LAB-1.md

LAB1 – Linux Basics

This document summarizes commands executed from Lab3 and Lab5 in Unit-1, along with sample outputs and brief explanations.

Lab5 – File Permissions and Ownership

Command : `ls -la`

```
ls -la
Output: -rwxr-xr--
```

Explanation:

- ls → Lists directory contents
- -l → Long format: shows permissions, ownership, size, and modification date
- -a → Includes hidden files (those starting with .

Code Snippet:
alt text

Command : `chmod`

-used to modify access permissions for files and directories

Basic Syntax:

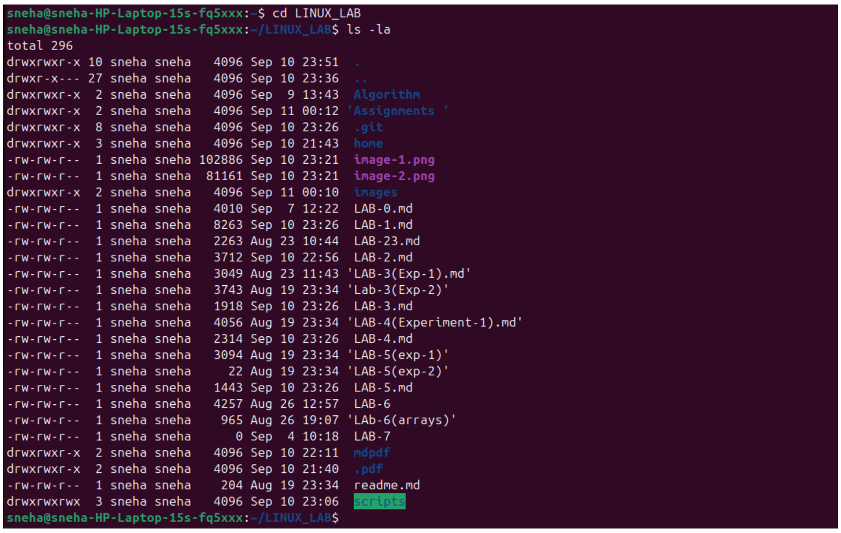
```
chmod 741 file.txt
chmod u+x file.txt
```

- `r` → Read (numeric value: 4)
- `w` → Write (numeric value: 2)
- `x` → Execute (numeric value: 1)

Explanation:

- 7 → User: read, write, execute
- 4 → Group: read only
- 1 → Others: execute only

Numeric method:



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~$ cd LINUX_LAB
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ ls -la
total 296
drwxrwxr-x 10 sneha sneha   4096 Sep 10 23:51  .
drwxr-x--- 27 sneha sneha   4096 Sep 10 23:36  ..
drwxrwxr-x  2 sneha sneha   4096 Sep  9 13:43  Algorithm
drwxrwxr-x  2 sneha sneha   4096 Sep 11 00:12 'Assignments '
drwxrwxr-x  8 sneha sneha   4096 Sep 10 23:26  .git
drwxrwxr-x  3 sneha sneha   4096 Sep 10 21:43  home
-rw-rw-r--  1 sneha sneha 102886 Sep 10 23:21  image-1.png
-rw-rw-r--  1 sneha sneha  81161 Sep 10 23:21  image-2.png
drwxrwxr-x  2 sneha sneha   4096 Sep 11 00:10  images
-rw-rw-r--  1 sneha sneha   4010 Sep  7 12:22  LAB-0.md
-rw-rw-r--  1 sneha sneha   8263 Sep 10 23:26  LAB-1.md
-rw-rw-r--  1 sneha sneha   2263 Aug 23 10:44  LAB-23.md
-rw-rw-r--  1 sneha sneha   3712 Sep 10 22:56  LAB-2.md
-rw-rw-r--  1 sneha sneha   3049 Aug 23 11:43 'LAB-3(Exp-1).md'
-rw-rw-r--  1 sneha sneha   3743 Aug 19 23:34 'Lab-3(Exp-2)'
-rw-rw-r--  1 sneha sneha   1918 Sep 10 23:26  LAB-3.md
-rw-rw-r--  1 sneha sneha   4056 Aug 19 23:34 'LAB-4(Experiment-1).md'
-rw-rw-r--  1 sneha sneha   2314 Sep 10 23:26  LAB-4.md
-rw-rw-r--  1 sneha sneha   3094 Aug 19 23:34 'LAB-5(exp-1)'
-rw-rw-r--  1 sneha sneha     22 Aug 19 23:34 'LAB-5(exp-2)'
-rw-rw-r--  1 sneha sneha   1443 Sep 10 23:26  LAB-5.md
-rw-rw-r--  1 sneha sneha   4257 Aug 26 12:57  LAB-6
-rw-rw-r--  1 sneha sneha    965 Aug 26 19:07 'LAb-6(arrays)'
-rw-rw-r--  1 sneha sneha      0 Sep  4 10:18  LAB-7
drwxrwxr-x  2 sneha sneha   4096 Sep 10 22:11  mdpdf
drwxrwxr-x  2 sneha sneha   4096 Sep 10 21:40  .pdf
-rw-rw-r--  1 sneha sneha    204 Aug 19 23:34  readme.md
drwxrwxrwx  3 sneha sneha   4096 Sep 10 23:06  scripts
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$
```

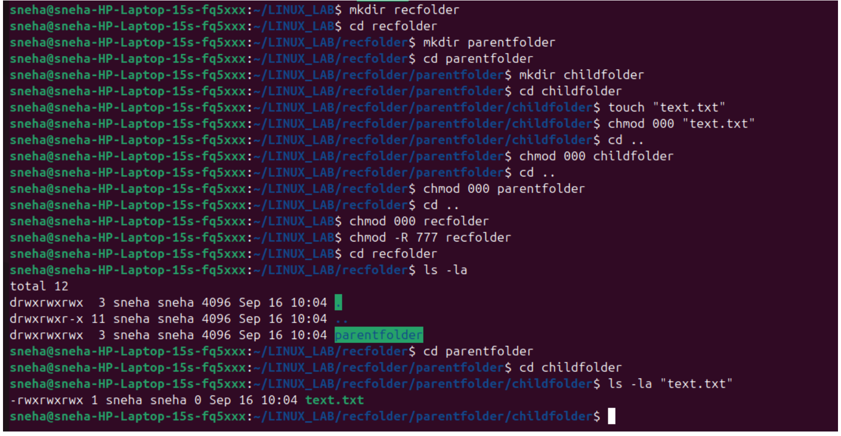
Recursive Permission Changes

-Apply changes to all files and subdirectories.

```
chmod -R 755 /mydir
```

- `-R` → Recursive flag

Code Snippet:



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ mkdir recfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ cd recfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ mkdir parentfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ cd parentfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder$ mkdir childfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder$ cd childfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ touch "text.txt"
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chmod 000 "text.txt"
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ cd ..
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder$ chmod 000 childfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder$ cd ..
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ chmod 000 parentfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ cd ..
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ chmod 000 recfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ chmod -R 777 recfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB$ cd recfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ ls -la
total 12
drwxrwxrwx  3 sneha sneha 4096 Sep 16 10:04 .
drwxrwxr-x 11 sneha sneha 4096 Sep 16 10:04 ..
drwxrwxrwx  3 sneha sneha 4096 Sep 16 10:04 parentfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder$ cd parentfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder$ cd childfolder
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ ls -la "text.txt"
-rwxrwxrwx 1 sneha sneha 0 Sep 16 10:04 text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$
```

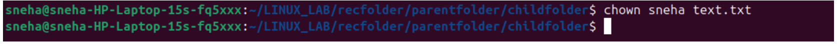
Command : chown

-Changes File Ownership.

Example:

```
chown newon:group2 data.txt
```

Explanation:
Assigns ownership of `data.txt` to user newon and group group2. **Code Snippet:**

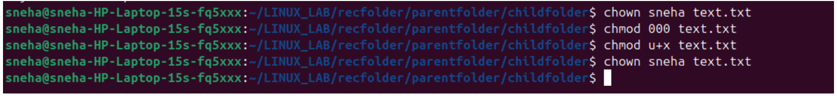


```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chown sneha text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ 
```

Applying All Concepts Together

```
chmod 700 project.sh         # Full access for user only
chmod u+x,g-w project.sh     # Add execute for user, remove write for group
chown root:admin project.sh  # Change owner to root and group to admin
```

Code Snippet:



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chown sneha text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chmod 000 text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chmod u+x text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ chown sneha text.txt
sneha@sneha-HP-Laptop-15s-fq5xxx:~/LINUX_LAB/recfolder/parentfolder/childfolder$ 
```

Lab6-Shell Scripting Basics

Shell Scripting

Shell scripting is the process of writing a series of commands for the shell (a command-line interpreter) to execute. It’s commonly used in Unix/Linux environments for automating tasks.

🔧What is a Shell?

A shell is a program that takes commands from the user and gives them to the operating system to execute basically The most common shells include:

- **bash** (Bourne Again SHell) – most common
- **sh** (Bourne shell)
- **zsh** (Z shell)
- **csh** (C shell)

📜 What is a Shell Script?

A shell script is a text file containing a sequence of shell commands. It usually has a .sh extension.

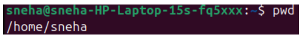
Example: **`first_script.sh`**

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

🔧 Steps to Create and Run a Shell Script in Linux

1- Check the current working directory:

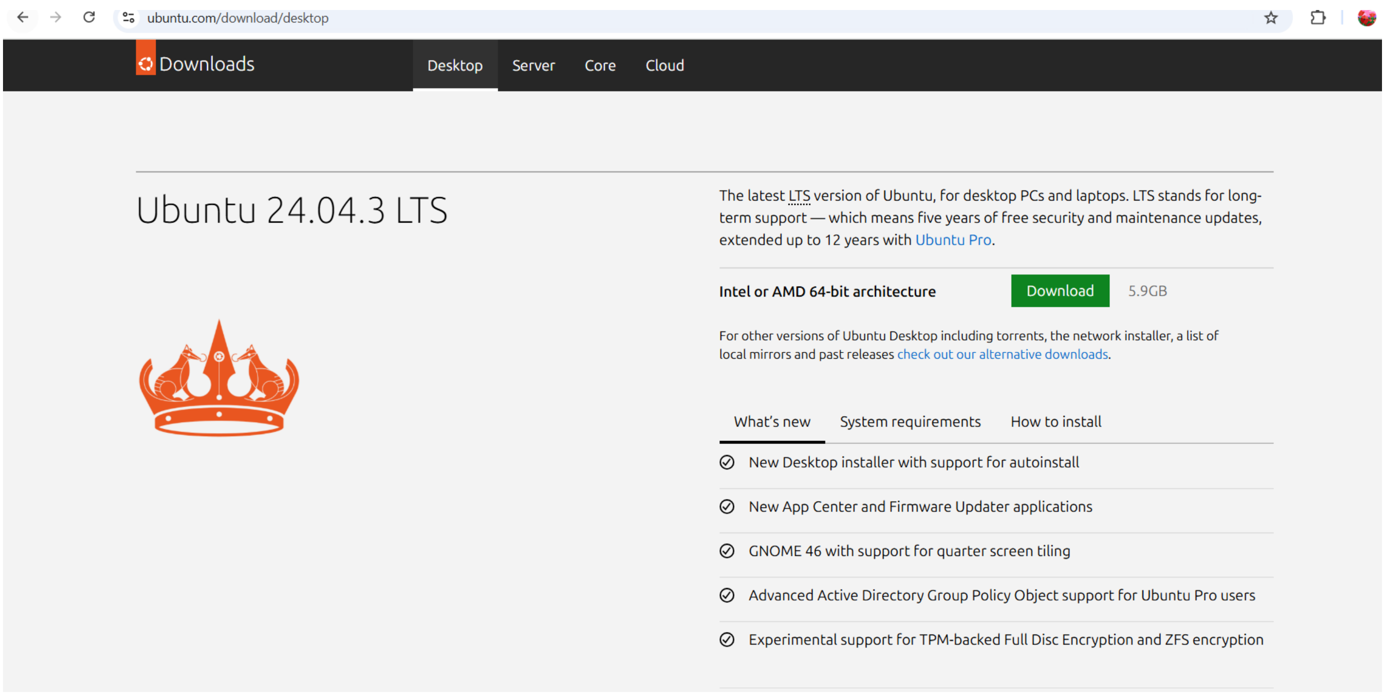
```
pwd
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~$ pwd
/home/sneha
```

2-Create a new folder named **`scripts`**:

```
mkdir scripts
```



Ubuntu 24.04.3 LTS

The latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years of free security and maintenance updates, extended up to 12 years with Ubuntu Pro.

Intel or AMD 64-bit architecture Download 5.9GB

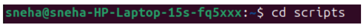
For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors and past releases check out our alternative downloads.

What's new System requirements How to install

- New Desktop installer with support for autoinstall
- New App Center and Firmware Updater applications
- GNOME 46 with support for quarter screen tiling
- Advanced Active Directory Group Policy Object support for Ubuntu Pro users
- Experimental support for TPM-backed Full Disc Encryption and ZFS encryption

3-Navigate into the **scripts** directory:

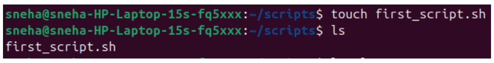
```
cd scripts
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~$ cd scripts
```

4- Create a new shell script file named **first_script.sh**:

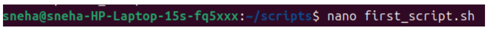
```
touch first_script.sh
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ touch first_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ls
first_script.sh
```

5-Open the script file using **nano** editor:

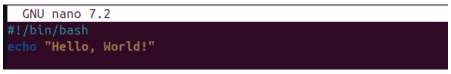
```
nano first_script.sh
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano first_script.sh
```

6-Inside **nano**, write the commands you want the script to execute.

For example,



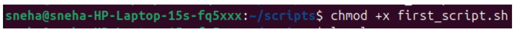
```
  GNU nano 7.2
#!/bin/bash
echo "Hello, World!"
```

7- Save and exit the editor:

- Press Ctrl + X
- Press Y to confirm saving
- Press Enter to finalize

8- Give execute permission to the script:

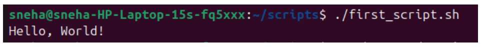
```
chmod +x first_script.sh
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ chmod +x first_script.sh
```

9- Run the script:

```
./first_script.sh
```

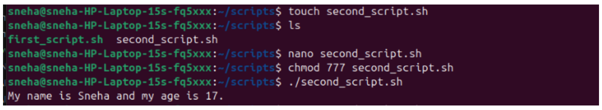


```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./first_script.sh
Hello, World!
```

📝 Variables

Store values inside names!

```
name="Sneha"
age=18
echo "Hi, I'm $name and I'm $age years old."
```

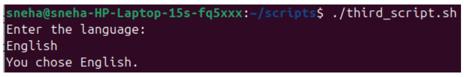


```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ touch second_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ls
first_script.sh  second_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano second_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ chmod 777 second_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./second_script.sh
My name is Sneha and my age is 17.
```

⚡ Tip: No spaces around =

🎤 User Input

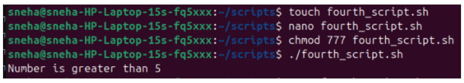
```
echo "Enter your fav subject:"
read subject
echo "You chose $subject "
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./third_script.sh
Enter the language:
English
You chose English.
```

⚖️ If–Else

```
num=10
if [ $num -gt 5 ]; then
  echo "Bigger than 5 "
else
  echo "Smaller or equal "
fi
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ touch fourth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano fourth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ chmod 777 fourth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./fourth_script.sh
Number is greater than 5
```

🧾 Common operators:

- -eq equal
- -ne not equal
- -gt greater
- -lt less

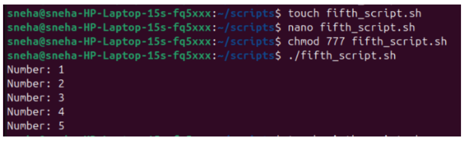
🔁 Loops

🔹 **For Loop**

```
for i in {1..3}
do
  echo "Round $i 🌀"
done
```

🔹 **While Loop**

```
count=1
while [ $count -le 3 ]
do
  echo "Count = $count"
  ((count++))
done
```

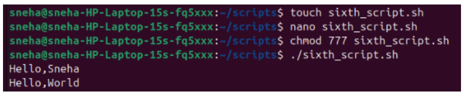


```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ touch fifth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano fifth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ chmod 777 fifth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./fifth_script.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

🔧 Functions

```
greet() {
  echo "Hello, $1 👋"
}

greet "Sneha"
greet "World"
```



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ touch sixth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano sixth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ chmod 777 sixth_script.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./sixth_script.sh
Hello,Sneha
Hello,World
```

💡 Command Line Arguments

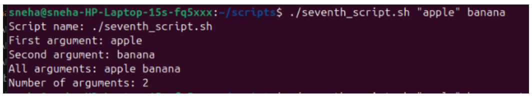
```
#!/bin/bash
echo "Script: $0"
echo "First: $1"
echo "Second: $2"
```

Run:

```
./test.sh apple banana
```

Output:

```
Script: ./test.sh
First: apple
Second: banana
```



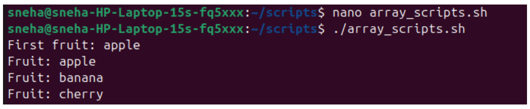
```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./seventh_script.sh "apple" banana
Script name: ./seventh_script.sh
First argument: apple
Second argument: banana
All arguments: apple banana
Number of arguments: 2
```

🍎🍌🍒 Arrays

```
fruits=("apple" "banana" "cherry")
echo "First fruit: ${fruits[0]}"
```

```
for f in "${fruits[@]}"; do
  echo "Fruit: $f"
done
```

OUTPUT:



```
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ nano array_scripts.sh
sneha@sneha-HP-Laptop-15s-fq5xxx:~/scripts$ ./array_scripts.sh
First fruit: apple
Fruit: apple
Fruit: banana
Fruit: cherry
```

Useful Commands 🛠️

- `date` → 📅 current time
- `whoami` → 🙋 current user
- `ls` → 📂 list files
- `pwd` → 📍 working directory
- `cat file.txt` → 📖 show file

Mini Project 🌟 – Auto Backup

```
#!/bin/bash
backup="/tmp/backup_$(date +%H%M).tar.gz"
tar -czf $backup $HOME
echo "Backup saved at $backup ✅"
```

📒 Cheat Sheet (Quick Recall)

Feature	Syntax	Example
Print	`echo`	`echo "Hi"`
Var	`var=value`	`x=10`
Input	`read var`	`read name`
If	`if [ cond ]`	`if [ $x -gt 5 ]`
For	`for i in {1..5}`	`echo $i`
While	`while [ cond ]`	`((count++))`

Extra Questions:

1. What is the Difference Between `chmod` and `chown`?

Command	Function	Affects
`chmod`	Modifies file or directory permissions	Who can read/write/execute
`chown`	Changes file or directory ownership	Who owns the file and group

- `chmod` controls **access rights** (read, write, execute).
- `chown` controls **ownership** (user and group).

---

2. How to Check Current Directory and User?

--> To check your current working directory:

```
pwd
```

Output: Displays the full path of your present location in the filesystem.

--> To check your current user:

```
whoami
```

Output: Displays the username of the currently logged-in user.
