
jobs
fg %1
```

Output:



A screenshot of a terminal window titled "lakshay42@lakshay42-VirtualBox:~". The terminal shows the following command sequence:

```
lakshay42@lakshay42-VirtualBox:~$ sleep 60 &
[1] 3500
lakshay42@lakshay42-VirtualBox:~$ jobs
[1]+  Running                  sleep 60 &
lakshay42@lakshay42-VirtualBox:~$ fg %1
sleep 60
```

The terminal window has a dark background and light-colored text. The bottom of the window shows the system tray with icons for network, file manager, and other system status indicators.

---

## Exercise 3: Killing a process

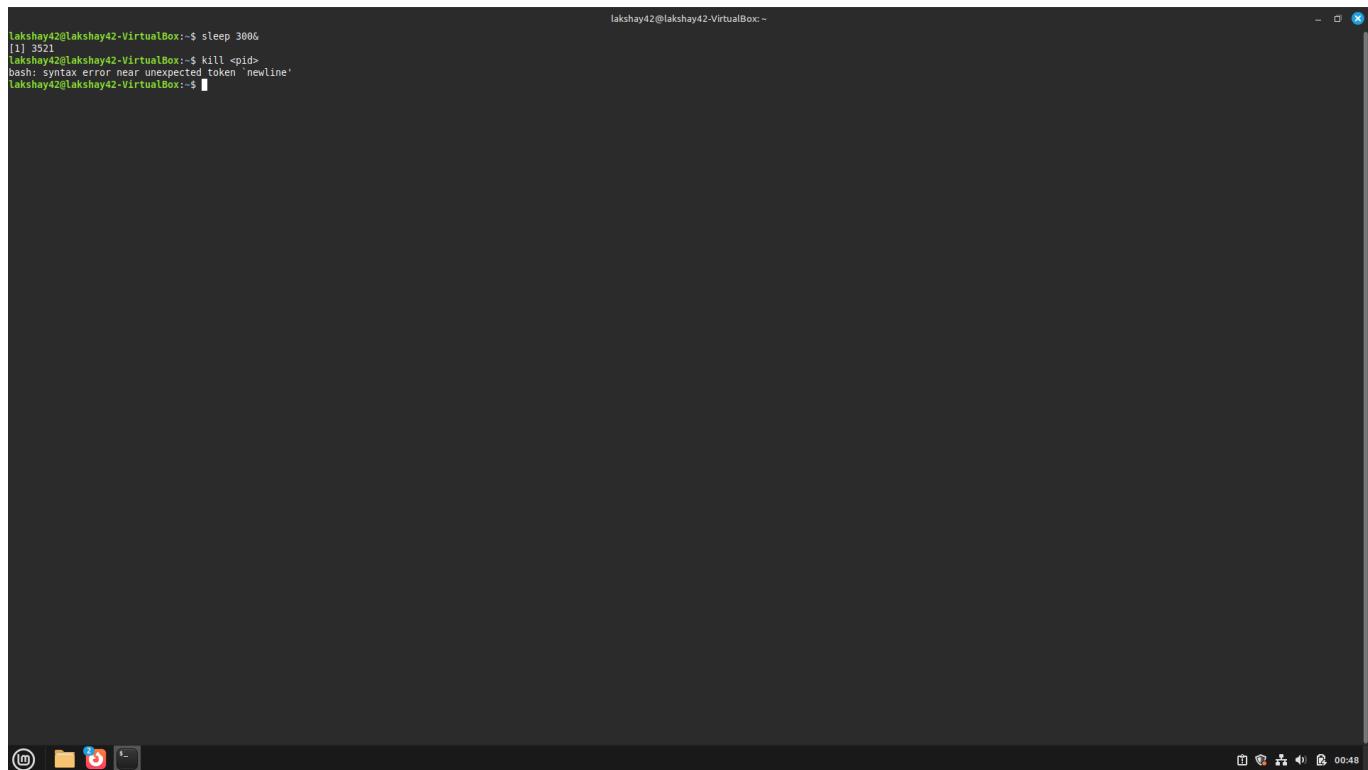
Task Statement:

Start a process and terminate it using **kill**.

Command(s):

```
sleep 300 &
ps aux | grep sleep
kill <pid>
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ sleep 300&
[1] 3521
lakshay42@lakshay42-VirtualBox:~$ kill <pid>
bash: syntax error near unexpected token 'newline'
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 4: Monitoring processes

Task Statement:

Use `ps` and `top` to monitor processes.

Command(s):

```
ps aux | head -5
top
```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ ps aux | head -5
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1  0.1  0.6  22736 13480 ?        Ss  00:20  0:02 /sbin/init splash
root      2  0.0  0.0   0   0 ?        S   00:20  0:00 [kthreadd]
root      3  0.0  0.0   0   0 ?        S   00:20  0:00 [pool_workqueue release]
root      4  0.0  0.0   0   0 ?        I<  00:20  0:00 [kworker/R-rcu_g]
lakshay42@lakshay42-VirtualBox:~$ top

```

```

top - 00:50:06 up 29 min, 1 user, load average: 0.02, 0.19, 0.39
Tasks: 193 total, 2 running, 191 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.0 us, 1.9 sy, 0.0 ni, 95.2 id, 0.0 wa, 0.0 hi, 2.3 si, 0.0 st
Mem: 1967.9 total, 223.2 free, 1381.1 used, 651.6 buff/cache
Swap: 2600.0 total, 2386.3 free, 341.7 used. 649.7 avail Mem

PID USER      PR NI VIRT  RES  SHR S %CPU %MEM    TIME+ COMMAND
1477 lakshay+ 20  0 3833956 217924 146132 S  4.7 10.8  3:17.30 cinnamon
847 root      20  0 498652 167160 118900 S  1.6  8.3  1:01.57 Xorg
2174 lakshay+ 20  0 496348 59560  10800 R  1.0  2.1  0:00.00 mlreport-tray
3034 lakshay+ 20  0 2415812 103320 71935 S  1.2  2.5  0:01.04 libextensions
2884 lakshay+ 20  0 11.10 452064 180384 S  0.9 23.5  0:36.09 firefox-bin
3149 lakshay+ 20  0 2415812 49280 36096 S  0.9  2.4  0:08.55 Web Content
3153 lakshay+ 20  0 2415812 49244 36066 S  0.9  2.4  0:08.64 Web Content
3120 lakshay+ 20  0 2917948 361132 109672 S  0.6 17.9  0:45.55 Isolated Web Co
3151 lakshay+ 20  0 2415808 49408 36224 S  0.6  2.5  0:00.61 Web Content
3510 lakshay+ 20  0 14568 5766 3584 R  0.0  0.0  0:00.00 top
164 root      20  0 10000  0  0  0  0.3  0.0  0:02.68 ksoftirqd/0
65 root      20  0 0  0  0  0  0.1  0.0  0:01.32 kworker/u12:events
3449 root      20  0 0  0  0  0  0.3  0.0  0:00.12 kworker/u5:0-events_freezeable_power_
1 root      20  0 22736 13480 9384 S  0.0  0.7  0:02.14 systemd
2 root      20  0 0  0  0  0  0.0  0.0  0:00.00 pool_workqueue_release
3 root      20  0 0  0  0  0  0.0  0.0  0:00.00 kworker/R-rcu_g
4 root      0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-slub
5 root      0  20 0  0  0  0  0.0  0.0  0:00.00 kworker/R-netns
6 root      0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-slub
7 root      0  20 0  0  0  0  0.0  0.0  0:00.00 kworker/R-netsn
8 root      20  0 0  0  0  0  0.1  0.0  0:00.00 kworker/R-0-events
12 root     0  20 0  0  0  0  0.0  0.0  0:00.00 kworker/R-mm_pe
13 root     20  0 0  0  0  0  0.1  0.0  0:00.00 rcu_tasks_kthread
14 root     20  0 0  0  0  0  0.1  0.0  0:00.00 rcu_tasks_rude_kthread
15 root     20  0 0  0  0  0  0.0  0.0  0:00.00 rcu_tasks_trace_kthread
17 root     20  0 0  0  0  0  0.1  0.0  0:01.56 rcu preempt
18 root     rt  0 0  0  0  0  0.5  0.0  0:00.04 migration/0
19 root     -51 0 0  0  0  0  0.5  0.0  0:00.00 idle_inject/0
20 root     20  0 0  0  0  0  0.5  0.0  0:00.00 cpuhp/0
21 root     20  0 0  0  0  0  0.5  0.0  0:00.00 cpuhp/1
22 root     -51 0 0  0  0  0  0.5  0.0  0:00.00 idle_inject/1
23 root     10  0 0  0  0  0  0.0  0.0  0:00.00 migration/1
24 root     20  0 0  0  0  0  0.1  0.0  0:01.85 ksoftirqd/1
29 root     20  0 0  0  0  0  0.5  0.0  0:00.00 kdevtmpfs
30 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-inet
31 root     20  0 0  0  0  0  0.1  0.0  0:00.42 kworker/u5:1-events_power_efficient
32 root     20  0 0  0  0  0  0.5  0.0  0:00.00 kauditd
33 root     20  0 0  0  0  0  0.5  0.0  0:00.00 khnqtasksd
34 root     20  0 0  0  0  0  0.0  0.0  0:00.00 kworker/R-0-worker
36 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-write
37 root     20  0 0  0  0  0  0.5  0.0  0:00.50 kcompactd0
38 root     25  5 0  0  0  0  0.5  0.0  0:00.00 ksmd
40 root     39 19 0  0  0  0  0.5  0.0  0:00.00 khugepaged
41 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-kinte
42 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-kblob
43 root     0  20 0  0  0  0  0.0  0.0  0:00.00 kworker/R-klog
44 root     -51 0 0  0  0  0  0.5  0.0  0:00.00 irq/9-spi
45 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-tpm_d
46 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-ata_s
47 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-md_md
48 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-mdac-
49 root     0  20 0  0  0  0  0.1  0.0  0:00.00 kworker/R-edac-

```

--

## Exercise 5: Using cron for scheduling

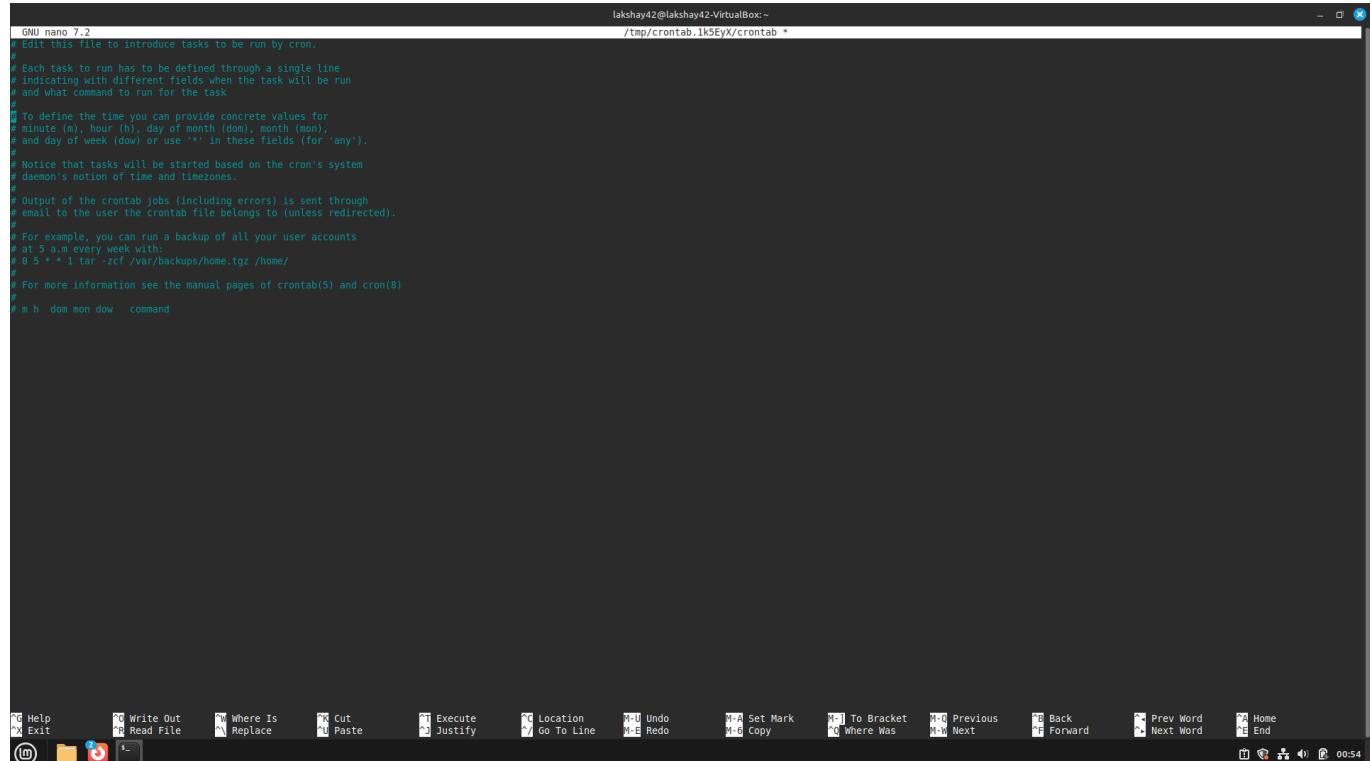
### Task Statement:

Schedule a script to run every day at 7:00 AM using cron.

### Command(s):

```
crontab -e
# Add the following line
0 7 * * * /home/user/myscript.sh
```

## Output:



```
GNU nano 7.2
lakshay42@lakshay42-VirtualBox: ~
/tmp/crontab.1k5EyX/crontab *

# Edit this file to introduce tasks to be run by cron.

# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task

# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').

# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezone.

# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).

# For example, you can run a backup of all your user accounts
# at 5 a.M every week with:
# 0 5 * * 1 tar -czf /var/backups/home.tgz /home/
# For more information see the manual pages of crontab(5) and cron(8)

# m h dom mon dow   command
```

The terminal window shows the crontab file being edited. The file contains a single line: '0 7 \* \* \* /home/user/myscript.sh'. The terminal title is 'lakshay42@lakshay42-VirtualBox: ~' and the path is '/tmp/crontab.1k5EyX/crontab \*'. The bottom of the window shows the nano editor's status bar with various keyboard shortcuts.

## Exercise 6: Using **at** for one-time scheduling

### Task Statement:

Schedule a script to run once at a specified time using **at**.

### Command(s):

```
echo "/home/user/myscript.sh" | at 08:30
atq
```

## Output:

The screenshot shows a terminal window titled 'lakshay42@lakshay42-VirtualBox: ~'. The session starts with the user opening a script named 'exp7.2.sh' using vim. When they try to run it with 'bash exp7.2.sh', they encounter errors for the 'at' command. To resolve this, they use 'cat' to copy the script's contents into a temporary file named 'myscript.sh' and then run it with 'atq'. The terminal window has a dark background and includes standard Linux desktop icons in its dock.

```
lakshay42@lakshay42-VirtualBox:~$ vim exp7.2.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp7.2.sh
exp7.2.sh: line 1: at: command not found
exp7.2.sh: line 2: atq: command not found
lakshay42@lakshay42-VirtualBox:~$ cat exp7.2.sh
echo "/home/user/myscript.sh" | at 08:30
atq
lakshay42@lakshay42-VirtualBox:~$
```

## Result

- Learned to create and run shell scripts.
- Managed processes using background, foreground, and kill commands.
- Monitored processes with `ps` and `top`.
- Scheduled recurring tasks with `cron` and one-time tasks with `at`.

## Challenges Faced & Learning Outcomes

- Challenge 1: Remembering the `crontab` time format. Solved by using online crontab generators and practice.
- Challenge 2: Ensuring `atd` service is running for `at` command. Fixed by starting the service with `systemctl start atd`.

## Learning:

- Gained hands-on knowledge of process creation and termination.
- Learned job control and scheduling using `cron` and `at`.

## Conclusion

This experiment provided practical experience with shell scripting, process management, and scheduling. These are critical skills for system administrators to automate and control Linux environments effectively.

## Experiment 8: Shell Programming (Continued)

Name: lakshay Dhanda Roll No.: 590029328 Date: 30-10-2025

Aim:

- To extend shell programming concepts by using conditional statements, advanced scripting constructs, and command-line arguments.
- To practice writing scripts that perform decision-making and parameter handling.

### Requirements

- A Linux system with bash shell.
- Text editor and permission to create/execute shell scripts.

## Theory

Conditional execution in shell scripts allows branching logic using `if`, `elif`, `else`, and `case` statements.

Scripts can accept command-line arguments using `$1`, `$2`, ... and `$@` for all arguments. Control flow constructs combined with user input and arguments allow dynamic and reusable scripts.

## Procedure & Observations

### Exercise 1: Using `if-else`

Task Statement:

Write a script to check whether a given number is positive, negative, or zero.

Explanation:

We used an `if-elif-else` construct to compare the number against 0.

Command(s):

```
#!/bin/bash
num=$1
if [ $num -gt 0 ]; then
    echo "$num is positive"
elif [ $num -lt 0 ]; then
    echo "$num is negative"
else
    echo "$num is zero"
fi
```

Output:

```

lakshay42@lakshay42-VirtualBox:~$ vim exp8.1.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp8.1.sh
exp8.1.sh: line 3: [: -gt: unary operator expected
exp8.1.sh: line 5: [: -lt: unary operator expected
is zero
lakshay42@lakshay42-VirtualBox:~$ cat exp8.2.sh
#!/bin/bash
case $ch in
(aeiouAEIOU)) echo "$ch is a vowel" ;;
(bcdflghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ)) echo "$ch is a consonant" ;;
([0-9])) echo "$ch is a digit" ;;
(*) echo "$ch is a special character" ;;
esac
lakshay42@lakshay42-VirtualBox:~$ 

```

## Exercise 2: Using `case`

Task Statement:

Write a script that takes a character as input and classifies it as vowel, consonant, digit, or special character.

Explanation:

The `case` statement provides pattern matching for multiple options.

Command(s):

```

#!/bin/bash
ch=$1
case $ch in
[aeiouAEIOU]) echo "$ch is a vowel" ;;
[bcdflghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]) echo "$ch is a consonant" ;;
([0-9])) echo "$ch is a digit" ;;
(*)) echo "$ch is a special character" ;;
esac

```

Output:

The screenshot shows a terminal window titled 'lakshay42@lakshay42-VirtualBox: ~'. The user has run the command 'vim exp8.2.sh' followed by 'bash exp8.2.sh'. The script contains a single line: 'is a special character'. The terminal window includes a title bar, a scroll bar, and a status bar at the bottom showing icons and the time '01:08'.

```
lakshay42@lakshay42-VirtualBox:~$ vim exp8.2.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp8.2.sh
is a special character
lakshay42@lakshay42-VirtualBox:~$
```

---

## Exercise 3: Command-line arguments

Task Statement:

Write a script that accepts filename(s) as arguments and prints the number of lines in each file.

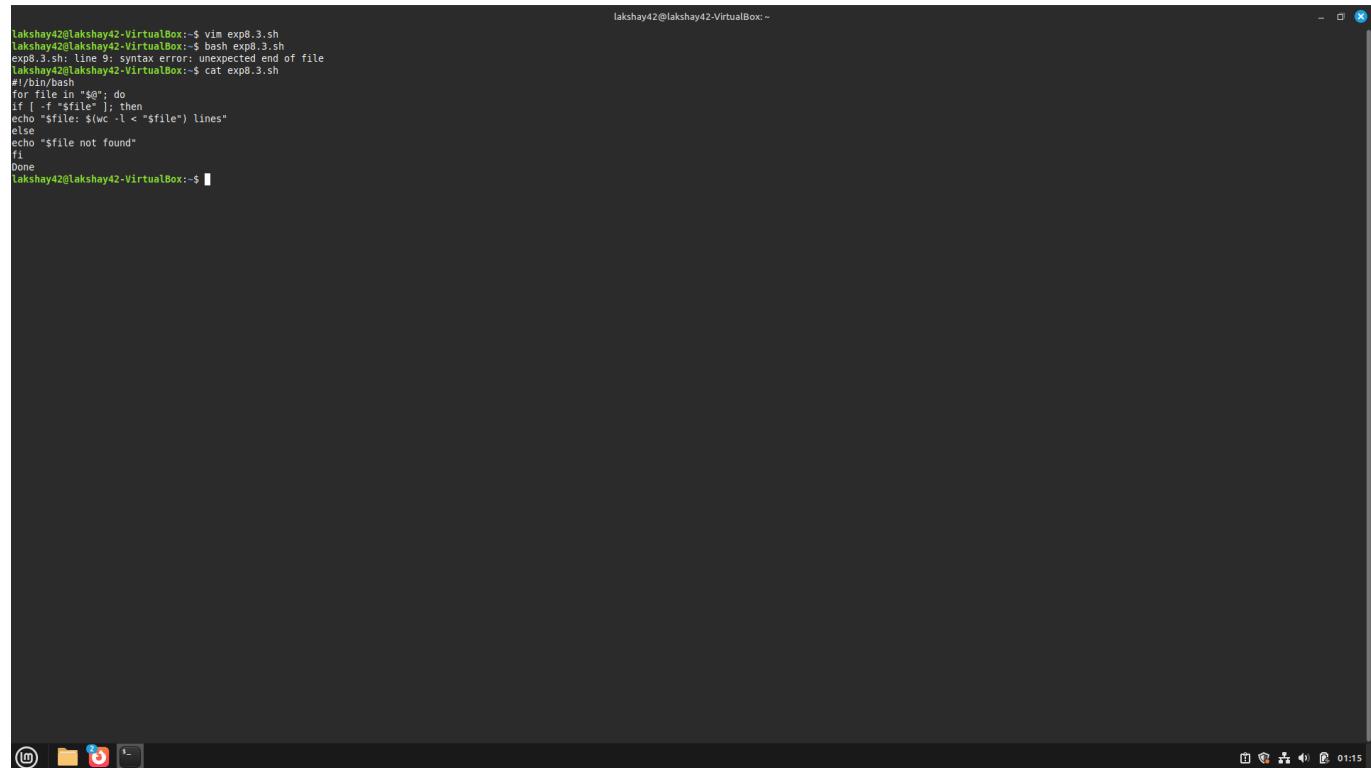
Explanation:

Command-line arguments are accessed using `$@`. Looping through each argument allows file-wise operations.

Command(s):

```
#!/bin/bash
for file in "$@"; do
    if [ -f "$file" ]; then
        echo "$file: $(wc -l < "$file") lines"
    else
        echo "$file not found"
    fi
done
```

Output:



A screenshot of a terminal window titled "lakshay42@lakshay42-VirtualBox:~". The terminal shows the following command sequence:

```
lakshay42@lakshay42-VirtualBox:~$ vim exp8.3.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp8.3.sh
exp8.3.sh: line 9: syntax error: unexpected end of file
lakshay42@lakshay42-VirtualBox:~$ cat exp8.3.sh
#!/bin/bash
for file in "$@"; do
if [ -f "$file" ]; then
lines=$(wc -l < "$file")
else
echo "$file not found"
fi
done
lakshay42@lakshay42-VirtualBox:~$
```

The terminal window has a dark background and light-colored text. The bottom right corner shows the system tray with icons for battery, signal, and time (01:15).

---

## Exercise 4: Nested conditionals

Task Statement:

Write a script to check if a year is a leap year.

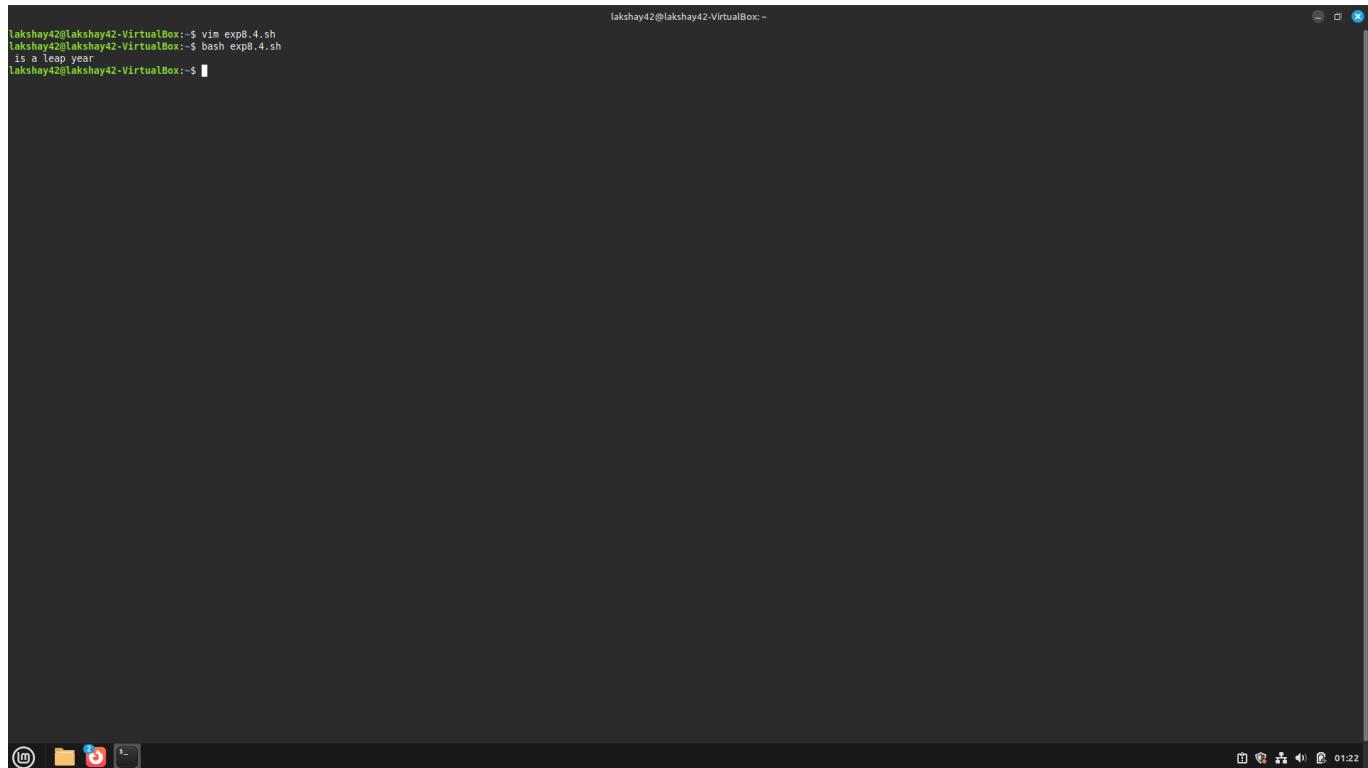
Explanation:

A leap year is divisible by 4, but if divisible by 100 it must also be divisible by 400.

Command(s):

```
#!/bin/bash
year=$1
if (( year % 400 == 0 )); then
    echo "$year is a leap year"
elif (( year % 100 == 0 )); then
    echo "$year is not a leap year"
elif (( year % 4 == 0 )); then
    echo "$year is a leap year"
else
    echo "$year is not a leap year"
fi
```

Output:



```
lakshay42@lakshay42-VirtualBox:~$ vim exp8.4.sh
lakshay42@lakshay42-VirtualBox:~$ bash exp8.4.sh
is a leap year
lakshay42@lakshay42-VirtualBox:~$
```

---

## Result

- Implemented conditional statements (`if-else`, `case`) in shell scripts.
- Practiced handling command-line arguments and nested conditions.
- Wrote reusable and flexible shell scripts.

## Challenges Faced & Learning Outcomes

- Challenge 1: Forgetting to quote variables in conditions — resolved by using "`$var`" to avoid word splitting.
- Challenge 2: Pattern matching in `case` — practiced with multiple examples.

### Learning:

- Learned practical use of branching and decision-making in shell scripting.
- Understood command-line argument handling for automation.

## Conclusion

This experiment extended shell programming by introducing decision-making and parameter handling. The scripts demonstrate the flexibility of shell programming for different use cases.