**Lab Report 1**



Course Title: Operating Systems Lab Course Code: CSE324

# Submitted by

**Wasikur Chowdhury Wasif**

Student ID: 0242220005101955

Section: 63\_M1

Semester: Spring-25

Department: Computer Science and Engineering

# Submitted to

**Nushrat Jahan Oyshi**

Lecturer

Computer Science & Engineering Daffodil International University

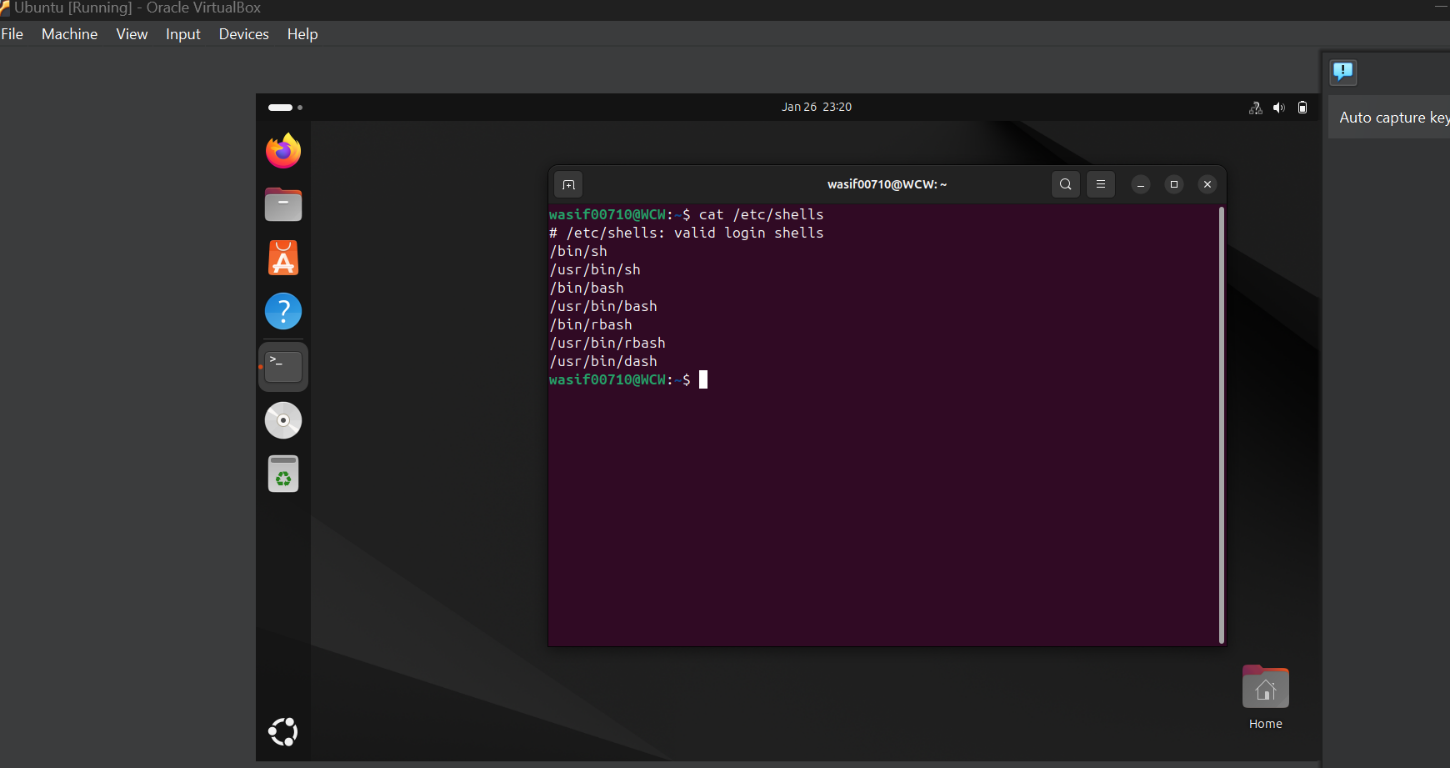
**Submission Date: 27 January, 2025**

**Introduction**

This report explores various fundamental Linux commands and their applications in managing files, directories, and scripts. The objective is to demonstrate how to navigate the Linux filesystem, modify file permissions, and create and execute Bash scripts effectively. Through practical examples, such as viewing system files like **/etc/shells,** creating folders and files using **mkdir** and **touch,** and making scripts executable with **chmod,** this report highlights the flexibility and power of Linux as an operating system.

The tasks performed include creating and organizing directories and files, managing file permissions, and executing a Bash script that incorporates user inputs and environmental variables. These foundational skills are crucial for anyone looking to work with the Linux operating system effectively.

## Working with Root folders



In Linux, the root directory (/) serves as the starting point for the entire filesystem. All other directories and files are organized under this root directory. The /etc directory, a critical subdirectory within the root, contains configuration files essential for system operations.

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.

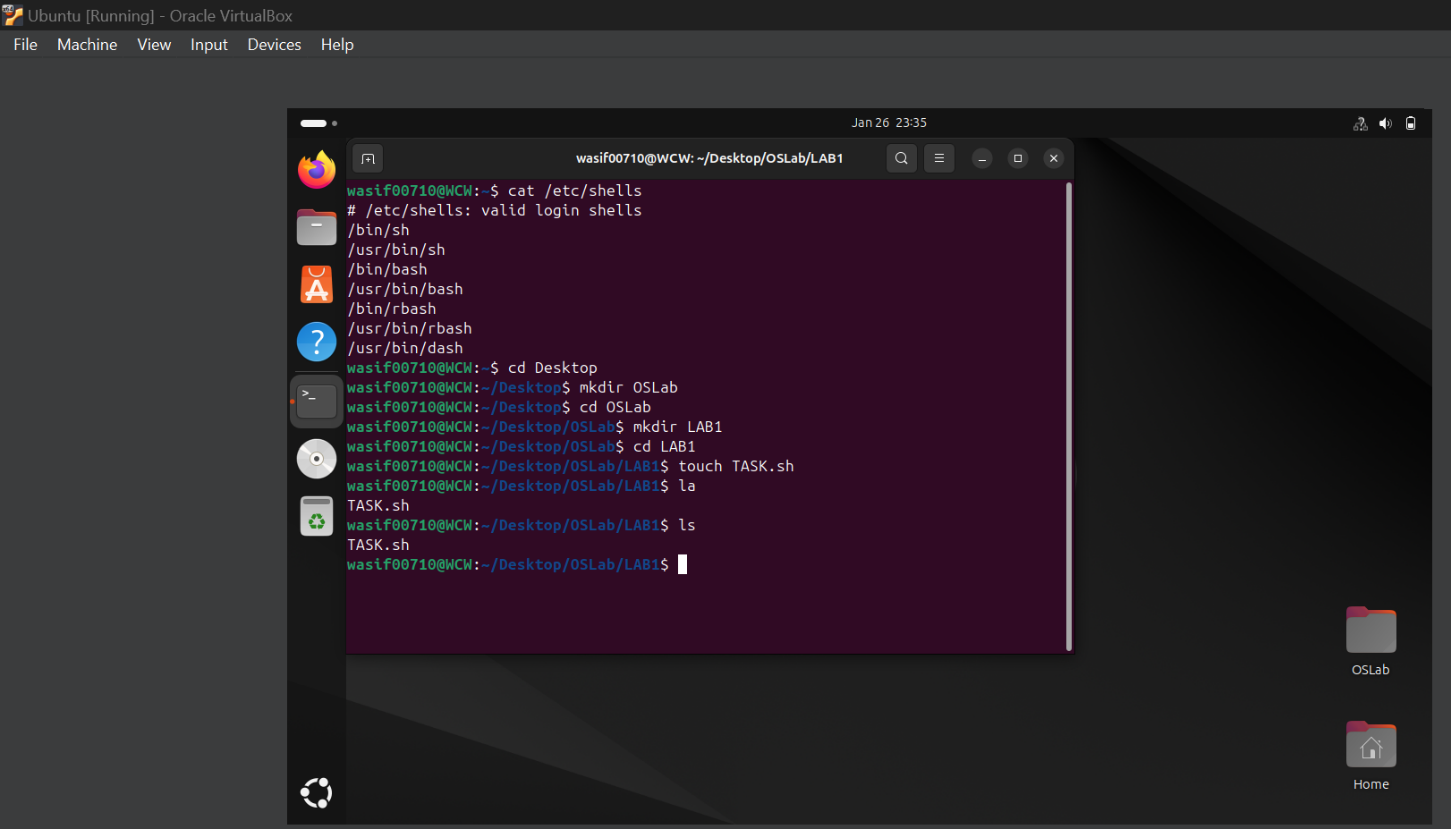
## Creating Directories and files

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 OSLab with the following command:

**$ mkdir OSLab**



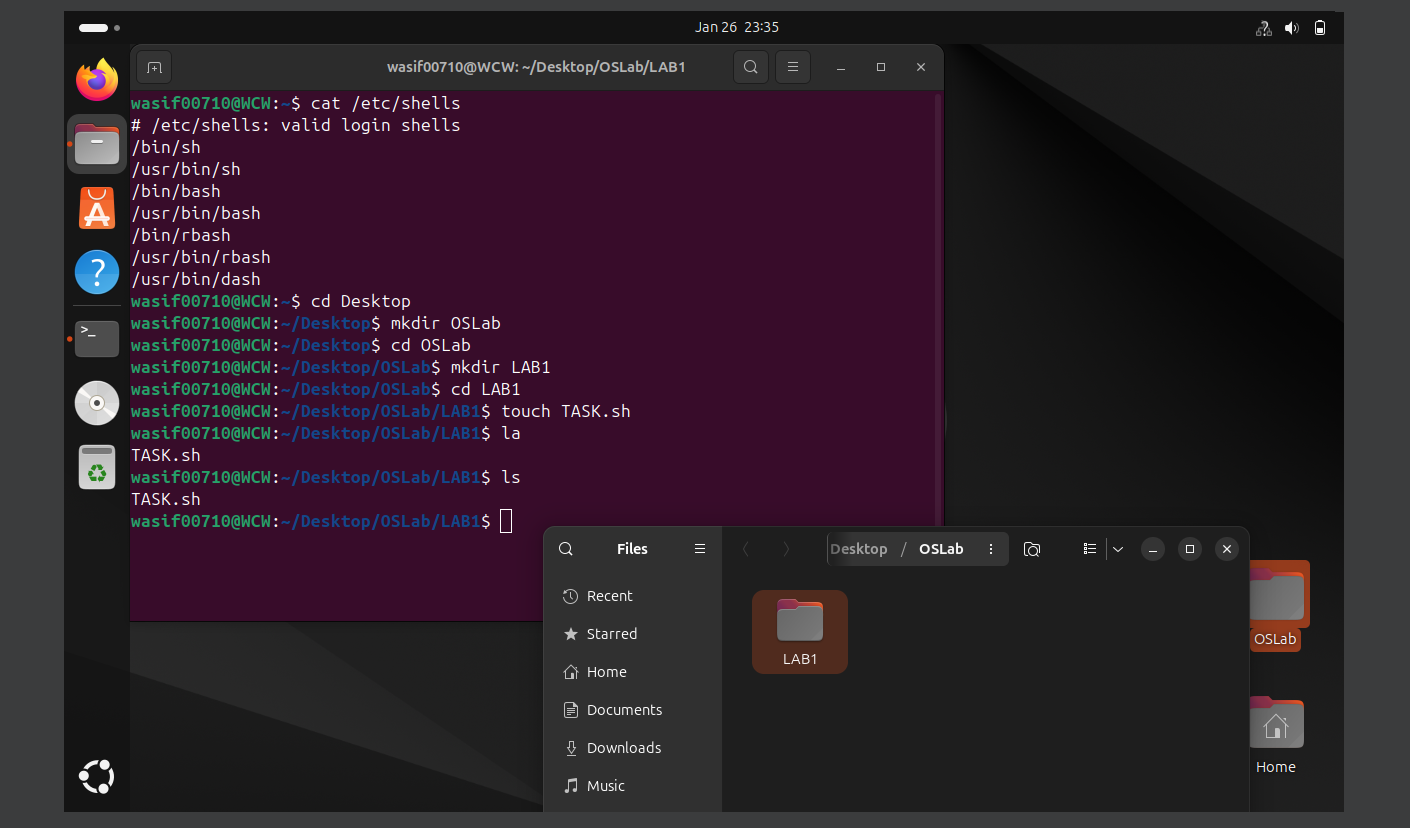
**Navigating into the Folder:**

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

**$ cd OSLab**

**Creating a Subfolder:**

Inside OSLab, I created another folder named LAB1 using:

**$ mkdir LAB1**

**Navigating into the Subfolder:**

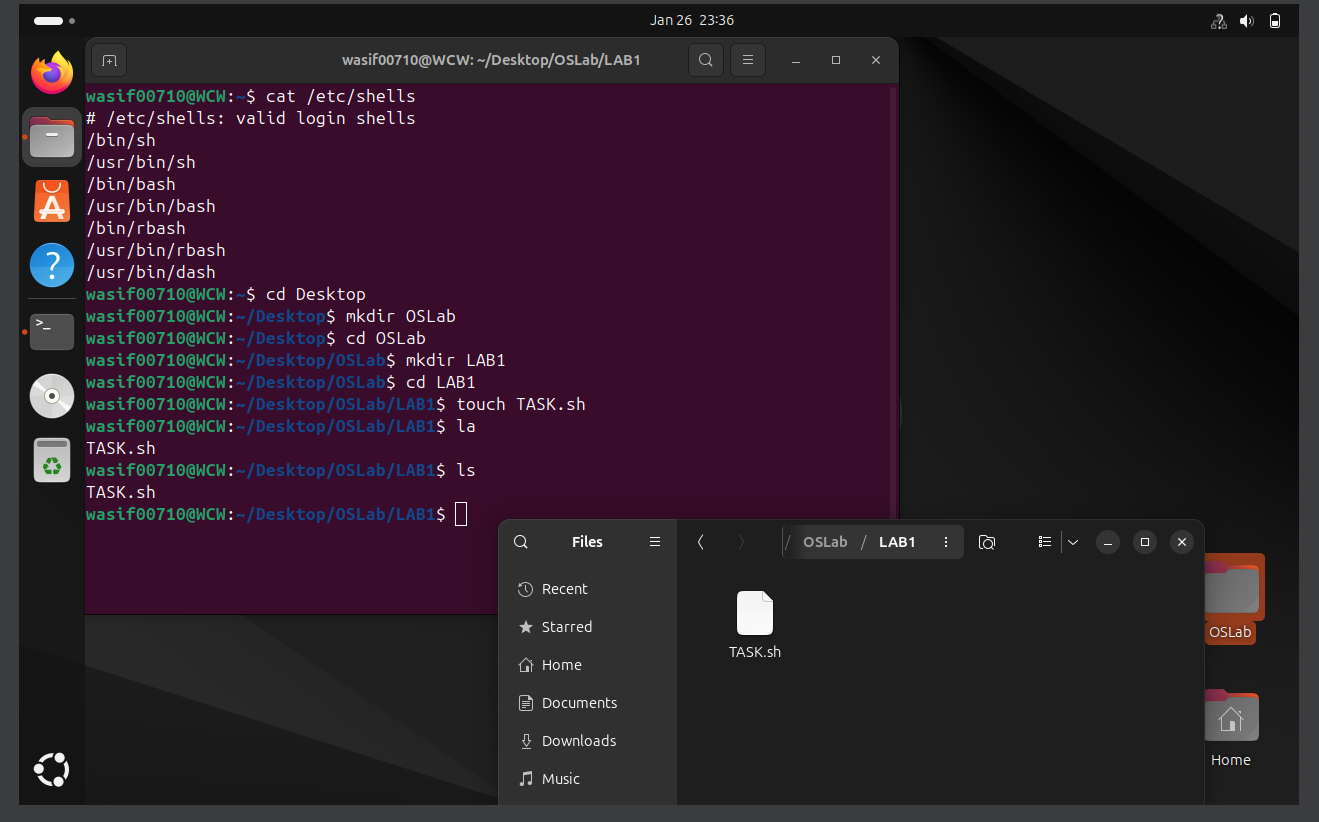
To access the Lab1 folder, I navigated into it using:

**$ cd Lab1**

**Creating a File:**

Within the Lab1 folder, I created a file named TASK.sh using the touch command:

**$ touch TASK.sh**



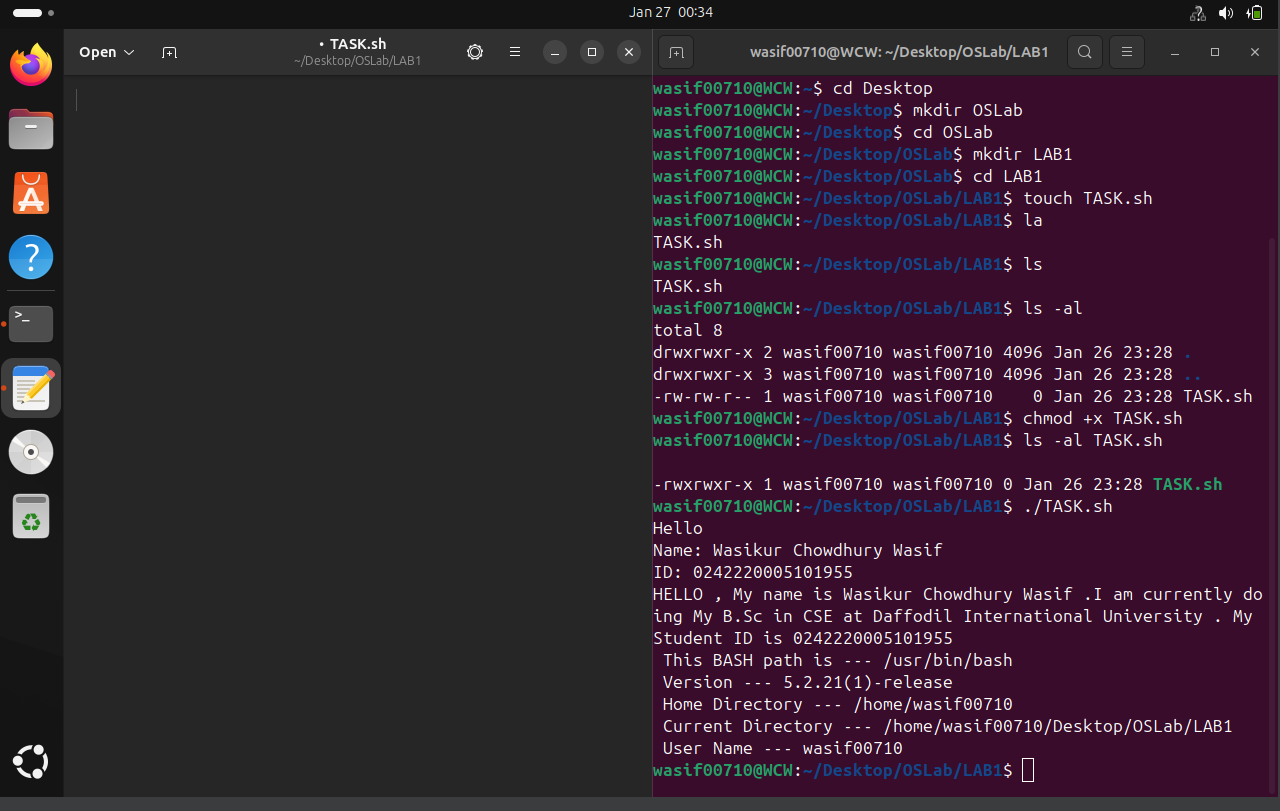
**Viewing the Folder Contents:**

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

**$ ls**

The output showed the file TASK.sh in the folder.

## 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 **TASK.sh**, the permissions were initially displayed as

**-rw-r--r--,** indicating it was readable and writable by the owner but not executable.

**Changing Permissions:**

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

**$ chmod +x TASK.sh**

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

**Verifying the Updated Permissions:**

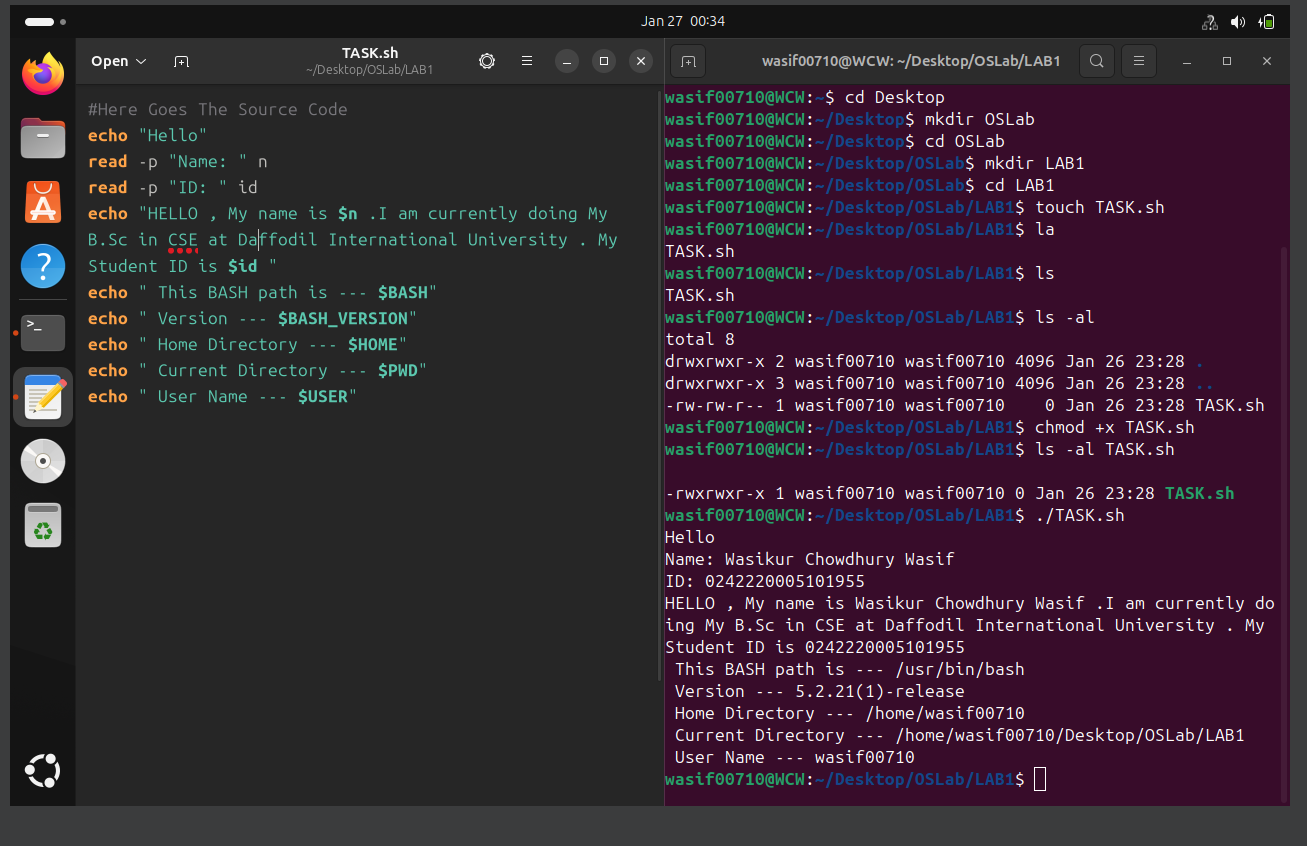
After changing the permissions, I used the following command to check the updated permissions of TASK.sh:

**$ ls -al TASK.sh**

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

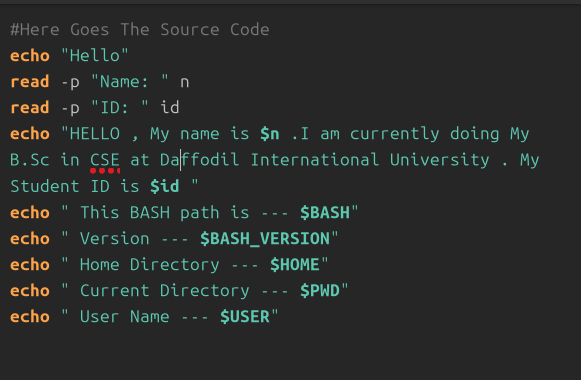
**Running Bash script using TASK.sh and use of variables**

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 TASK.sh file, I wrote the following code:



**Running the Script:**

To execute the script, I used the command:

**$ ./TASK.sh**

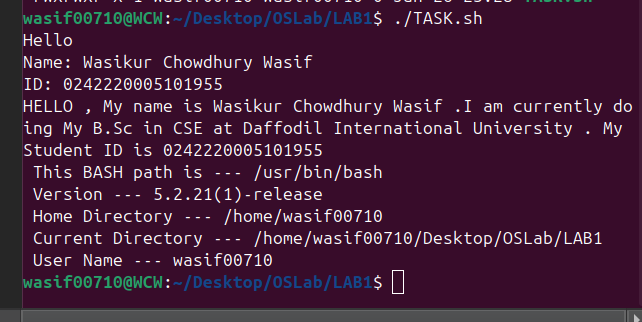
**Providing Input:**

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

**$ Name: Wasikur Chowdhury Wasif**

**$ ID: 0242220005101955**

**Output:** After providing the inputs, the script displayed the following

**Output:**

**Conclusion**

This report explored fundamental Linux commands, including creating directories and files with **mkdir**, **cd**, and **touch**, managing file permissions with **chmod,** and automating tasks with Bash scripts. These exercises demonstrated the importance of navigating the Linux filesystem, controlling access through permissions, and leveraging scripts for efficient task execution. The skills gained are foundational for advancing in system management and scripting, essential for academic and professional pursuits in computer science.

**Discussion**

The report emphasized Linux's flexibility and power, from managing directories and permissions to automating tasks with Bash scripts. Understanding the hierarchical filesystem and file permissions is crucial for maintaining security and efficiency in multi-user environments. Writing interactive Bash scripts highlighted the practical benefits of automation in streamlining workflows. These tasks provided valuable hands-on experience, forming a strong base for tackling advanced Linux operations in real-world scenarios.