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:

Command : `chmod`

-used to modify access permissions for files and directories

Basic Syntax:

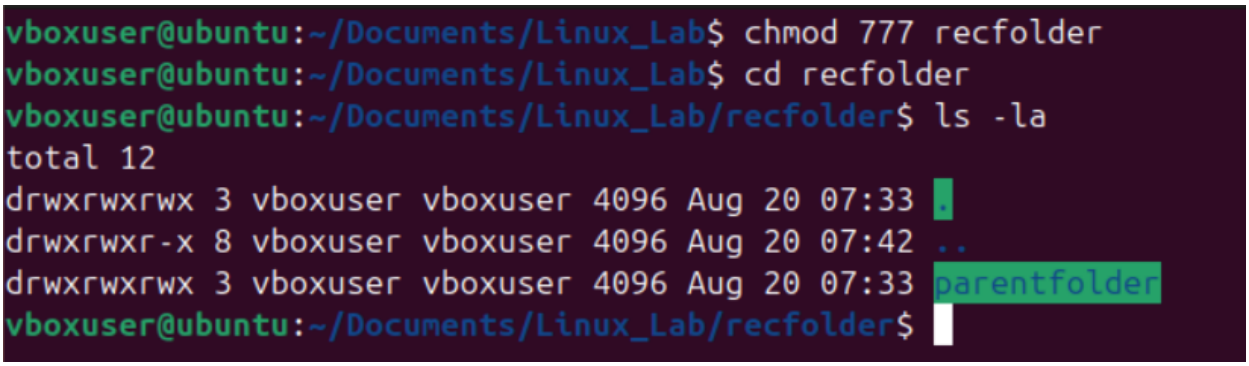
```
chmod 741 file.txt
chmod u+x file.txt
```

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

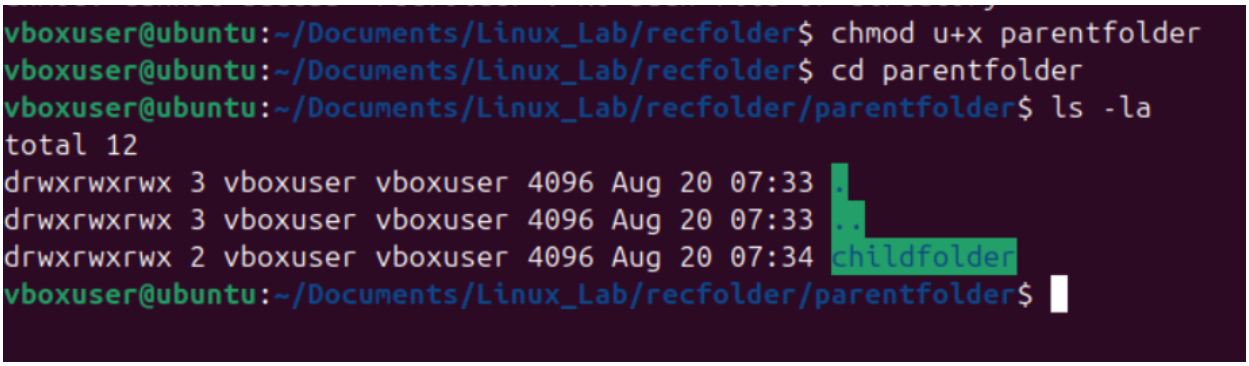
Explanation:

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

Numeric method:



Symbolic method:



Recursive Permission Changes

-Apply changes to all files and subdirectories.

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

- `-R` → Recursive flag

Code Snippet:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ mkdir recfolder
vboxuser@ubuntu:~/Documents/Linux_Lab$ cd recfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ mkdir parentfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ cd parentfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder$ mkdir childfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder$ cd childfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$ touch "text.txt"
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$ chmod 000 "text.txt"
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$ cd ..
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder$ chmod 000 childfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder$ cd ..
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ chmod 000 parentfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ cd ..
vboxuser@ubuntu:~/Documents/Linux_Lab$ chmod 000 recfolder
vboxuser@ubuntu:~/Documents/Linux_Lab$ chmod -R 777 recfolder
vboxuser@ubuntu:~/Documents/Linux_Lab$ cd recfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ ls -la
total 12
drwxrwxrwx 3 vboxuser vboxuser 4096 Aug 20 07:33 .
drwxrwxr-x 8 vboxuser vboxuser 4096 Aug 20 07:33 ..
drwxrwxrwx 3 vboxuser vboxuser 4096 Aug 20 07:33 parentfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder$ cd parentfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder$ cd childfolder
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$ ls -la "text.txt"
-rwxrwxrwx 1 vboxuser vboxuser 0 Aug 20 07:34 text.txt
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$
```

Command : chown

-Changes File Ownership.

Example:

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

Explanation:

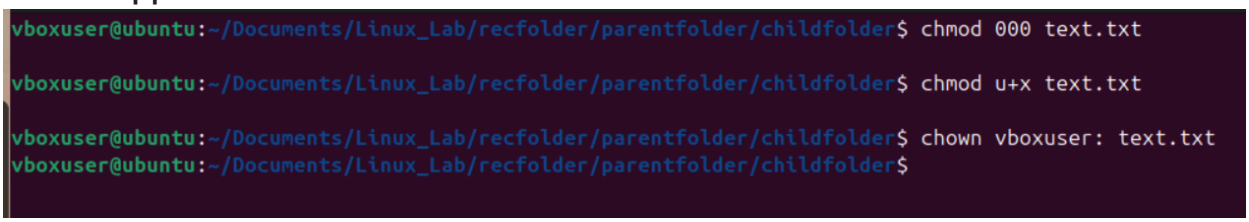
Assigns ownership of `data.txt` to user `newon` and group `group2` . **Code Snippet:**

```
vboxuser@ubuntu:~/Documents/Linux_Lab/recfolder/parentfolder/childfolder$ chown
vboxuser: "text.txt"
vboxuser@ubuntu:~/Documents/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:



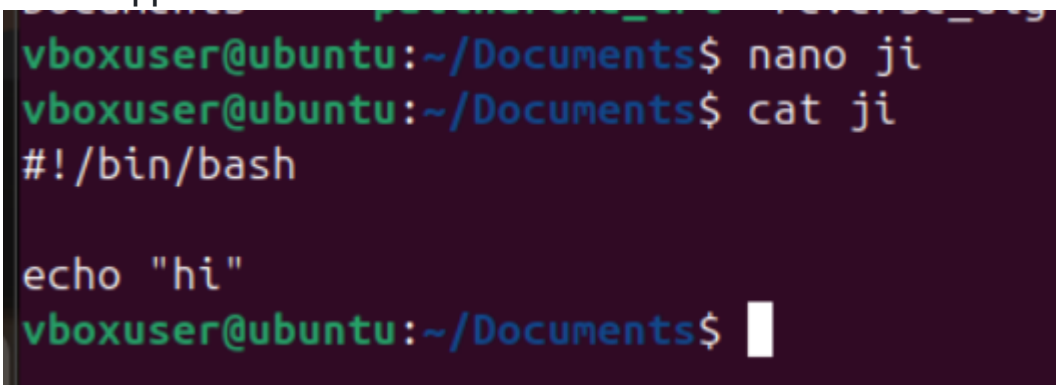
Commands used: nano , echo

`nano` -Create and edit shell scripts
`echo` -prints a string in the output

```
Create and Edit a file:
nano file.sh
Type:
#!/bin/bash
echo "Hello, World!"

Save and exit (CTRL+O, CTRL+X in nano).

Make it executable:
chmod +x hello.sh

Run it:
./hello.sh
```

```
Output:
Hello, World!
```

Code Snippet:



Commands used : read , echo

read → takes input from the user.
$username → retrieves the value.

```
#!/bin/bash

echo "Enter your name:"
read username

echo "Hello, $username! Welcome to shell scripting."
```

Code Snippet:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat Experiment5.sh
#!/bin/bash
echo "hi"

name="Divyanshi"
age=17

echo "My name is $name and I am $age years old."

echo $HOME

echo "Whats your fav food:"
read food
echo "My fav food is $food"vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Output:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ ./Experiment5.sh
hi
My name is Divyanshi and I am 17 years old.
/home/vboxuser
Whats your fav food:
pizza
My fav food is pizza
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Statements used : `if`, `else`

-provides conditional execution, allowing different blocks of code to run based on whether a condition evaluates to true or false.

```
Our code's objective:
Checks some specified conditions with if-else.
```

Code Snippet:

vboxuser@ubuntu: ~/Documents/Linux_Lab

```
#!/bin/bash
num=8
empt=""
greet="hello"
name="hari"
age=50
if [ $num -eq 8 ]; then
        echo "$num = 8"
else
        echo "$num is not equal to 8"
fi
if [ -z  "$empt" ]; then
        echo "string variable empt is empty"

fi
if [ $greet = "hello" ]; then
        echo "hello, there!"
fi
if [ -n "$greet" ]; then
        echo "greet string variable is not empty"
fi

if [ "$name" = "hari" ] && [ "$age" -gt 49 ]; then
        echo "The name and age is satisfied"
fi
if [ "$num" -eq 8] || [ "$num" -gt 8 ]; then
        echo "num satisfies the condition and might be greater than 8"
fi
shell.md (END)
```

OUTPUT:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ ./shell.md
8 = 8
string variable empt is empty
hello, there!
greet string variable is not empty
The name and age is satisfied
num satisfies the condition and might be greater than 8
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Lab5 – PRACTICE

🧪 Practice Experiment – Creating Users and Groups

This section documents the commands executed during the practice experiment, along with brief explanations and screenshots.

Procedure:

```
1. Create a New User
2. Create a New Group
3. Add User to the Group
4. Create a File
```

5. Assign Ownership to User and Group
6. Verify Ownership

Code Snippet:

```
vboxuser@ubuntu:~/Documents$ sudo adduser tree
info: Adding user `tree' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `tree' (1001) ...
info: Adding new user `tree' (1001) with group `tree (1001)' ...
info: Creating home directory `/home/tree' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for tree
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `tree' to supplemental / extra groups `users' ...
info: Adding user `tree' to group `users' ...
vboxuser@ubuntu:~/Documents$ sudo groupadd forest
vboxuser@ubuntu:~/Documents$ sudo usermod -aG forest tree
vboxuser@ubuntu:~/Documents$ touch ownershipfile.txt
vboxuser@ubuntu:~/Documents$ sudo chown forest:tree ownershipfile.txt
chown: invalid user: ‘forest:tree’
vboxuser@ubuntu:~/Documents$ sudo chown tree:forest ownershipfile.txt
vboxuser@ubuntu:~/Documents$ ls -l ownershipfile.txt
-rw-rw-r-- 1 tree forest 0 Aug 28 14:58 ownershipfile.txt
vboxuser@ubuntu:~/Documents$
```

Explanation:

```
sudo useradd -m newuser
-Creates a new user `newuser` and sets up a home directory at `/home/newuse

sudo groupadd newgroup
-Creates a new group named `newgroup`.

sudo usermod -aG newgroup newuser
-Adds `newuser` to `newgroup` without removing them from existing groups.

touch testfile.txt
-Creates an empty file named `testfile.txt`.

sudo chown newuser:newgroup testfile.txt
-Assign ownership of the file to newuser and newgroup
```

```
ls -l testfile.txt
-Verify Ownership
```

Lab6 – Shell Scripting Basics

Commands include : Accessing Variables, Taking a user's input

To access a variable in shell scripting we prefix its name with a dollar sign
We use `read` command to take a user's input

```
Code
#!/bin/bash
echo "hi"
name="Divyanshi"
age=17
echo "My name is $name and I am $age years old."
echo $HOME
echo "Whats your fav food:"
read food
echo "My fav food is $food"
```

Code Snippet:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat Experiment5.sh
#!/bin/bash
echo "hi"

name="Divyanshi"
age=17

echo "My name is $name and I am $age years old."

echo $HOME

echo "Whats your fav food:"
read food
echo "My fav food is $food"vboxuser@ubuntu:~/Documents/Linux_Lab$
```

OUTPUT:

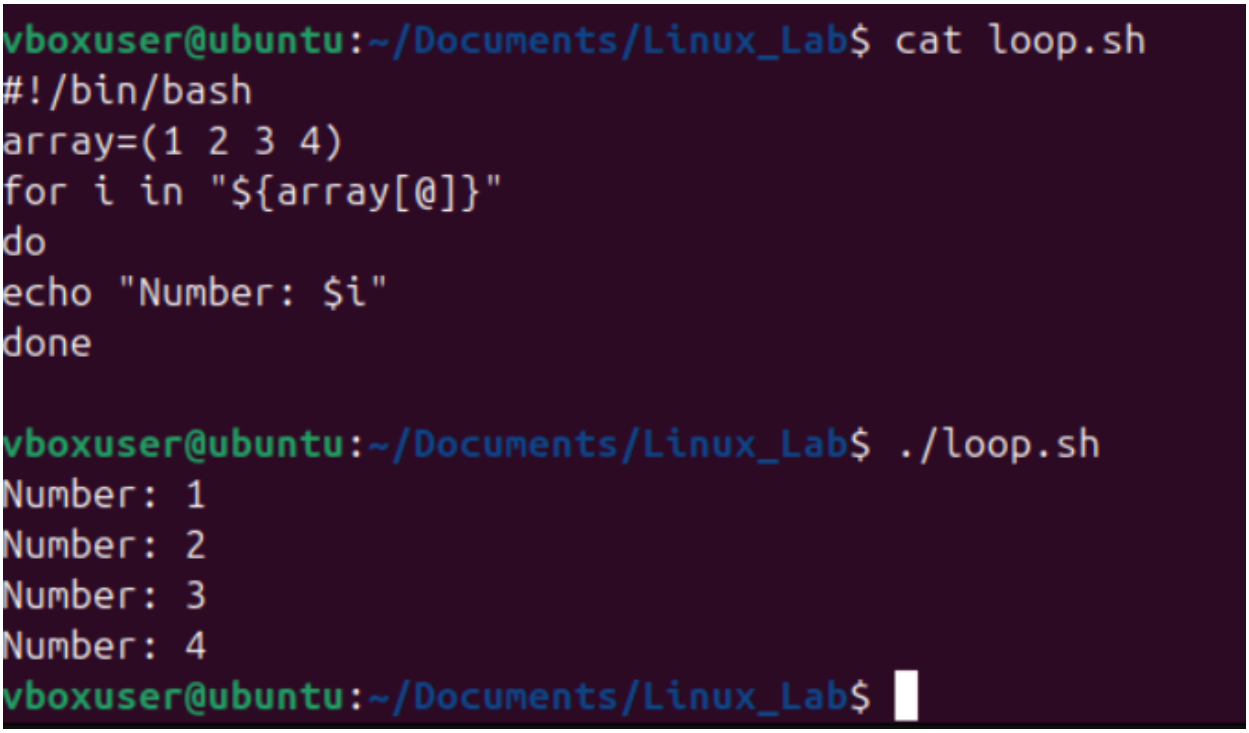
```
vboxuser@ubuntu:~/Documents/Linux_Lab$ ./Experiment5.sh
hi
My name is Divyanshi and I am 17 years old.
/home/vboxuser
Whats your fav food:
pizza
My fav food is pizza
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Command : for loop

-control flow statement used to iterate over a list of items and execute a block of commands for each item.

Code Snippet:

Objective: Iterate over an array with for loop.



Command : `while loop`

-a while loop repeatedly executes a block of commands as long as a specified condition remains true.

Code Snippet:

Objective: Looping through an array by adding +1 a variable in each iteration.

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat while_loop.sh
#!/bin/bash

list=(2459)

index=0

while [ $index -le 4 ]
do echo "${list[@]}"
        ((index += 1))
done
vboxuser@ubuntu:~/Documents/Linux_Lab$ ./while_loop.sh
2459
2459
2459
2459
2459
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Command : `until` loop

-The until loop in shell scripting is a control flow statement that repeatedly executes a block of commands as long as a given condition remains false.

Code Snippet:

Objective: A chosen element from an array is getting subtracted till it becomes equal to another chosen element from the same array.

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat until_loop.sh
#!/bin/bash

array=(4 1 0 5)
first_element=${array[0]}
echo "The number $first_element is getting minused till it reaches 0"
until [ "$first_element"  -eq "${array[2]}" ]
do
        echo "Still not equal to ${array[2]}"
        first_element=$((first_element - 1))
        if [ "$first_element" -eq "${array[2]}" ]; then
                echo "Finally equal to ${array[2]}"
                break
        fi
done
vboxuser@ubuntu:~/Documents/Linux_Lab$ ./until_loop.sh
The number 4 is getting minused till it reaches 0
Still not equal to 0
Still not equal to 0
Still not equal to 0
Still not equal to 0
Finally equal to 0
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Commands include: Creating funtions

Objective of this code: To say bye to the given input.

Code Snippet:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat function.sh
#!/bin/bash

goodbye() {
        echo "Alwida , $1 $2!"
}

goodbye mere fans

vboxuser@ubuntu:~/Documents/Linux_Lab$ ./function.sh
Alwida , mere fans!
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

Commands include: Iterating over the elements of an array named `fruits`

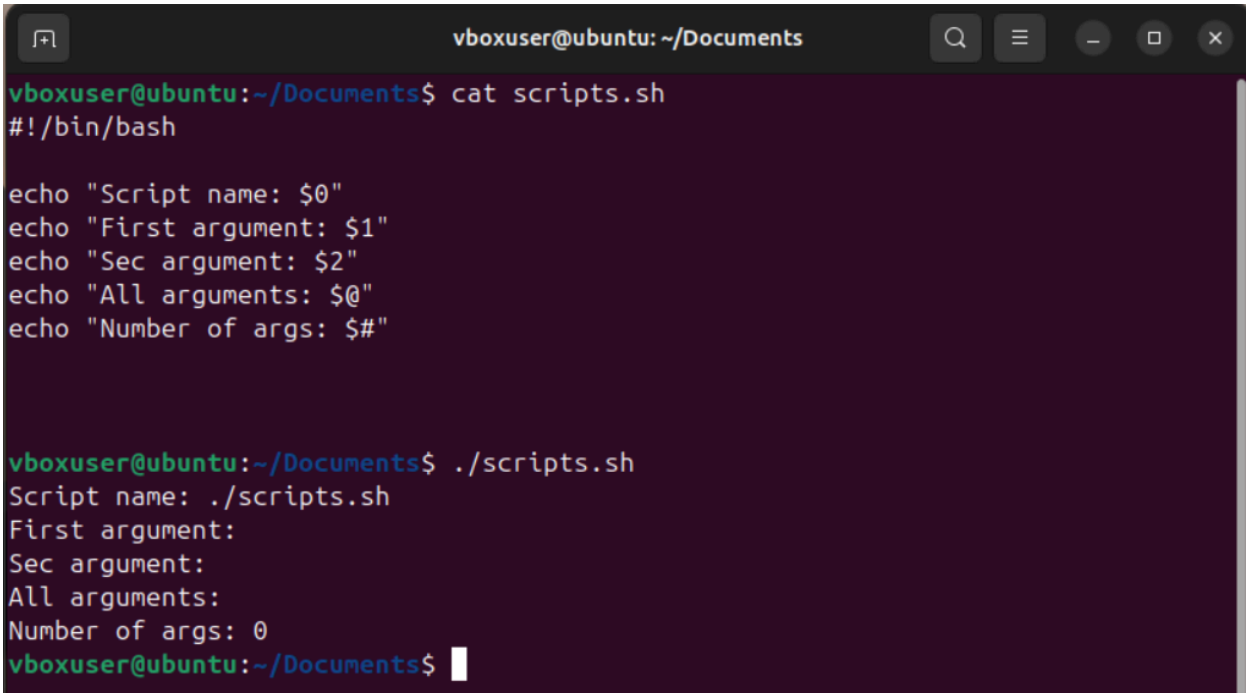
Code Snippet:

```
vboxuser@ubuntu:~/Documents/Linux_Lab$ cat loop.sh
#!/bin/bash
array=(1 2 3 4)
for i in "${array[@]}"
do
echo "Number: $i"
done

vboxuser@ubuntu:~/Documents/Linux_Lab$ ./loop.sh
Number: 1
Number: 2
Number: 3
Number: 4
vboxuser@ubuntu:~/Documents/Linux_Lab$
```

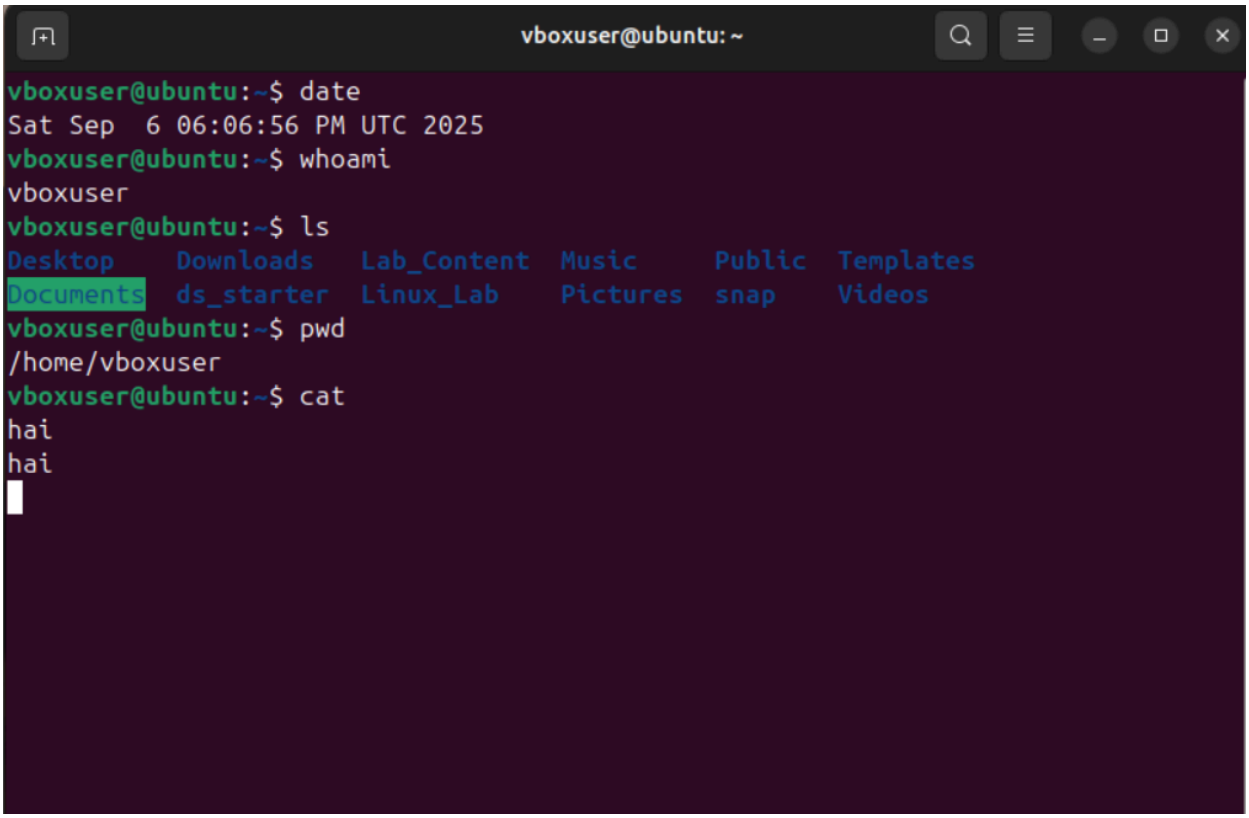
Some command line arguments:

- Accessing the arguments passed by user in command line.

Code snippet:



Useful commands: `date`, `whoami`, `ls`, `cat`



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.