Operating Systems – COC 3071L

SE 5th A – Fall 2025 Part

1: File and Directory Operations

1. Create the following directory structure in your home directory:

```
Lab_3/
├── docs/
│   └── drafts/
├── data/
│   ├── raw/
│   └── processed/
└── scripts/
```

2. Inside `docs/`:
   - Create three files: `intro.txt`, `notes.txt`, `summary.txt`.
   - Add at least **two lines of text** into each using `echo >>`.
   - Copy `summary.txt` into the `drafts/` folder using `cp` command.
3. Inside `data/raw/`:
   - Create two files: `raw1.txt`, `raw2.txt`.
   - Append the **current date** into `raw1.txt` using the `date` command.
   - Move `raw2.txt` into `processed/` using `mv`. The syntax is:

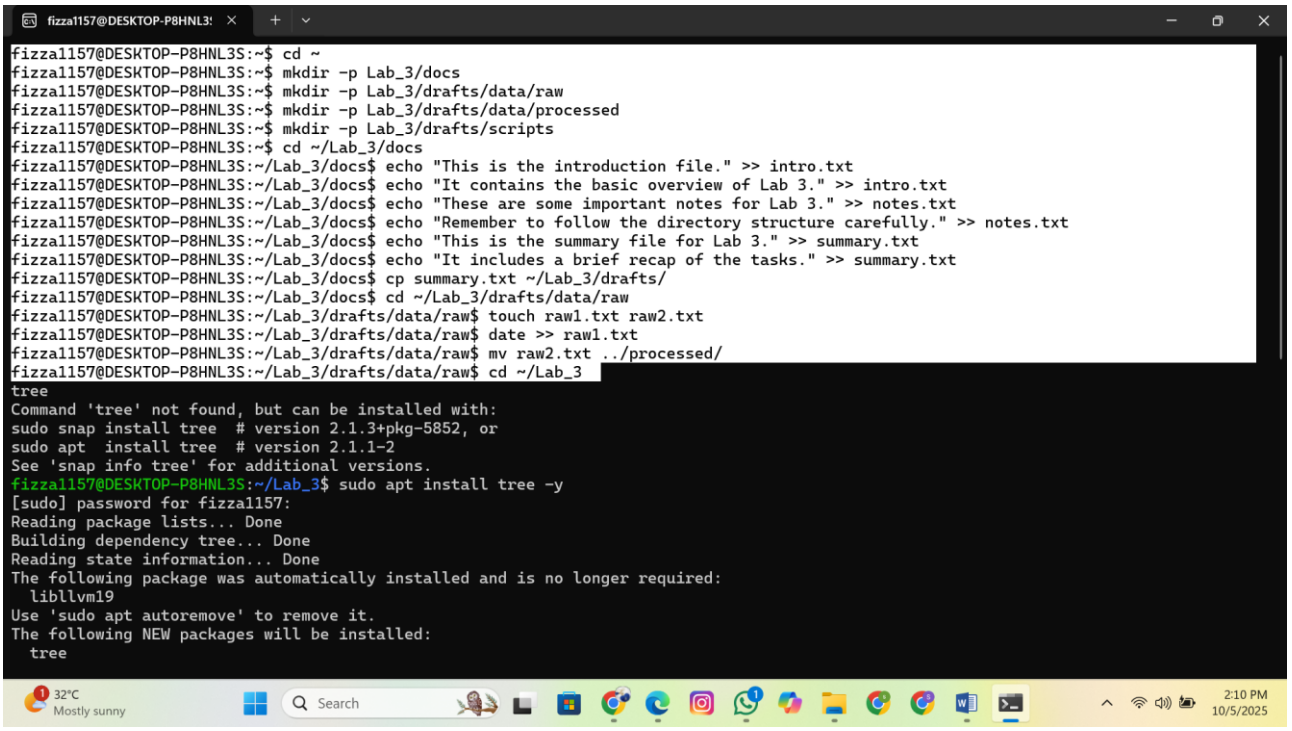
```
mv source destination
```

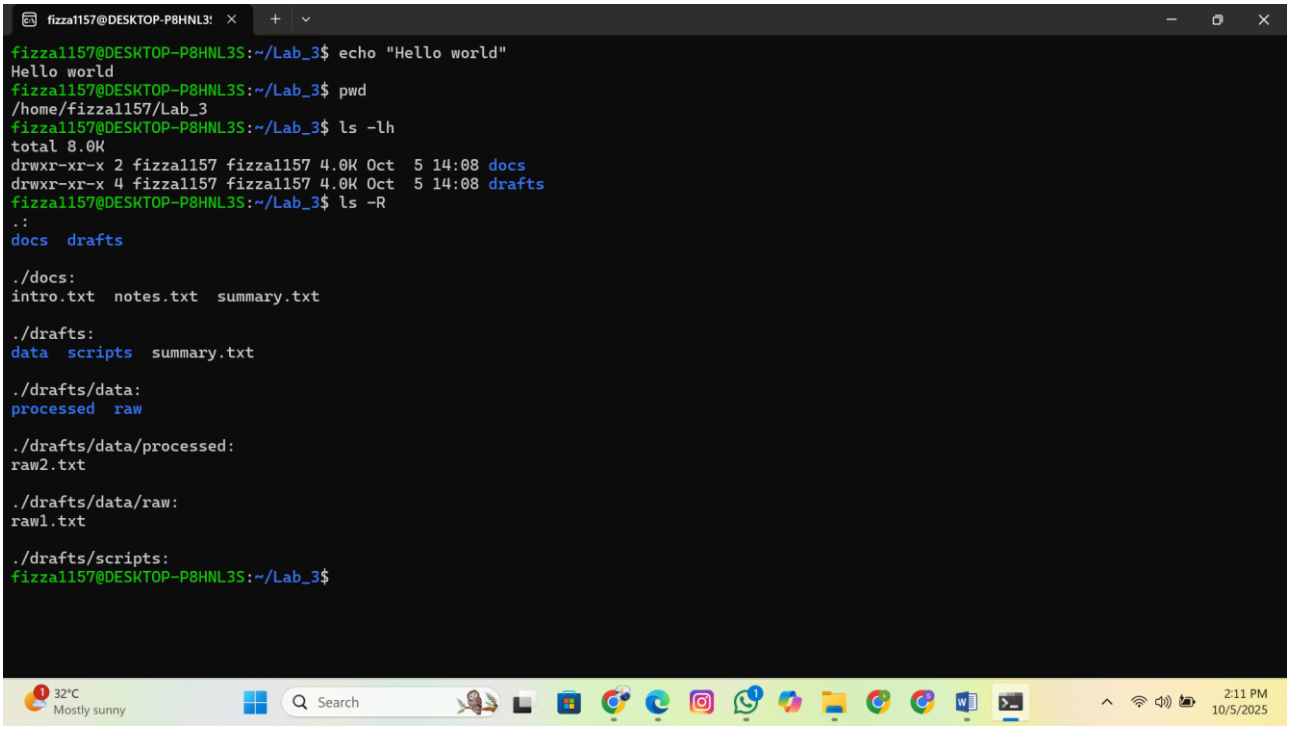
4. Inside `scripts/`:
   - Create a script named `hello.sh` with the following content:

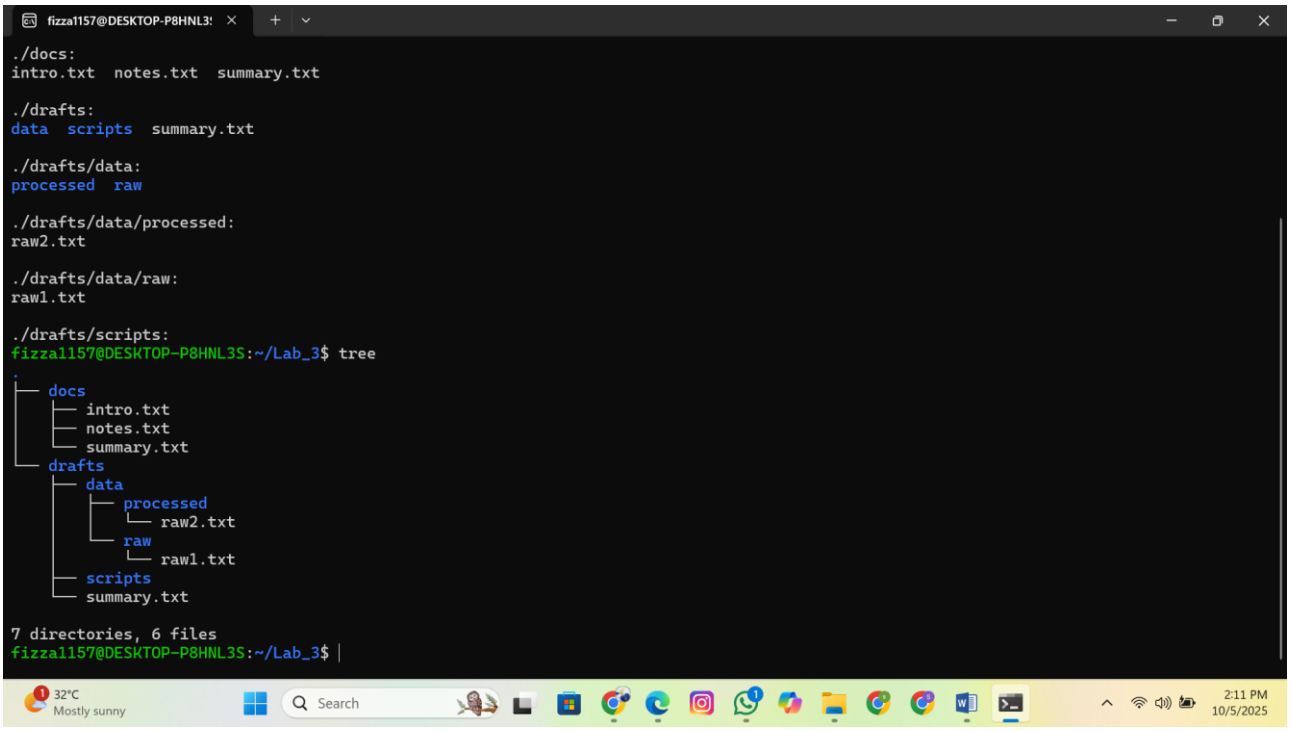
```
echo "Hello World"
pwd
ls -lh
```

   - Later, you will make it executable (in Part 3).
5. Display the directory structure recursively and take a screenshot:

```
ls -R
```


fizza1157@DESKTOP-P8HNL3S:~$ cd ~
fizza1157@DESKTOP-P8HNL3S:~$ mkdir -p Lab_3/docs
fizza1157@DESKTOP-P8HNL3S:~$ mkdir -p Lab_3/drafts/data/raw
fizza1157@DESKTOP-P8HNL3S:~$ mkdir -p Lab_3/drafts/data/processed
fizza1157@DESKTOP-P8HNL3S:~$ mkdir -p Lab_3/drafts/scripts
fizza1157@DESKTOP-P8HNL3S:~$ cd ~/Lab_3/docs
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "This is the introduction file." >> intro.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "It contains the basic overview of Lab 3." >> intro.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "These are some important notes for Lab 3." >> notes.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "Remember to follow the directory structure carefully." >> notes.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "This is the summary file for Lab 3." >> summary.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ echo "It includes a brief recap of the tasks." >> summary.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ cp summary.txt ~/Lab_3/drafts/
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/docs$ cd ~/Lab_3/drafts/data/raw
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/drafts/data/raw$ touch raw1.txt raw2.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/drafts/data/raw$ date >> raw1.txt
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/drafts/data/raw$ mv raw2.txt ../processed/
fizza1157@DESKTOP-P8HNL3S:~/Lab_3/drafts/data/raw$ cd ~/Lab_3
tree
Command 'tree' not found, but can be installed with:
sudo snap install tree # version 2.1.3+pkg-5852, or
sudo apt install tree # version 2.1.1-2
See 'snap info tree' for additional versions.
fizza1157@DESKTOP-P8HNL3S:~/Lab_3$ sudo apt install tree -y
[sudo] password for fizza1157:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
libllvm19
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
tree
32°C
Mostly sunny
Search
2:10 PM
10/5/2025


fizza1157@DESKTOP-P8HNL3S:~/Lab_3$ echo "Hello world"
Hello world
fizza1157@DESKTOP-P8HNL3S:~/Lab_3$ pwd
/home/fizza1157/Lab_3
fizza1157@DESKTOP-P8HNL3S:~/Lab_3$ ls -lh
total 8.0K
drwxr-xr-x 2 fizza1157 fizza1157 4.0K Oct 5 14:08 docs
drwxr-xr-x 4 fizza1157 fizza1157 4.0K Oct 5 14:08 drafts
fizza1157@DESKTOP-P8HNL3S:~/Lab_3$ ls -R
.:
docs drafts
./docs:
intro.txt notes.txt summary.txt
./drafts:
data scripts summary.txt
./drafts/data:
processed raw
./drafts/data/processed:
raw2.txt
./drafts/data/raw:
raw1.txt
./drafts/scripts:
fizza1157@DESKTOP-P8HNL3S:~/Lab_3$
32°C
Mostly sunny
Search
2:11 PM
10/5/2025

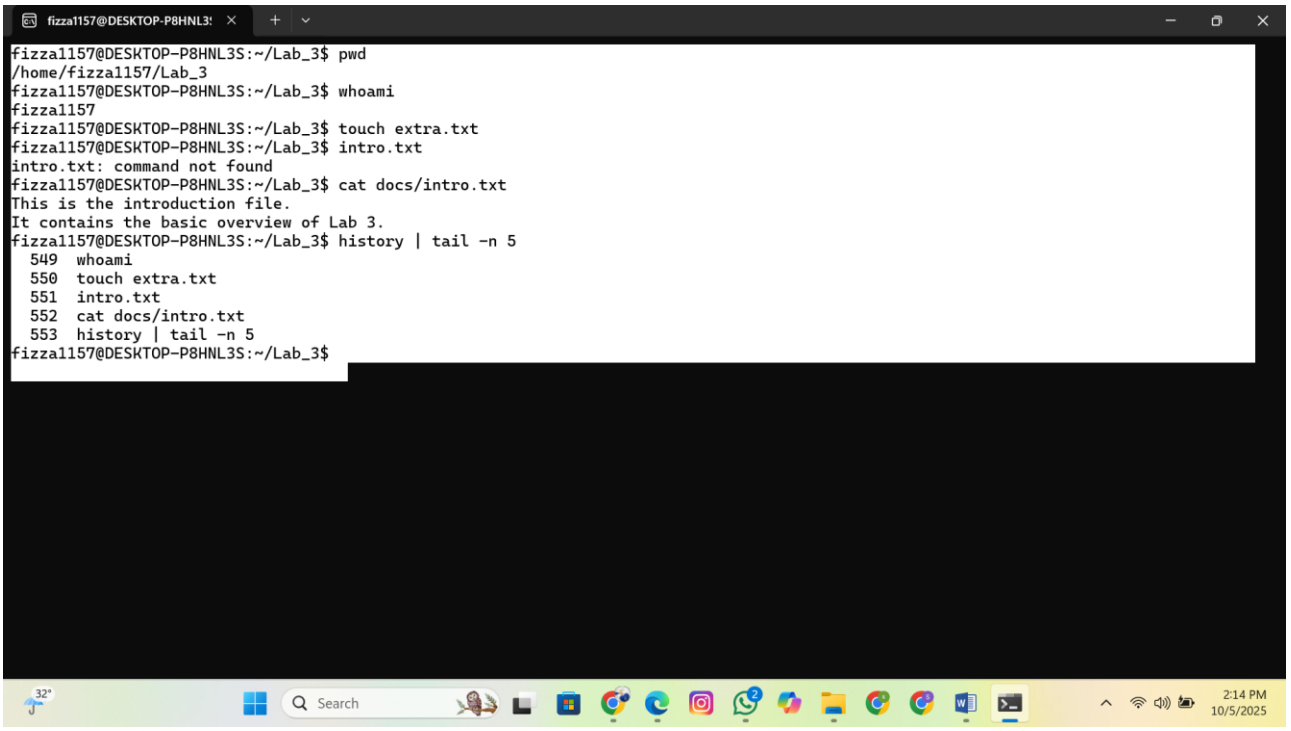


Part 2: Practice with Basic Linux Commands

Run the following commands inside `Lab_3/` and note their outputs:

- `pwd` → Show current working directory.
- `whoami` → Display the current logged-in user.
- `touch extra.txt` → Create an empty file. `cat`
- `intro.txt` → Display file contents. `rm extra.txt` →
- Delete a file.
- `history | tail -n 5` → Show your last 5 executed commands. `clear` → Clear
- the terminal.

Take screenshots of commands and outputs.



Part 3: File Permissions and Ownership

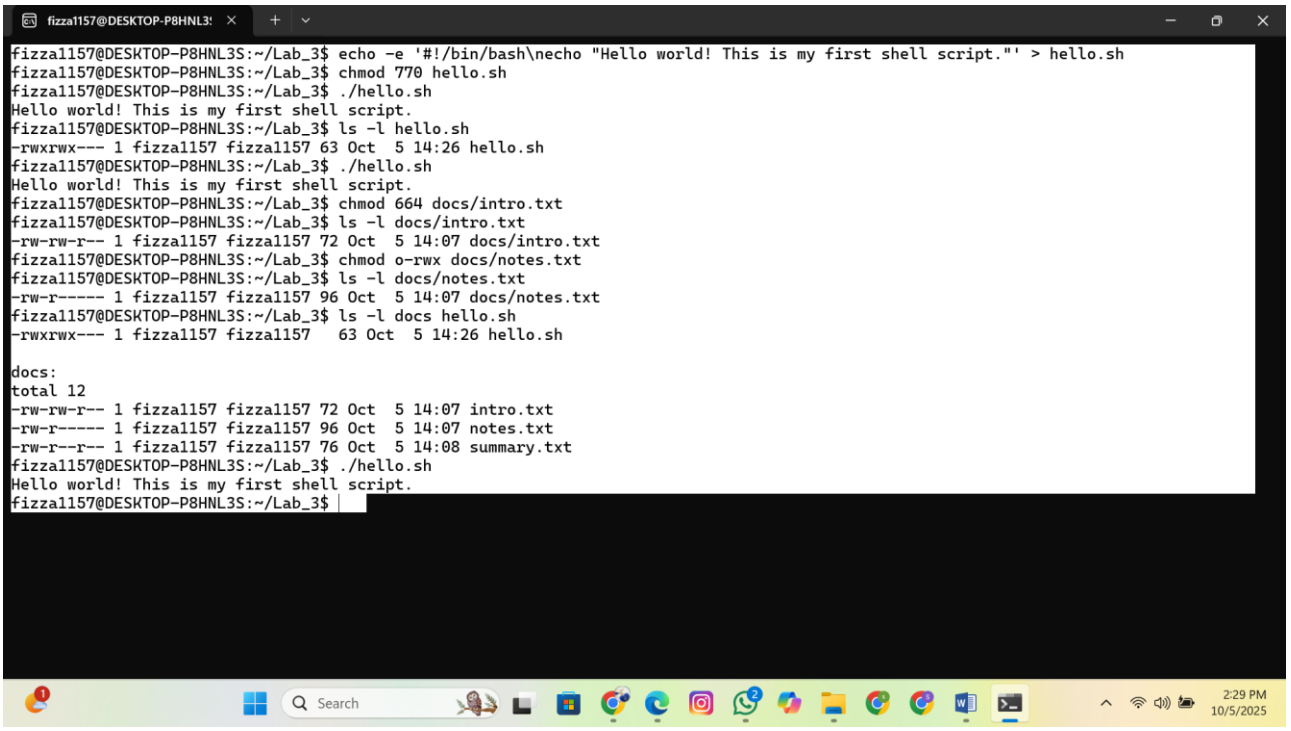
1. Change the permissions of `hello.sh` so that:
   - Owner → Read, Write & Execute
   - Group → Read, Write & Execute
   - Others → No permissions
   - Run the script using:

     ```
     ./hello.sh
     ```

     Take a screenshot of its output.
2. Change the permissions of `intro.txt` using **numeric notation** so that:
   - Owner → Read & Write
   - Group → Read & Write
   - Others → Read only
3. Change the permissions of `notes.txt` using **symbolic notation** so that `others` don't have any permission on it.
4. Verify all changes with:

   ```
   ls -l
   ```

   Take a screenshot of the output.

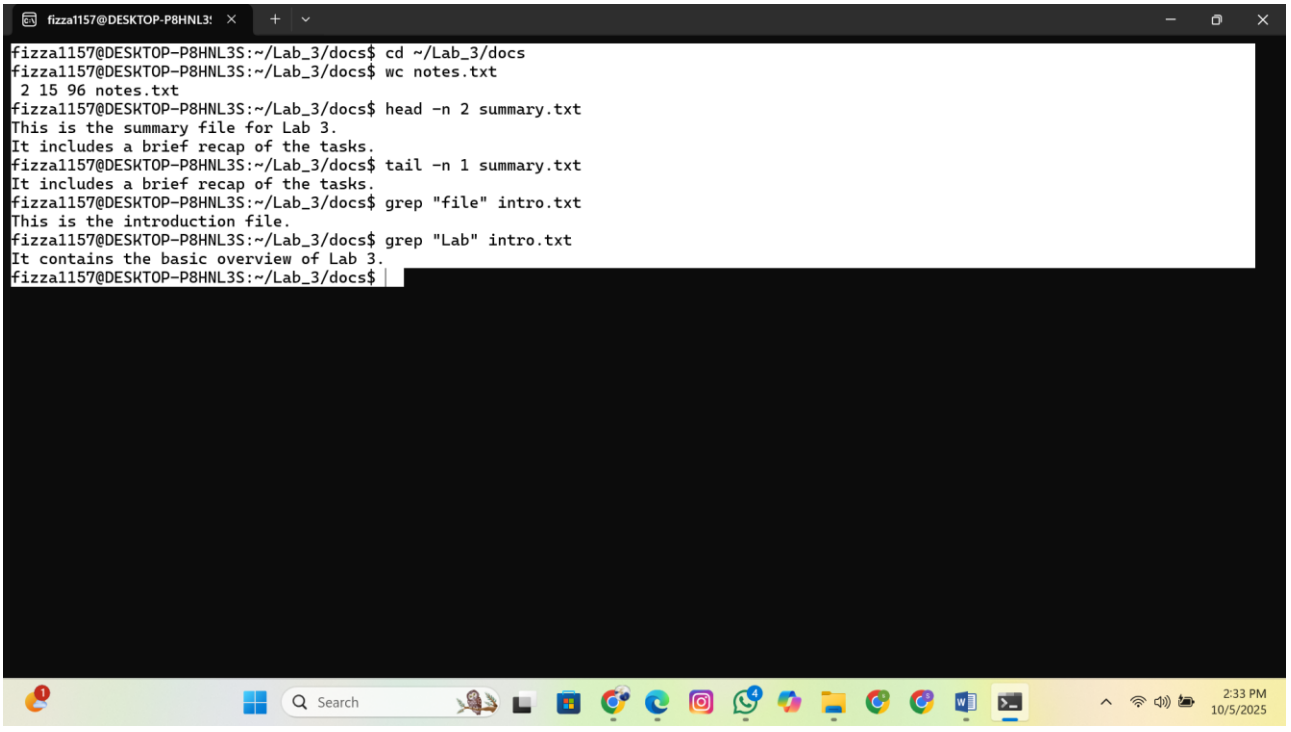


Part 4: Reading & Searching Files

Inside `docs/` :

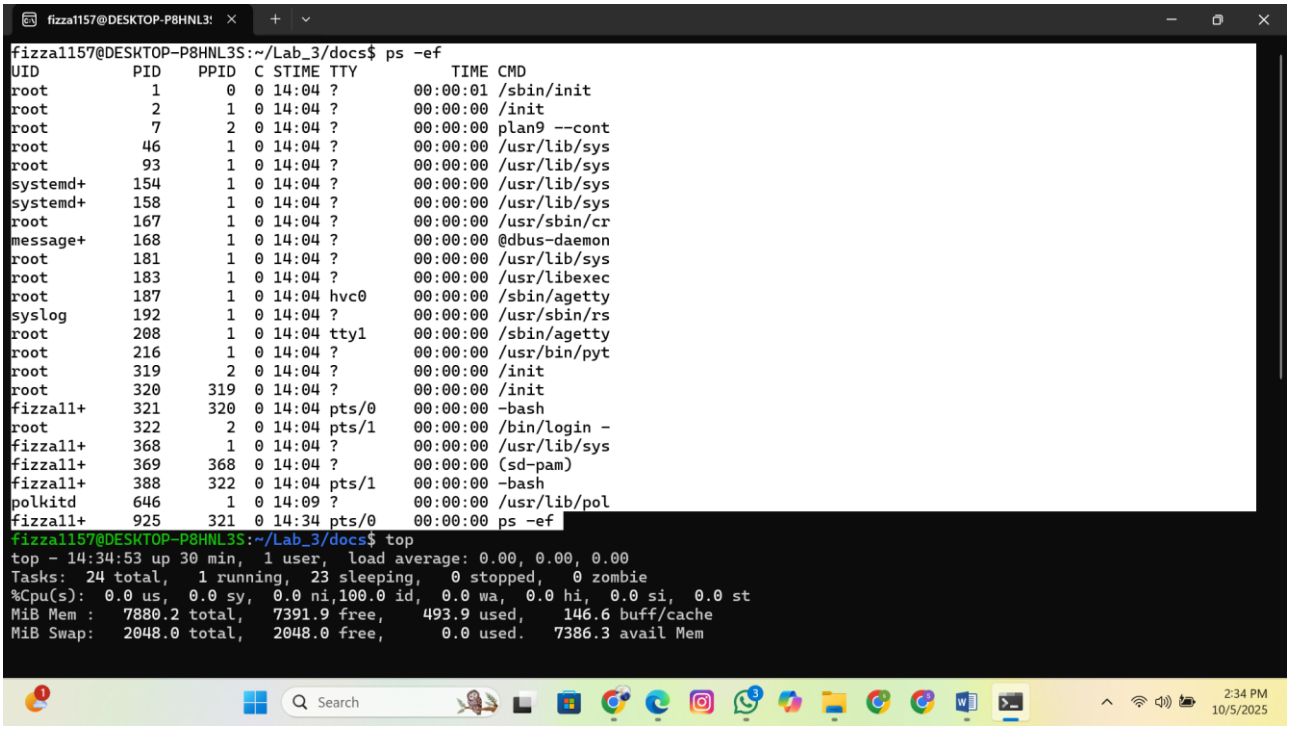
1. Count the number of lines, words, and characters in `notes.txt` using `wc` .
2. Show only the **first 2 lines** of `summary.txt` using `head -n 2` .
3. Show the **last line** of `summary.txt` using `tail -n 1` .
4. Search for a keyword (of your choice) in `intro.txt` using `grep` .

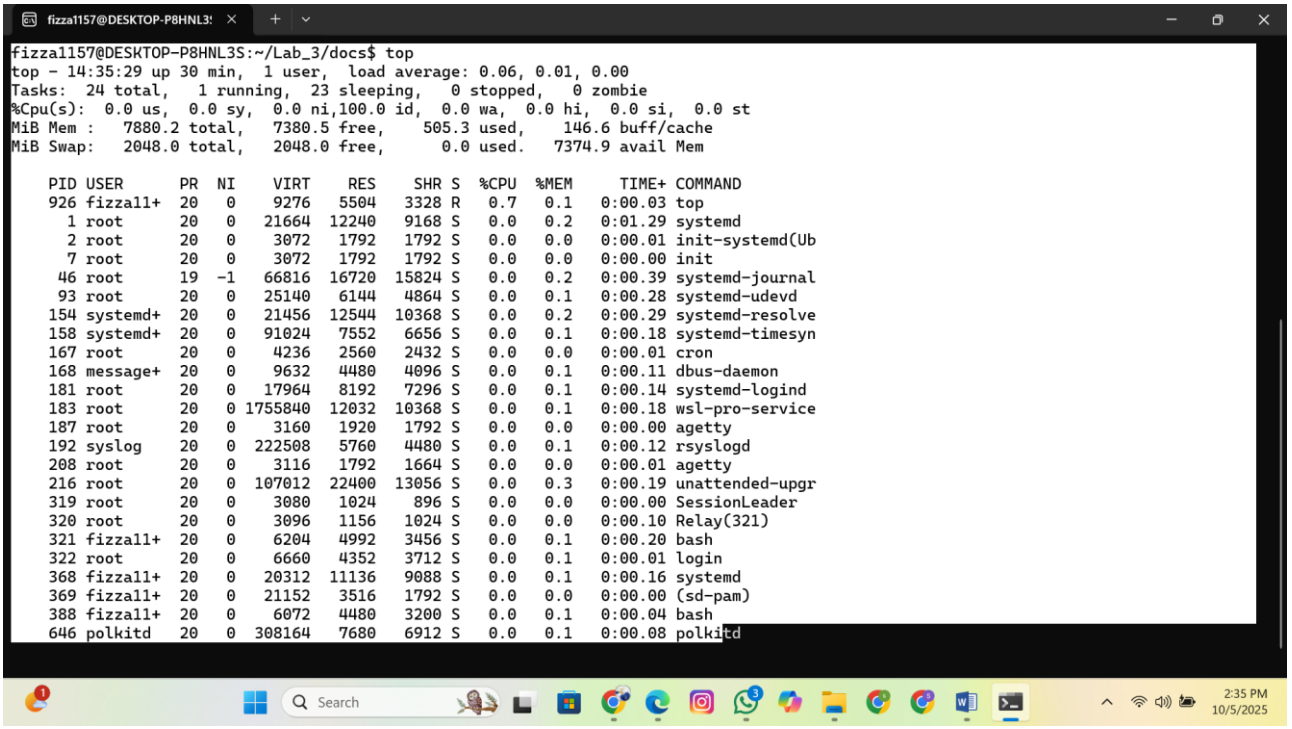
Take screenshots.



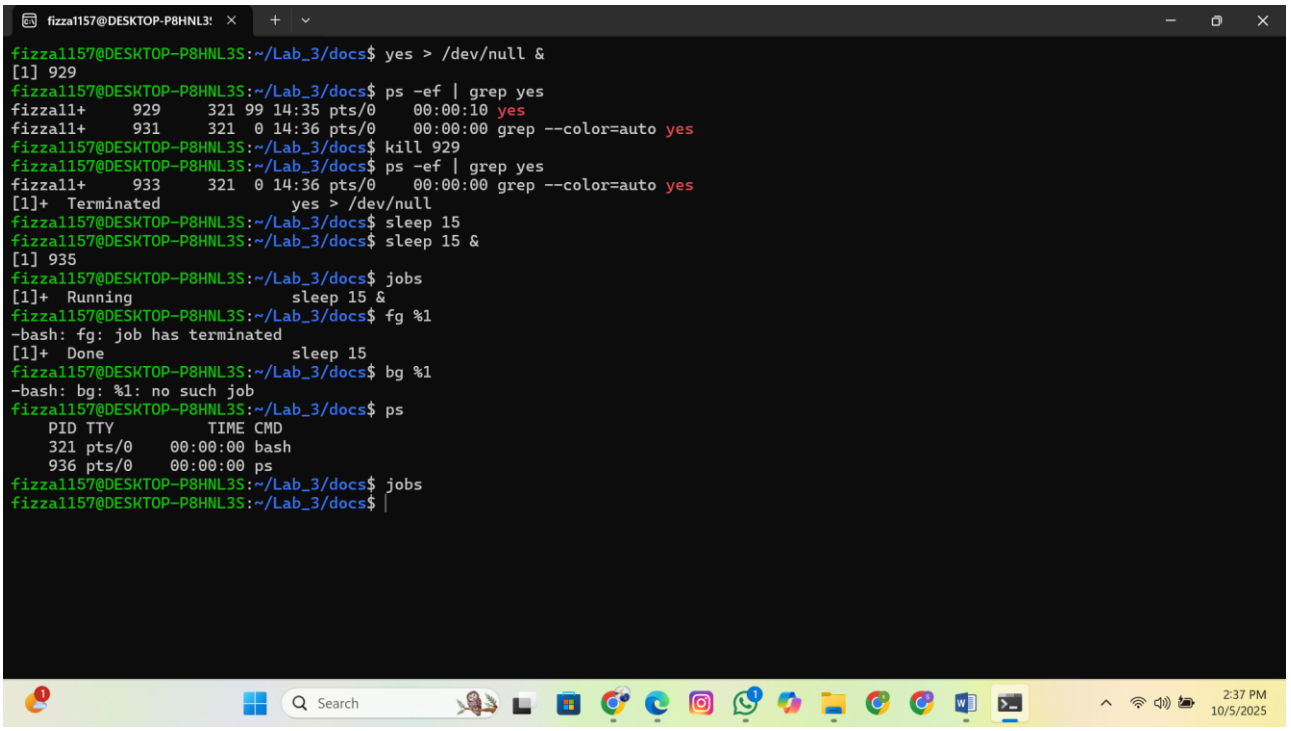
Part 5: Linux Process Commands

1. **Exploring Processes**
   - Use `ps -ef` and identify **3 processes** running on your system. Note their **PID, PPID, and command**.
   - Run `top` for 20–30 seconds. Write down:
     - Which process is consuming the most CPU.
     - Which process is consuming the most memory.





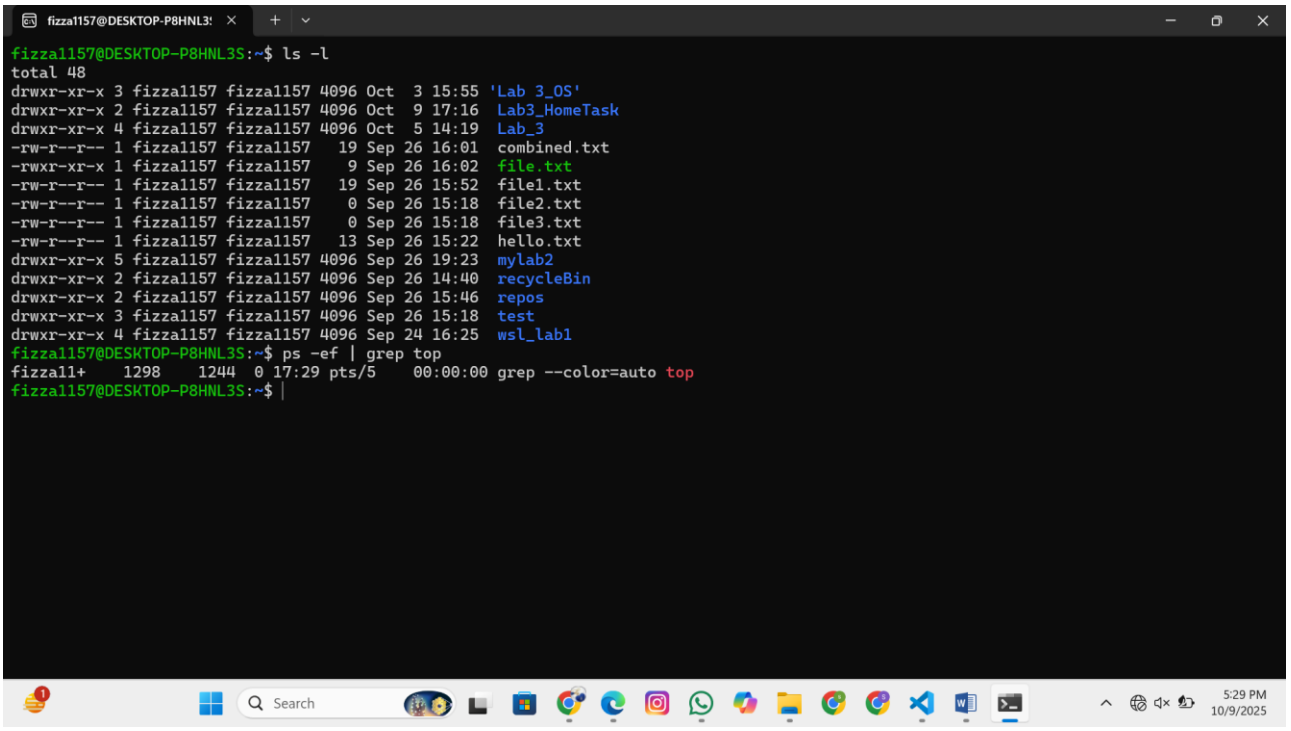
2. **Practice with Infinite Process**
   - Start:
     ```
     yes > /dev/null &
     ```
   - Locate its PID using `ps -ef | grep yes`.
   - Kill it using `kill <PID>` and verify using `ps`.
3. **Foreground & Background Jobs**
   - Run `sleep 60` in **foreground** and terminate it with **Ctrl + C**.
   - Run `sleep 60 &` in **background**, bring it to foreground with `fg`, stop with **Ctrl + Z**, then resume in background using `bg`.



Part 6: C Programs on Processes

Program 1 – Exec with `top`

- Modify the exec program so that the child runs **`top`** instead of `ls -l`.
- Run the program.
- In another terminal, use `ps -ef | grep top` (or run `top`) to find the child's PID.
- Use the child's process ID to kill it manually.



Program 2 – Incomplete Program

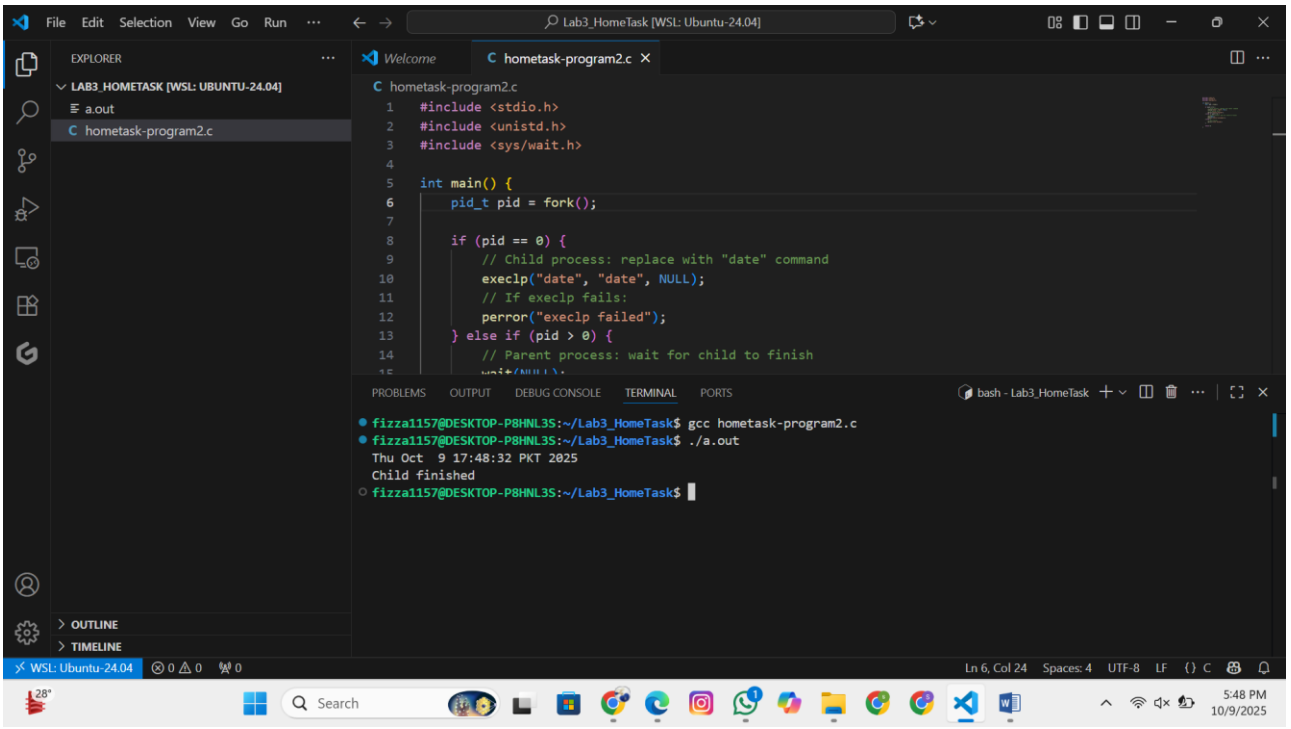
```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // TODO: Replace this child process with the "date" command using execlp
        // Hint: execlp("date", "date", NULL);
    } else {
        // TODO: Make parent wait for child before printing "Child finished"
    }

    return 0;
}
```

Task: Complete the missing parts, run the program, and take a screenshot of the output.



Submission Guidelines

- Submit a **single PDF file** including:
  - Screenshots of all said commands & outputs.
  - Modified & completed C program code and outputs.
- **Deadline:** 9th October, 2025, 11:59 PM.
