

# Operating Systems – COC 3071L

SE 5th A – Fall 2025

## Lab 2: Linux Basics and Introduction

### Part 1: Linux Environment Orientation

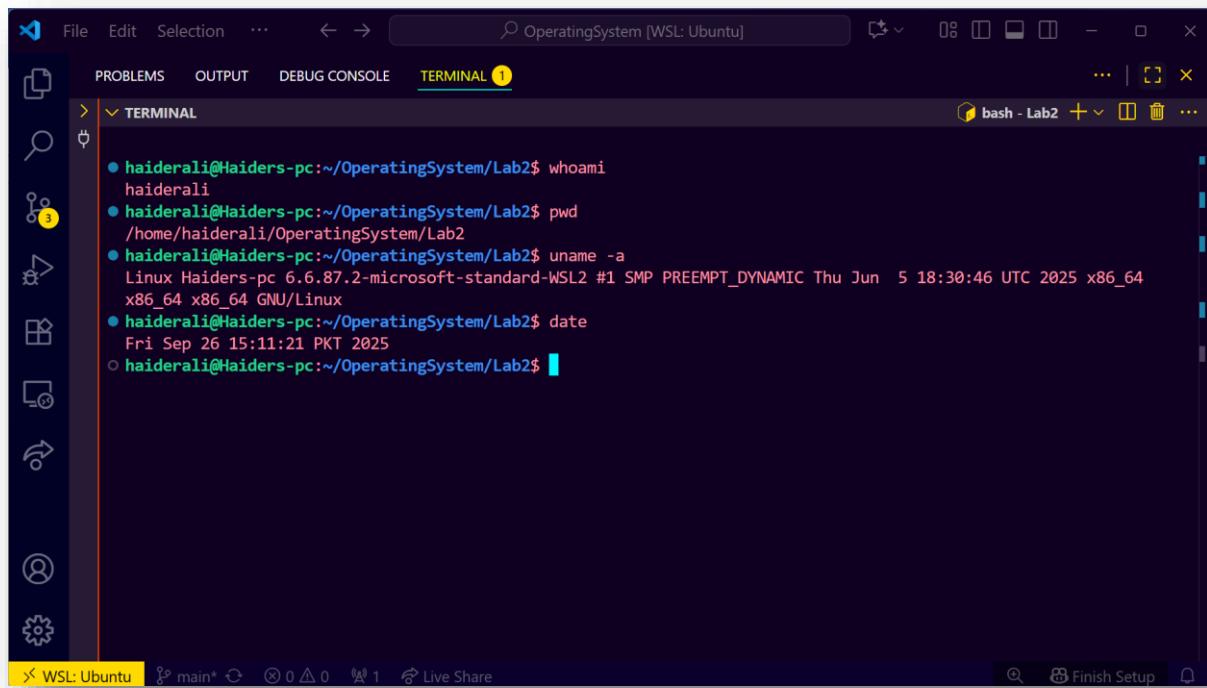
#### 1.1 Understanding the Linux Environment

- **Concepts to Cover:**

- What is Linux? Brief history and distributions
- Linux vs Windows: Key differences
- Understanding the shell (bash)
- WSL2 as a Linux environment

- **Hands-on Activity:**

```
# Students open WSL2 terminal and explore
whoami           # Check current user
pwd              # Print working directory
uname -a          # System information
date             # Current date and time
```

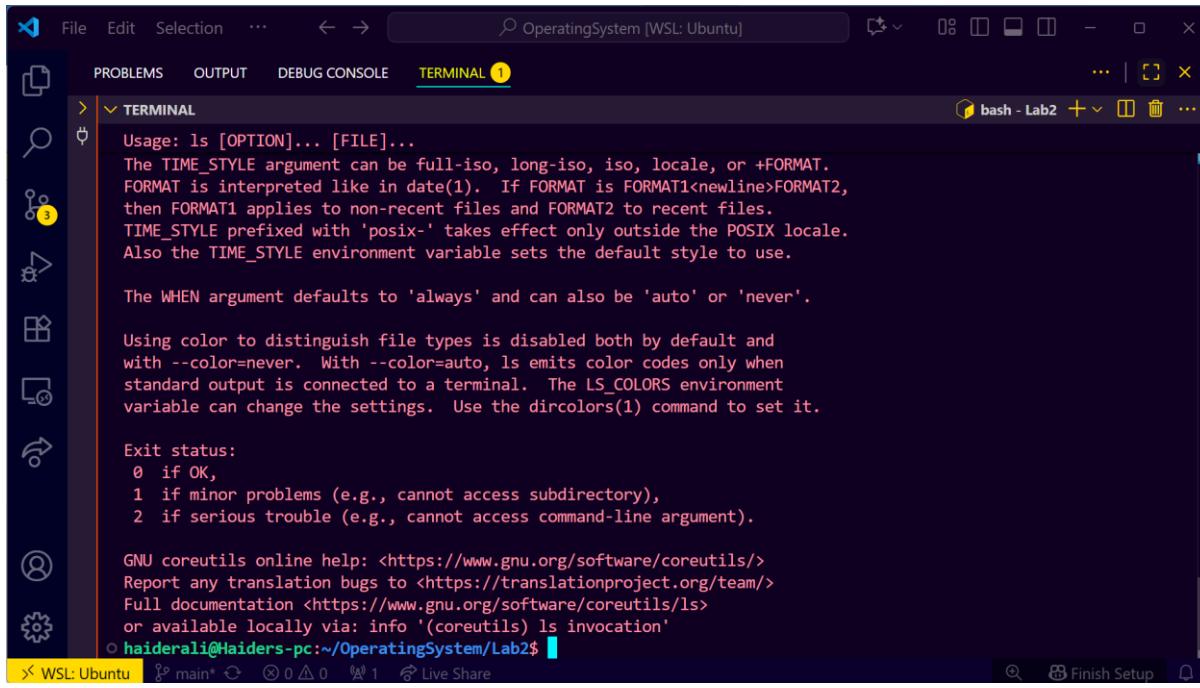


```
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ whoami
haiderali
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ pwd
/home/haiderali/OperatingSystem/Lab2
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ uname -a
Linux Haiders-pc 6.6.87.2-microsoft-standard-WSL2 #1 SMP PREEMPT_DYNAMIC Thu Jun 5 18:30:46 UTC 2025 x86_64
x86_64 x86_64 GNU/Linux
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ date
Fri Sep 26 15:11:21 PKT 2025
haiderali@Haiders-pc:~/OperatingSystem/Lab2$
```

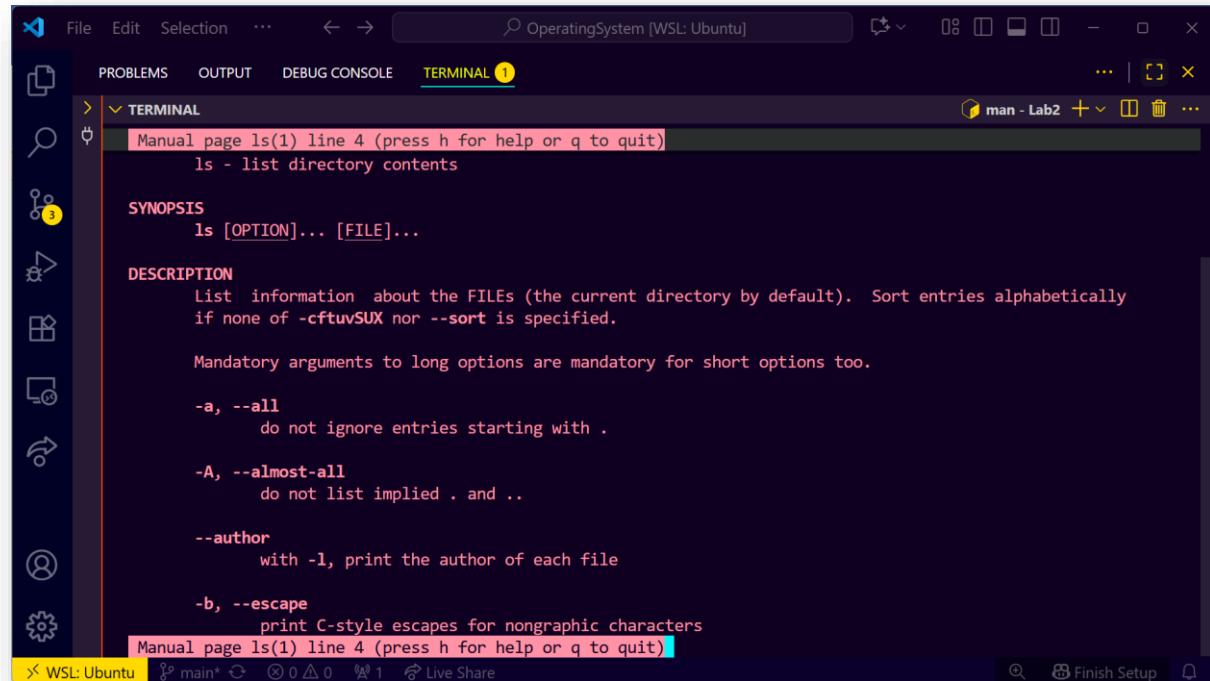
## 1.2 Getting Help in Linux

- Commands to demonstrate:

```
man ls          # Manual pages  
ls --help       # Built-in help  
which ls        # Location of commands  
type ls         # Command type information
```



```
OperatingSystem [WSL: Ubuntu]  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1  
TERMINAL  
Usage: ls [OPTION]... [FILE]...  
The TIME_STYLE argument can be full-iso, long-iso, iso, locale, or +FORMAT.  
FORMAT is interpreted like in date(1). If FORMAT is FORMAT1<newline>FORMAT2,  
then FORMAT1 applies to non-recent files and FORMAT2 to recent files.  
TIME_STYLE prefixed with 'posix-' takes effect only outside the POSIX locale.  
Also the TIME_STYLE environment variable sets the default style to use.  
  
The WHEN argument defaults to 'always' and can also be 'auto' or 'never'.  
  
Using color to distinguish file types is disabled both by default and  
with --color=never. With --color=auto, ls emits color codes only when  
standard output is connected to a terminal. The LS_COLORS environment  
variable can change the settings. Use the dircolors(1) command to set it.  
  
Exit status:  
0 if OK,  
1 if minor problems (e.g., cannot access subdirectory),  
2 if serious trouble (e.g., cannot access command-line argument).  
  
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>  
Report any translation bugs to <https://translationproject.org/team/>  
Full documentation <https://www.gnu.org/software/coreutils/ls>  
or available locally via: info '(coreutils) ls invocation'  
haiderali@Haiders-pc:~/OperatingSystem/Lab2$
```



```
OperatingSystem [WSL: Ubuntu]  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1  
TERMINAL  
Manual page ls(1) line 4 (press h for help or q to quit)  
ls - list directory contents  
  
SYNOPSIS  
ls [OPTION]... [FILE]...  
  
DESCRIPTION  
List information about the FILES (the current directory by default). Sort entries alphabetically  
if none of -cftuvSUX nor --sort is specified.  
  
Mandatory arguments to long options are mandatory for short options too.  
  
-a, --all  
      do not ignore entries starting with .  
  
-A, --almost-all  
      do not list implied . and ..  
  
--author  
      with -l, print the author of each file  
  
-b, --escape  
      print C-style escapes for nongraphic characters  
Manual page ls(1) line 4 (press h for help or q to quit)
```

A screenshot of a terminal window in the Visual Studio Code (VS Code) interface. The title bar indicates the window is titled 'OperatingSystem [WSL: Ubuntu]'. The terminal tab is active, showing the following command history:

```
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ which ls
/usr/bin/ls
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ type ls
ls is aliased to `ls --color=auto'
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ S
```

The left sidebar shows various icons for file operations like Open, Save, Find, and Settings. The bottom status bar displays 'WSL: Ubuntu' and other system information.

## Part 2: File System Navigation

### 2.1 Understanding Linux Directory Structure

- **Concepts to Cover:**

- Root directory (/)
- Important directories: /home, /usr, /etc, /var, /tmp
- Absolute vs relative paths
- Hidden files and directories

- **Demonstration:**

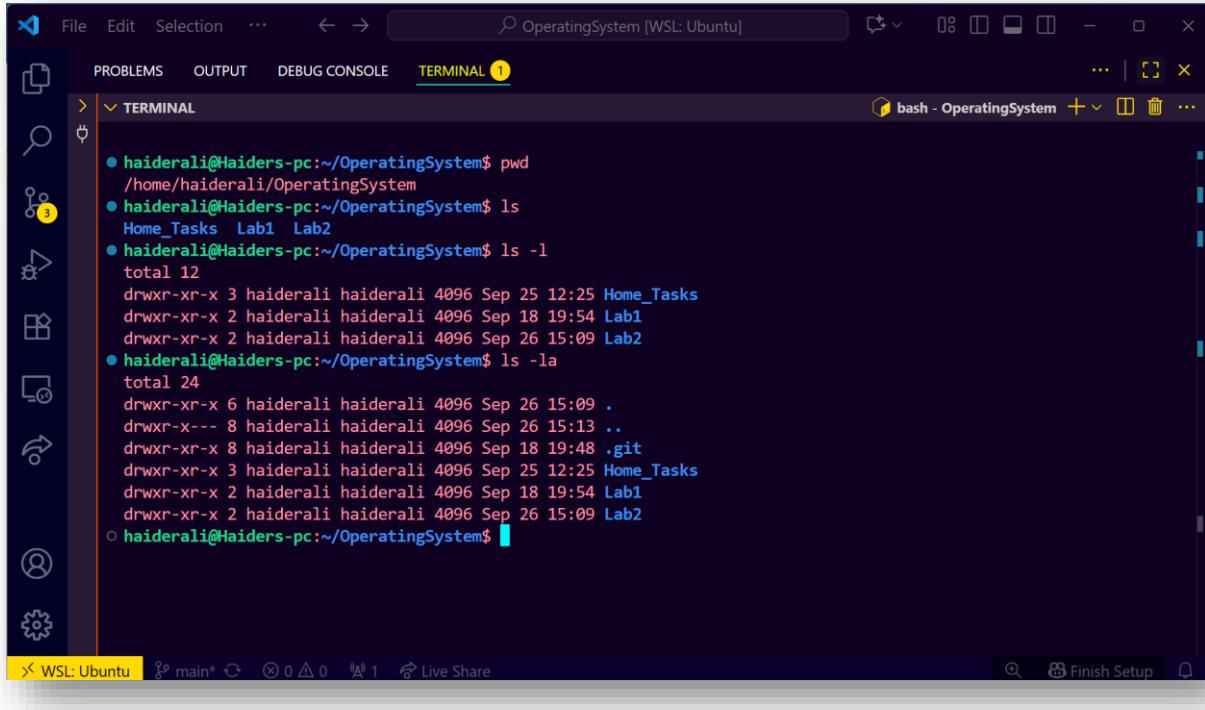
```
ls /          # Root directory contents
ls -la        # Long listing with hidden files
cd /home      # Change directory
cd ~         # Home directory shortcut
cd -         # Previous directory
```

```
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ ls /
bin          dev  init      lib64      mnt      root    sbin usr-is-merged sys  var
bin usr-is-merged etc  lib       lost+found opt     run   snap      tmp
boot        home lib usr-is-merged media    proc   sbin  srv      usr
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ ls -la
total 8
drwxr-xr-x 2 haiderali haiderali 4096 Sep 26 15:09 .
drwxr-xr-x 6 haiderali haiderali 4096 Sep 26 15:09 ..
haiderali@Haiders-pc:~/OperatingSystem/Lab2$ cd /home
haiderali@Haiders-pc:/home$ cd ~
haiderali@Haiders-pc:~$ cd -
/home
haiderali@Haiders-pc:/home$
```

## 2.2 Basic Navigation Commands (15 minutes)

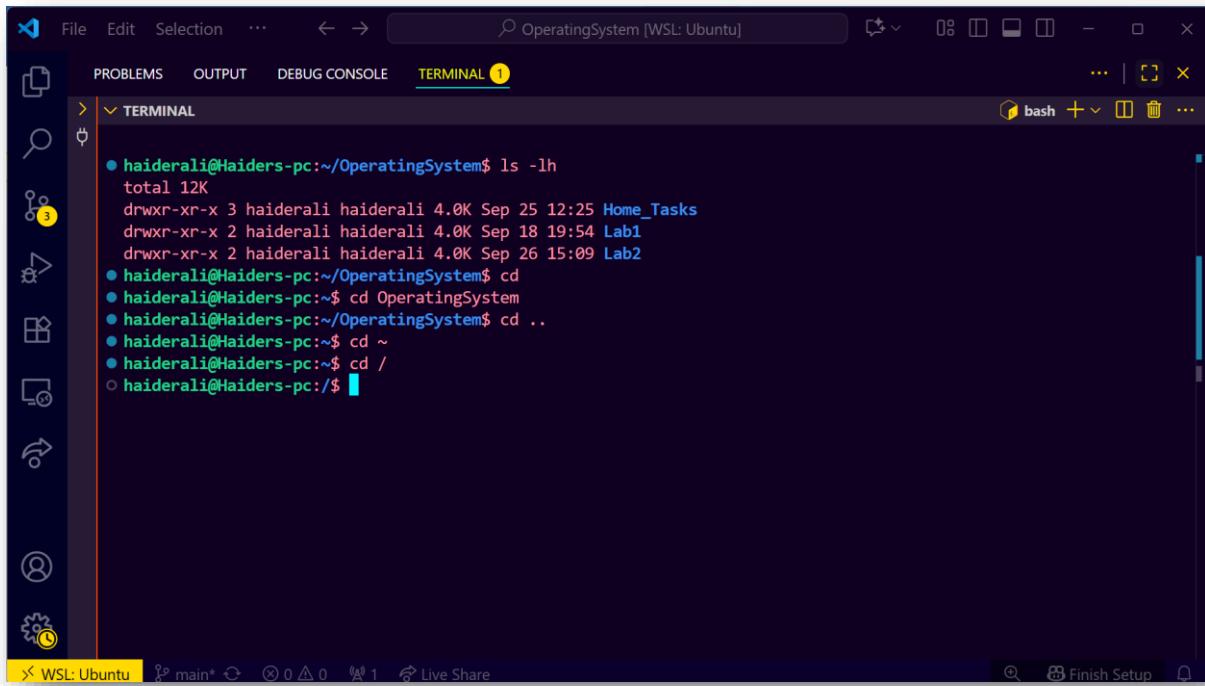
- Commands to practice:

```
pwd          # Present working directory
ls           # List directory contents
ls -l        # Long format
ls -la       # Include hidden files
ls -lh       # Human readable sizes
cd           # Change directory
cd ..        # Parent directory
cd ~         # Home directory
cd /         # Root directory
```



A screenshot of the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) Ubuntu environment. The terminal tab is active, displaying the following command-line session:

```
● haiderali@Haiders-pc:~/OperatingSystem$ pwd
/home/haiderali/OperatingSystem
● haiderali@Haiders-pc:~/OperatingSystem$ ls
Home_Tasks Lab1 Lab2
● haiderali@Haiders-pc:~/OperatingSystem$ ls -l
total 12
drwxr-xr-x 3 haiderali haiderali 4096 Sep 25 12:25 Home_Tasks
drwxr-xr-x 2 haiderali haiderali 4096 Sep 18 19:54 Lab1
drwxr-xr-x 2 haiderali haiderali 4096 Sep 26 15:09 Lab2
● haiderali@Haiders-pc:~/OperatingSystem$ ls -la
total 24
drwxr-xr-x 6 haiderali haiderali 4096 Sep 26 15:09 .
drwxr-xr-x 8 haiderali haiderali 4096 Sep 26 15:13 ..
drwxr-xr-x 8 haiderali haiderali 4096 Sep 18 19:48 .git
drwxr-xr-x 3 haiderali haiderali 4096 Sep 25 12:25 Home_Tasks
drwxr-xr-x 2 haiderali haiderali 4096 Sep 18 19:54 Lab1
drwxr-xr-x 2 haiderali haiderali 4096 Sep 26 15:09 Lab2
○ haiderali@Haiders-pc:~/OperatingSystem$
```



A screenshot of the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) Ubuntu environment. The terminal tab is active, displaying the following command-line session:

```
● haiderali@Haiders-pc:~/OperatingSystem$ ls -lh
total 12K
drwxr-xr-x 3 haiderali haiderali 4.0K Sep 25 12:25 Home_Tasks
drwxr-xr-x 2 haiderali haiderali 4.0K Sep 18 19:54 Lab1
drwxr-xr-x 2 haiderali haiderali 4.0K Sep 26 15:09 Lab2
● haiderali@Haiders-pc:~/OperatingSystem$ cd
● haiderali@Haiders-pc:~$ cd OperatingSystem
● haiderali@Haiders-pc:~/OperatingSystem$ cd ..
● haiderali@Haiders-pc:~$ cd ~
● haiderali@Haiders-pc:~$ cd /
○ haiderali@Haiders-pc:$
```

## Part 3: File and Directory Operations

### \*\*3.1 Creating and Managing Files/Directories

- **Commands to demonstrate:**

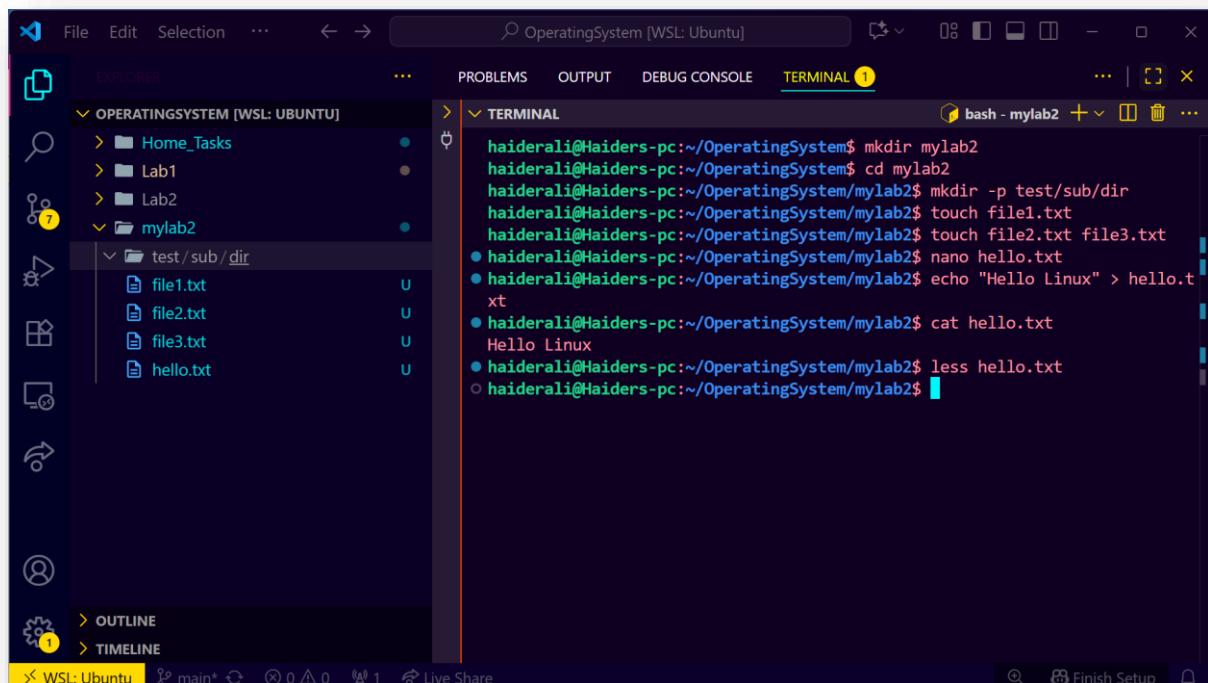
```
mkdir mylab2          # Create directory
mkdir -p test/sub/dir # Create nested directories
touch file1.txt        # Create empty file
touch file2.txt file3.txt # Multiple files

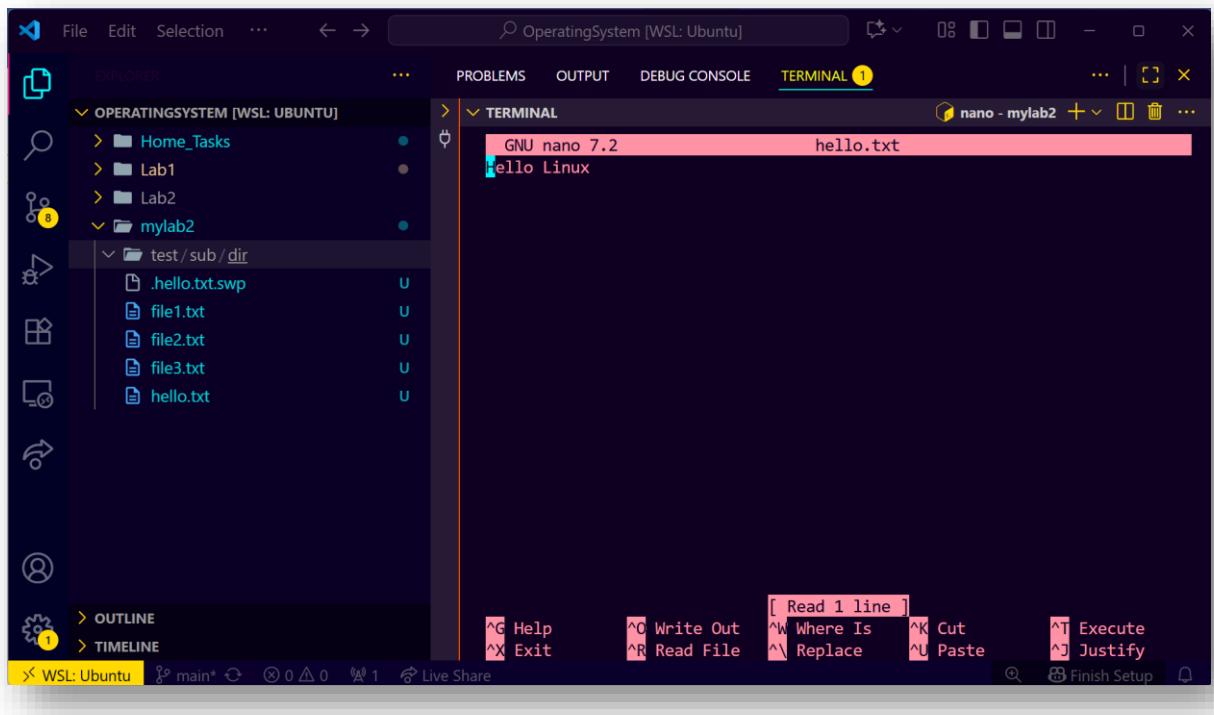
# Text editors introduction
nano hello.txt        # Simple text editor
# OR
echo "Hello Linux!" > hello.txt # Redirect output to file
```

- **File viewing commands:**

```
cat hello.txt          # Display file contents
less hello.txt         # Page through file
head hello.txt         # First 10 lines

tail hello.txt         # Last 10 lines
wc hello.txt           # Word count
```





The screenshot shows the VS Code interface with the title bar "OperatingSystem [WSL: Ubuntu]". The Explorer sidebar on the left shows a file structure under "OPERATINGSYSTEM [WSL: UBUNTU]" with folders like Home\_Tasks, Lab1, Lab2, and mylab2, and files like file1.txt, file2.txt, file3.txt, and hello.txt. The Terminal tab is active, displaying the following command-line session:

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ head hello.txt
Hello Linux
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ tail hello.txt
Hello Linux
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ wc hello.txt
1 2 12 hello.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$
```

This screenshot is similar to the one above, but the terminal output has been captured by the "less" command. The terminal window shows:

```
Hello Linux
hello.txt (END)
```

## 3.2 Copying, Moving, and Deleting

- Commands to practice:

```
cp hello.txt backup.txt      # Copy file
cp -r mylab2 mylab2_backup  # Copy directory recursively
mv backup.txt renamed.txt    # Move/rename file
rm renamed.txt               # Remove file
rm -r mylab2_backup          # Remove directory
rmdir empty_directory         # Remove empty directory
```

The screenshot shows the Visual Studio Code interface running in a Windows environment. The title bar indicates the active workspace is 'OperatingSystem [WSL: Ubuntu]'. The left sidebar features the Explorer, Problems, Output, Debug Console, and Terminal tabs. The Explorer tab is selected, displaying a file tree with folders like 'Home\_Tasks', 'Lab1', 'Lab2', and 'mylab2'. Inside 'mylab2', there is a 'test/sub\_dir' folder containing files 'file1.txt', 'file2.txt', 'file3.txt', 'hello.txt', and 'rename.txt'. The terminal tab is also selected, showing a bash session with the following command history:

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ cp hello.txt backup.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ cp -r mylab2 mylab2_backup
cp: cannot stat 'mylab2': No such file or directory
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ cd ..
haiderali@Haiders-pc:~/OperatingSystem$ cp -r mylab2 mylab2_backup
haiderali@Haiders-pc:~/OperatingSystem$ cd mylab2
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ mv backup.txt rename.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ rm rename.txt
```

This screenshot is nearly identical to the one above, showing the same VS Code interface and terminal session in WSL: Ubuntu. The command history in the terminal is identical, indicating no new interactions have occurred since the previous screenshot.

A screenshot of the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) environment. The title bar indicates the workspace is 'OperatingSystem [WSL: Ubuntu]'. The left sidebar shows a file tree with directories 'empty\_directory', 'Home\_Tasks', 'Lab1', 'Lab2', and 'mylab2'. Inside 'mylab2', there is a 'test/sub\_dir' folder containing files 'file1.txt', 'file2.txt', 'file3.txt', and 'hello.txt'. The right side of the interface features a terminal window titled 'bash - OperatingSystem'. The terminal displays the following command history:

```
haiderali@Haiders-pc:~/OperatingSystem$ mkdir empty_directory
haiderali@Haiders-pc:~/OperatingSystem$ rmdir empty_directory
haiderali@Haiders-pc:~/OperatingSystem$
```

A screenshot of the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) environment. The title bar indicates the workspace is 'OperatingSystem [WSL: UBUNTU]'. The left sidebar shows a file tree with directories 'empty\_directory', 'Home\_Tasks', 'Lab1', 'Lab2', and 'mylab2'. Inside 'mylab2', there is a 'test/sub\_dir' folder containing files 'file1.txt', 'file2.txt', 'file3.txt', and 'hello.txt'. The right side of the interface features a terminal window titled 'bash - OperatingSystem'. The terminal displays the following command history:

```
haiderali@Haiders-pc:~/OperatingSystem$ mkdir empty_directory
haiderali@Haiders-pc:~/OperatingSystem$ rmdir empty_directory
haiderali@Haiders-pc:~/OperatingSystem$
```

**Hands-on Exercise:** Students create a directory structure, add files, and practice file operations.

# Part 4: File Permissions and Ownership

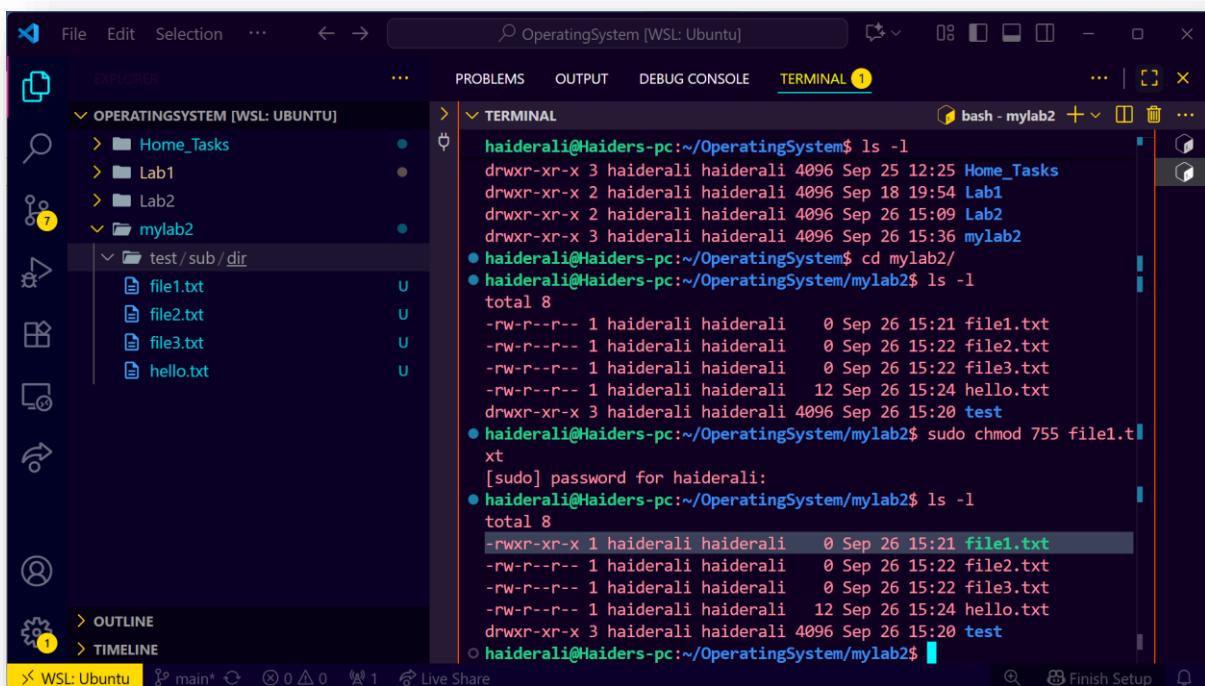
## 4.1 Understanding File Permissions

- Concepts to Cover:

- Permission types: read (r), write (w), execute (x)
- Permission groups: user (u), group (g), others (o)
- Numeric notation: 755, 644, etc.

- Commands to demonstrate:

```
ls -l          # View permissions  
chmod 755 file.txt    # Change permissions (numeric)  
chmod u+x file.txt    # Add execute permission for user  
chmod g-w file.txt    # Remove write permission for group  
chown user:group file.txt # Change ownership (if applicable)
```



A screenshot of the Visual Studio Code interface running in a Windows Subsystem for Linux (WSL) environment. The title bar indicates the workspace is 'OperatingSystem [WSL: Ubuntu]'. The left sidebar shows a file tree with folders 'Home\_Tasks', 'Lab1', 'Lab2', and 'mylab2', which contains a 'test/sub\_dir' folder with files 'file1.txt', 'file2.txt', 'file3.txt', and 'hello.txt'. The right side features a terminal window titled 'bash - mylab2'. The terminal output shows the user navigating to the directory, listing files, changing permissions, and testing them.

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ chmod u+x file2.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
haiderali@Haiders-pc:~/OperatingSystem/mylab2$
```

A second screenshot of the Visual Studio Code interface in the same WSL workspace. The terminal output is identical to the first screenshot, demonstrating the execution of the command 'chmod u+x file2.txt' and its effect on the file permissions listed in the 'ls -l' command.

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ chmod g+w file3.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ chmod u-w file3.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-r--r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
haiderali@Haiders-pc:~/OperatingSystem/mylab2$
```

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ chown haiderali:haiderali hello.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ ls -l
total 8
-rwxr-xr-x 1 haiderali haiderali 0 Sep 26 15:21 file1.txt
-rw-r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file2.txt
-r--r--r-- 1 haiderali haiderali 0 Sep 26 15:22 file3.txt
-rw-r--r-- 1 haiderali haiderali 12 Sep 26 15:24 hello.txt
drwxr-xr-x 3 haiderali haiderali 4096 Sep 26 15:20 test
```

## Part 5: Text Processing and Utilities

### 5.1 Essential Text Commands

- Commands to demonstrate:

```
grep "pattern" file.txt      # Search for patterns
grep -i "pattern" file.txt  # Case-insensitive search
grep -n "pattern" file.txt  # Show line numbers
```

```
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ grep "pattern" file1.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ grep -i "pattern" file1.txt
haiderali@Haiders-pc:~/OperatingSystem/mylab2$ grep -n "pattern" file1.txt
```

## 5.2 Pipes and Redirection

- Concepts and commands:

```
ls -l | grep ".txt"           # Pipe output
cat file1.txt file2.txt > combined.txt # Redirect output
echo "new line" >> file.txt # Append to file
sort file.txt | uniq          # Chain commands
```

---

# Part 6: Introduction to Processes

## 6.1 Understanding Processes

- Concepts to Cover:
  - What is a process?
  - Process ID (PID)
  - Parent-child relationships
  - Process states
- Commands to demonstrate:

```
ps                      # Show current processes
ps aux                  # Detailed process list
ps -ef                  # Full format listing
pstree                 # Process tree
top                     # Real-time process viewer
htop                   # Enhanced process viewer (if available)
kill PID                # Terminate process by PID
killall process_name    # Kill processes by name
pkill pattern           # Kill processes matching pattern
```