



**Daffodil**  
*International*  
**University**

## Lab Report

Course Code: CSE324

Course Title: Operating Systems Lab

Experiment No: 01

Experiment Name: Introduction to Linus Commands

Submitted To:

Teacher Name: Nushrat Jahan Oyshi

Department: CSE

Daffodil International University.

Submitted By:

Name: Sumaiya Salekin Saba

ID: 0242220005101631

Section: 63\_M

Semester: Spring-25

Department: CSE

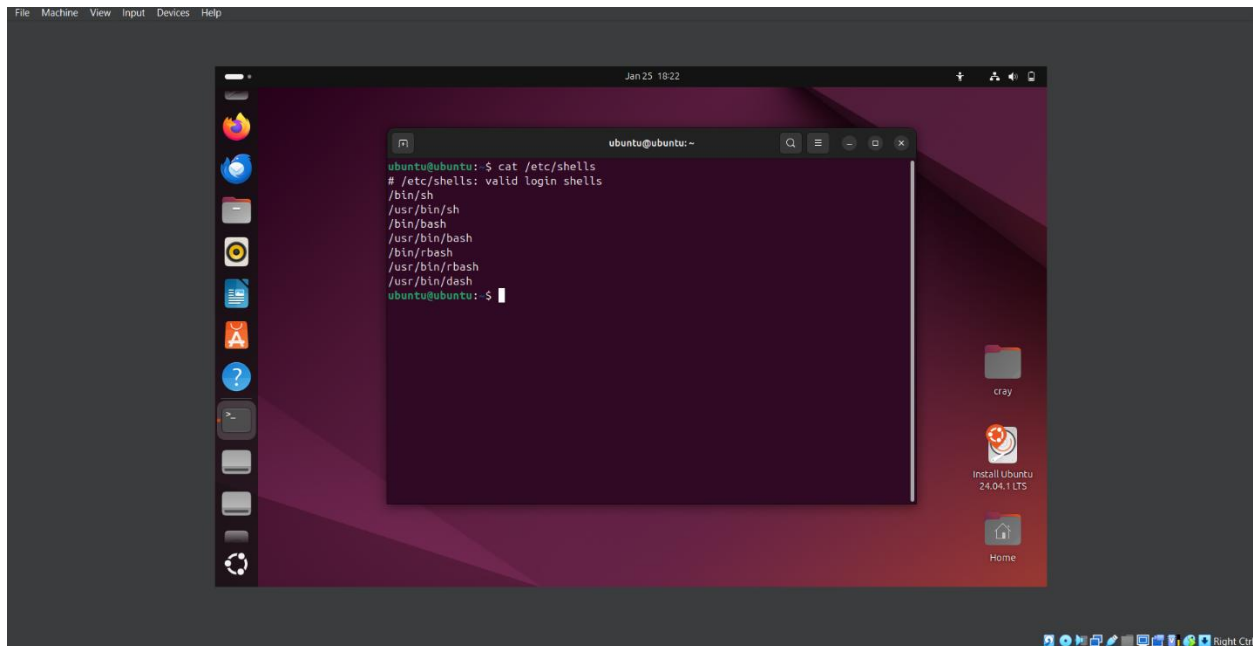
Daffodil International University.

Submission Date: 25 January, 2025

## Introduction:

This report delves into essential Linux commands and their practical uses for managing files, directories, and scripts. It aims to demonstrate effective navigation of the Linux filesystem, modification of file permissions, and the creation and execution of Bash scripts. Through hands-on examples—such as creating directories, inspecting file permissions, and running a custom script—this report highlights Linux's flexibility and capabilities as an operating system. Key tasks include folder and file creation, permission management, and executing a Bash script that utilizes user input and environmental variables, showcasing fundamental skills for budding computer science professionals.

## Working with Root folders



In Linux, the root directory (/) acts as the foundation of the entire filesystem, with all other files and directories branching out from it. Among its key subdirectories is /etc, which plays a vital role by storing configuration files necessary for the system's functionality.

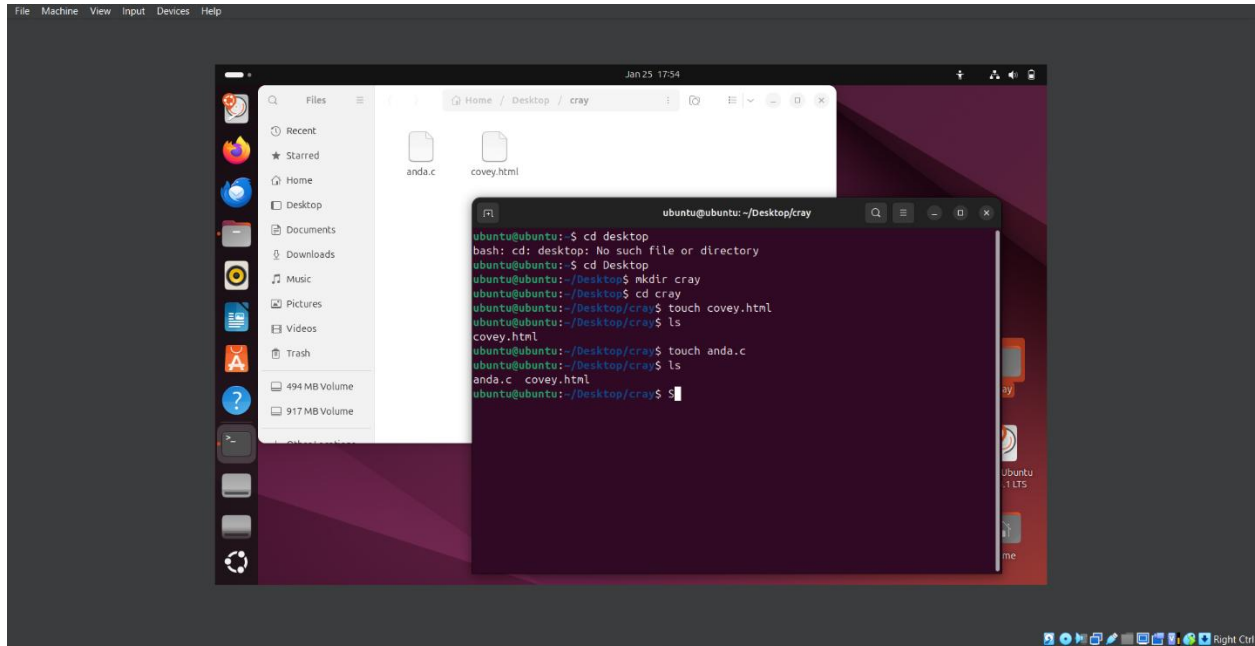
One example of accessing such files is viewing the contents of the /etc/shells file using the command:

**\$ cat /etc/shells**

This command retrieves and displays the list of valid login shells available on the system. While the /etc directory is part of the root-level structure, this specific command does not require

administrative privileges because /etc/shells is readable by all users. However, modifying files in the /etc directory typically requires superuser (root) privileges.

## Creating Directories and files:



In Linux, directories and files can be easily created and organized using terminal commands. Here is an example demonstrating this process:

Creating a Folder: I used the mkdir command to create a folder named cray with the following command:

```
$ mkdir cray
```

Navigating into the Folder:

To work inside the newly created folder, I used the cd command: `$ cd cray`

Creating a File:

Within the cray folder, I created files named covey.html & anda.c using the touch command:

```
$ touch covey.html
```

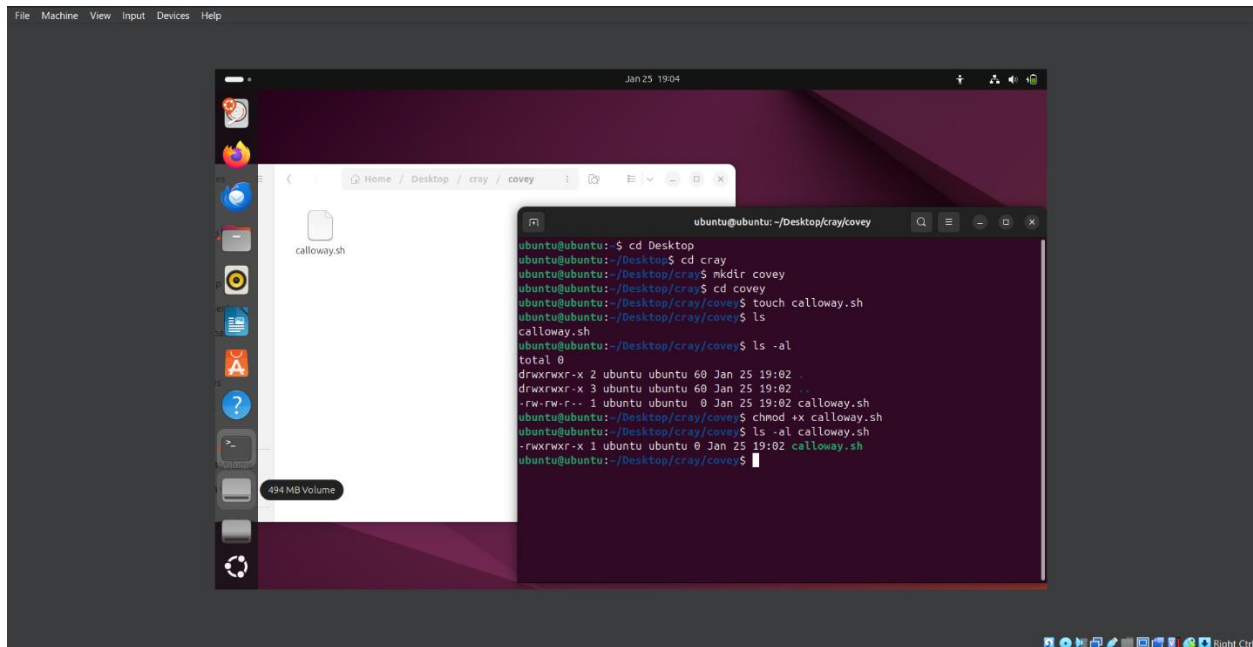
```
$ touch anda.c
```

Viewing the Folder Contents:

To confirm the creation of the file, I used the ls command to list the contents of the folder:

```
$ ls
```

### File Permissions:



In Linux, file permissions determine who can read, write, or execute a file. These permissions can be viewed and modified using specific commands. Below is an example of working with file permissions:

Viewing File Permissions: I used the ls -al command to list all files and directories in the current folder along with their permissions:

```
$ ls -al
```

This command displayed detailed information, including the file name, ownership, and permissions. For the file calloway.sh, the permissions were initially displayed as -rw-rw-r--, indicating it was readable and writable by the owner but not executable.

Changing Permissions: To make the file calloway.sh executable, I used the chmod command with the +x flag:

## **\$ chmod +x calloway.sh**

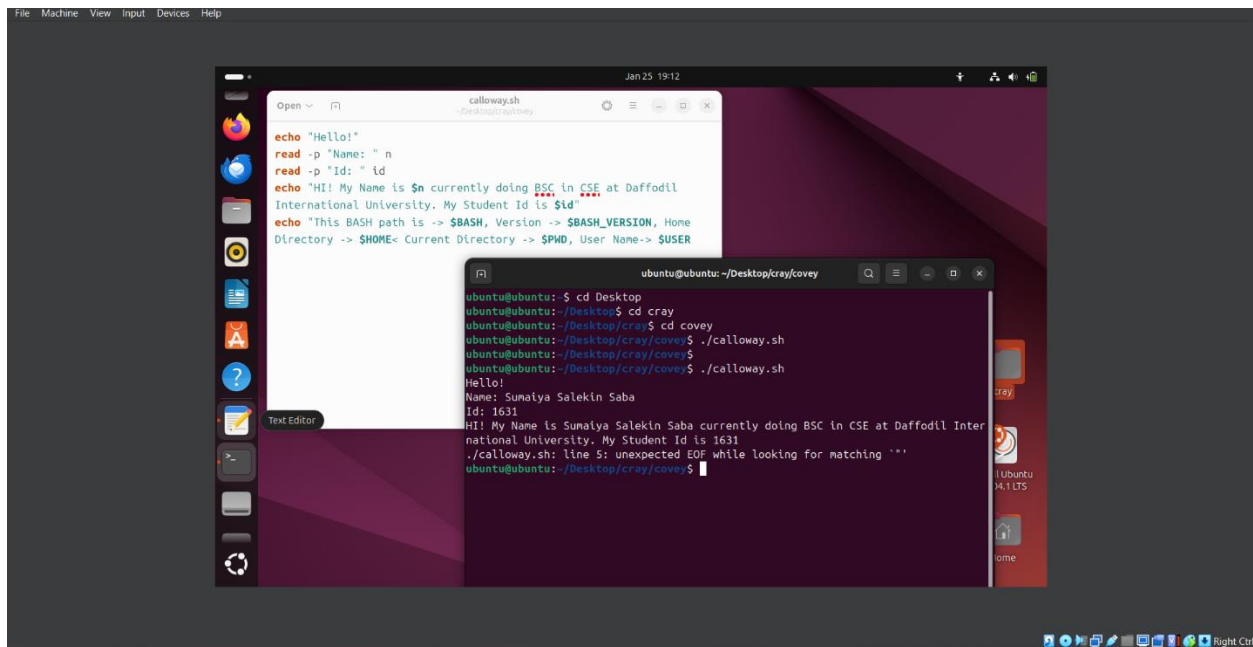
This command added execute permissions to the file, allowing it to be run as a script.

Verifying the Updated Permissions: After changing the permissions, I used the following command to check the updated permissions of calloway.sh:

## **\$ ls -al calloway.sh**

The output showed the new permissions as -rwxrwxr-x, indicating that the file is now executable by the owner while retaining read permissions for others.

## **Running Bash Script using calloway.sh and use of variables:**



```
File Machine View Input Devices Help
Jan 25 19:12
calloway.sh
echo "Hello!"
read -p "Name: " n
read -p "Id: " id
echo "HI! My Name is $n currently doing BSC in CSE at Daffodil International University. My Student Id is $id"
echo "This BASH path is -> $BASH, Version -> $BASH_VERSION, Home Directory -> $HOME< Current Directory -> $PWD, User Name-> $USER

ubuntu@ubuntu: ~/Desktop/cray/covey
ubuntu@ubuntu:~/Desktop$ cd Desktop
ubuntu@ubuntu:~/Desktop$ cd cray
ubuntu@ubuntu:~/Desktop/cray$ cd covey
ubuntu@ubuntu:~/Desktop/cray/covey$ ./calloway.sh
ubuntu@ubuntu:~/Desktop/cray/covey$ ./calloway.sh
Hello!
Name: Sunaiya Salekin Saba
Id: 1631
HI! My Name is Sunaiya Salekin Saba currently doing BSC in CSE at Daffodil International University. My Student Id is 1631
./calloway.sh: line 5: unexpected EOF while looking for matching `''
ubuntu@ubuntu:~/Desktop/cray/covey$
```

Bash scripts allow automation and execution of commands sequentially. Below is an example of creating and running a Bash script:

Writing the Script: In the calloway.sh file, I wrote the following code:

```
$ echo "Hello!"
```

```
$ read -p "Name: " n
```

```
$ read -p "ID: " id
```

```
$ echo "Hi! My name is $n currently doing BSc in CSE at Daffodil International  
University. My Student ID is $id"
```

```
$ echo "This BASH path is -> $BASH, Version -> $BASH_VERSION, Home Directory -  
> $HOME, Current Directory -> $PWD, User Name -> $USER"
```

Running the Script: To execute the script, I used the command:

```
$ ./calloway.sh
```

Providing Input: When the script prompted for inputs, I entered the following:

```
$ Name: Sumaiya Salekin Saba
```

```
$ ID: 1631
```

Output:

After providing the inputs, the script displayed the following output:

```
Hello! Name: Sumaiya Salekin Saba ID:1631 Hi! My name is Sumaiya Salekin  
Saba doing BSc in CSE at Daffodil International University. My Student ID is  
YourIDHere
```

```
This BASH path is -> /bin/bash, Version -> 5.x.x, Home Directory ->  
/home/username, Current Directory ->
```

```
/path/to/current/directory, User Name -> username
```

**This example demonstrates how Bash scripts can interact with users through input**

## **Conclusion:**

This report focused on fundamental Linux commands and operations, beginning with directory and file creation using commands like `mkdir`, `cd`, and `touch`. We then examined file permission management with `chmod`, ensuring secure control over file accessibility. Additionally, we demonstrated the creation and execution of a Bash script, showcasing Linux's ability to automate tasks, process user input, and utilize environmental variables for dynamic results.

These exercises provided practical experience with Linux's command-line interface, laying a solid foundation for advanced topics in system management and scripting. The skills acquired are invaluable for academic projects, real-world problem-solving, and professional pursuits in computer science.