National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Akasha Fatima

Reg. number:

23-NTU-CS-FL-1132

Semester:

5th - A

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
```

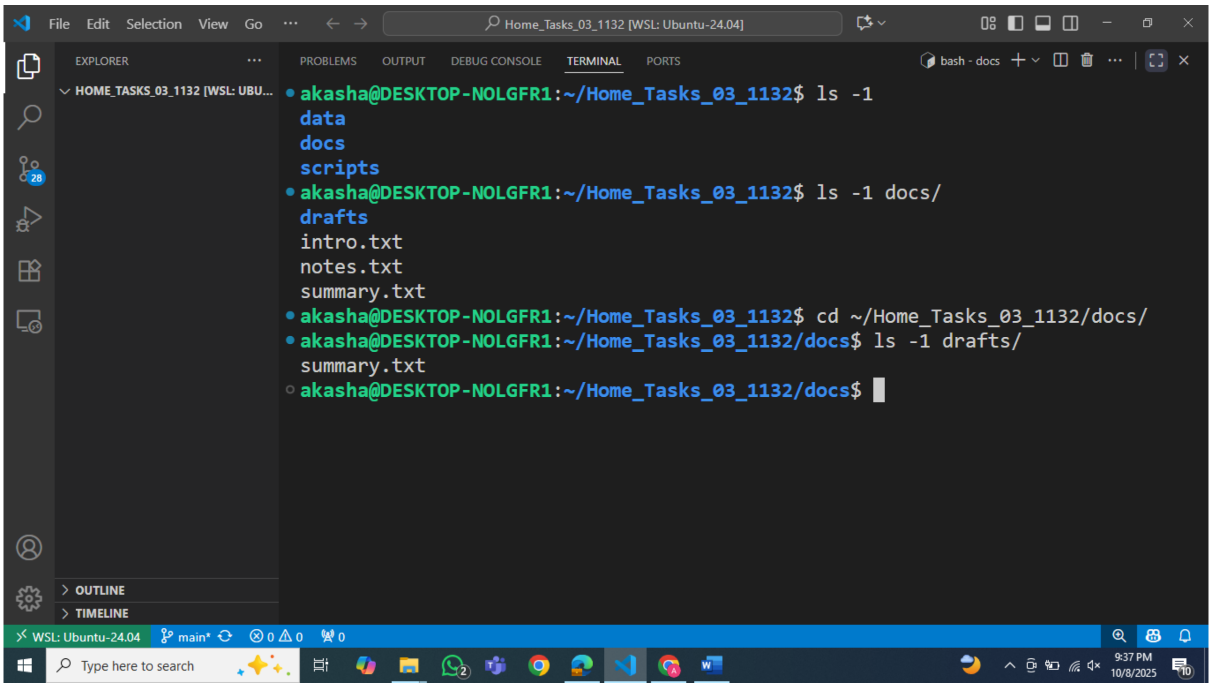
4. Inside `scripts/`:
   - Create a script named `hello.sh` with the following content:

   ```
   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:

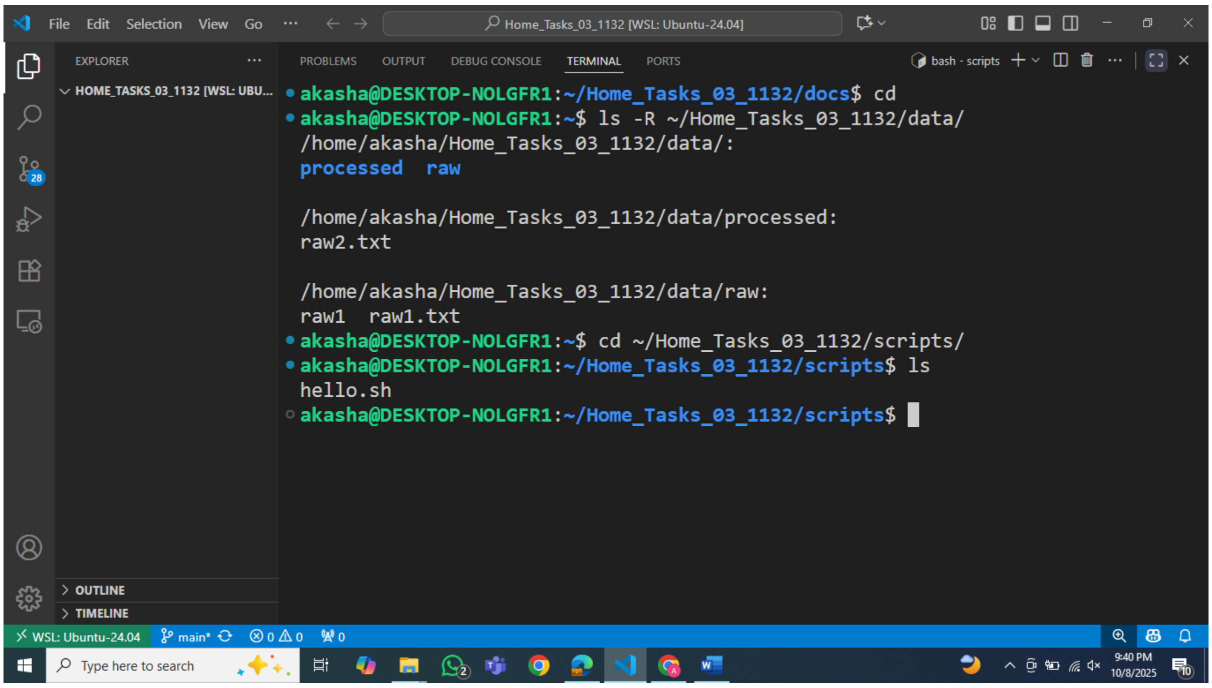
```
ls -R
```

Part 1 Output:



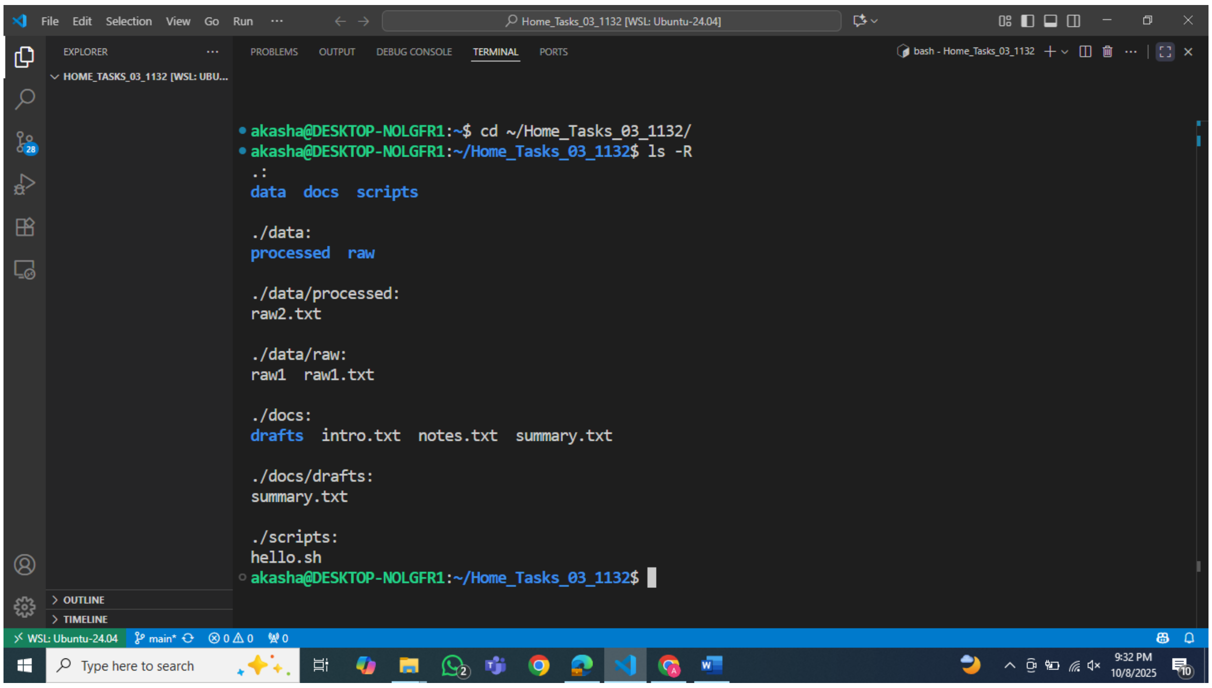
```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ls -1
data
docs
scripts
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ls -1 docs/
drafts
intro.txt
notes.txt
summary.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ cd ~/Home_Tasks_03_1132/docs/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ ls -1 drafts/
summary.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
```

Part 2 and 3 Output:



```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ cd
akasha@DESKTOP-NOLGFR1:~$ ls -R ~/Home_Tasks_03_1132/data/
/home/akasha/Home_Tasks_03_1132/data/:
processed  raw

/home/akasha/Home_Tasks_03_1132/data/processed:
raw2.txt

/home/akasha/Home_Tasks_03_1132/data/raw:
raw1  raw1.txt
akasha@DESKTOP-NOLGFR1:~$ cd ~/Home_Tasks_03_1132/scripts/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$ ls
hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$
```

3. Overall Output:



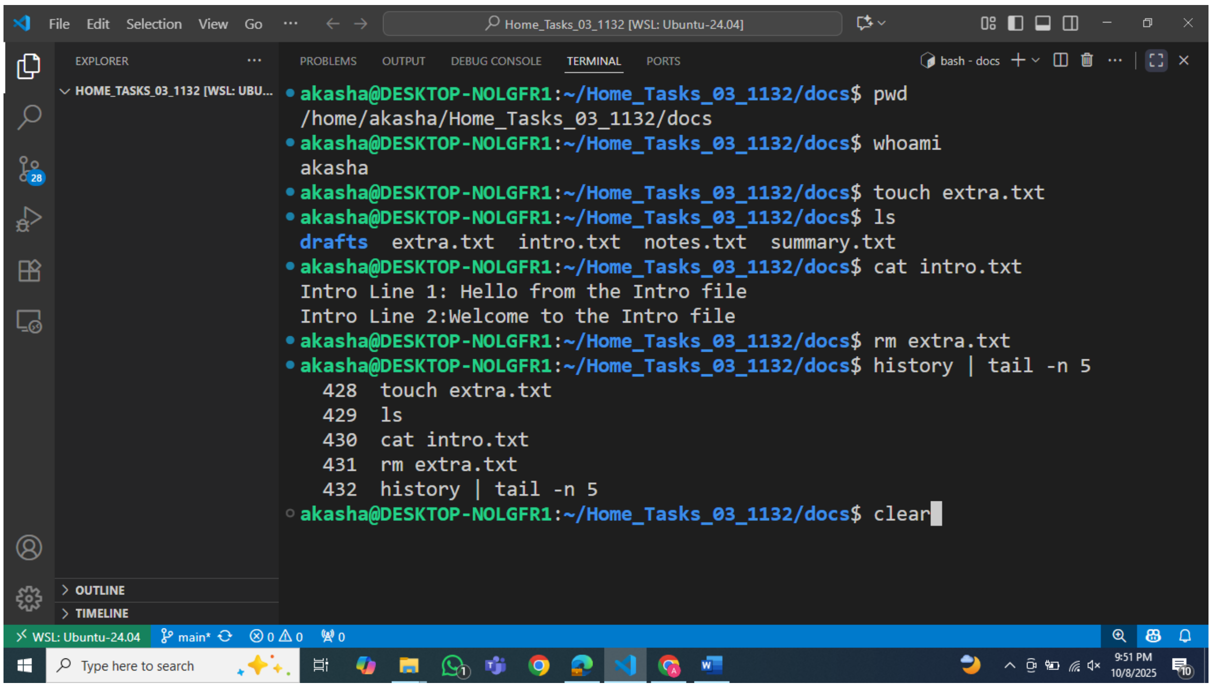
```
akasha@DESKTOP-NOLGFR1:~$ cd ~/Home_Tasks_03_1132/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ls -R
.:
data  docs  scripts

./data:
processed  raw

./data/processed:
raw2.txt

./data/raw:
raw1  raw1.txt

./docs:
drafts  intro.txt  notes.txt  summary.txt

./docs/drafts:
summary.txt

./scripts:
hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$
```

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.



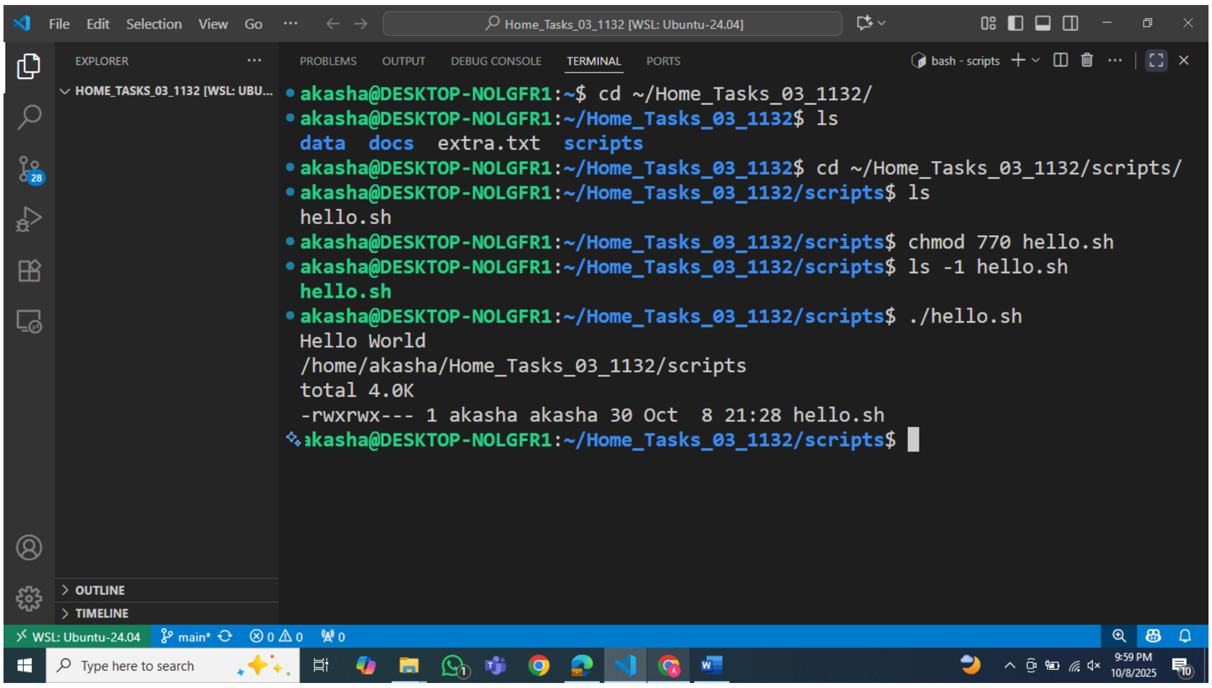
```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ pwd
/home/akasha/Home_Tasks_03_1132/docs
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ whoami
akasha
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ touch extra.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ ls
drafts  extra.txt  intro.txt  notes.txt  summary.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ cat intro.txt
Intro Line 1: Hello from the Intro file
Intro Line 2:Welcome to the Intro file
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ rm extra.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ history | tail -n 5
  428  touch extra.txt
  429  ls
  430  cat intro.txt
  431  rm extra.txt
  432  history | tail -n 5
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ clear
```

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.



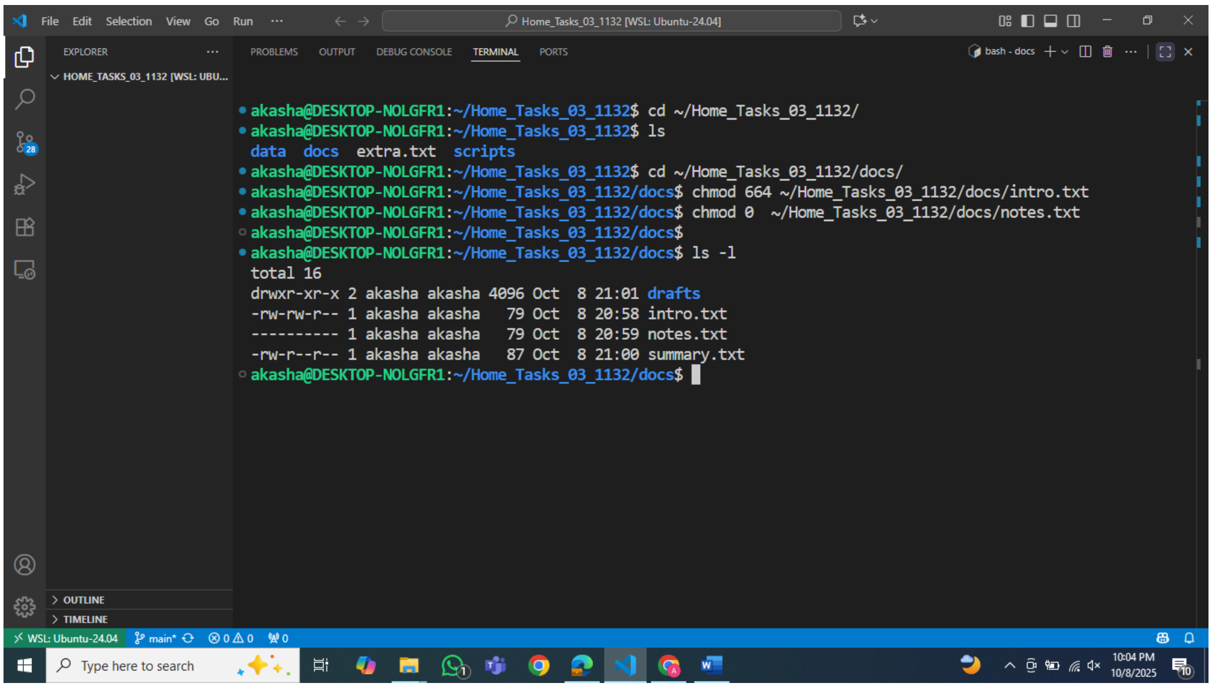
```
akasha@DESKTOP-NOLGFR1:~$ cd ~/Home_Tasks_03_1132/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ls
data  docs  extra.txt  scripts
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ cd ~/Home_Tasks_03_1132/scripts/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$ ls
hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$ chmod 770 hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$ ls -1 hello.sh
hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$ ./hello.sh
Hello World
/home/akasha/Home_Tasks_03_1132/scripts
total 4.0K
-rwxrwx--- 1 akasha akasha 30 Oct  8 21:28 hello.sh
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/scripts$
```

File Edit Selection View Go
Home_Tasks_03_1132 [WSL: Ubuntu-24.04]
EXPLORER
HOME_TASKS_03_1132 [WSL: UBU...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - scripts
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
main*
Type here to search
9:59 PM
10/8/2025

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.



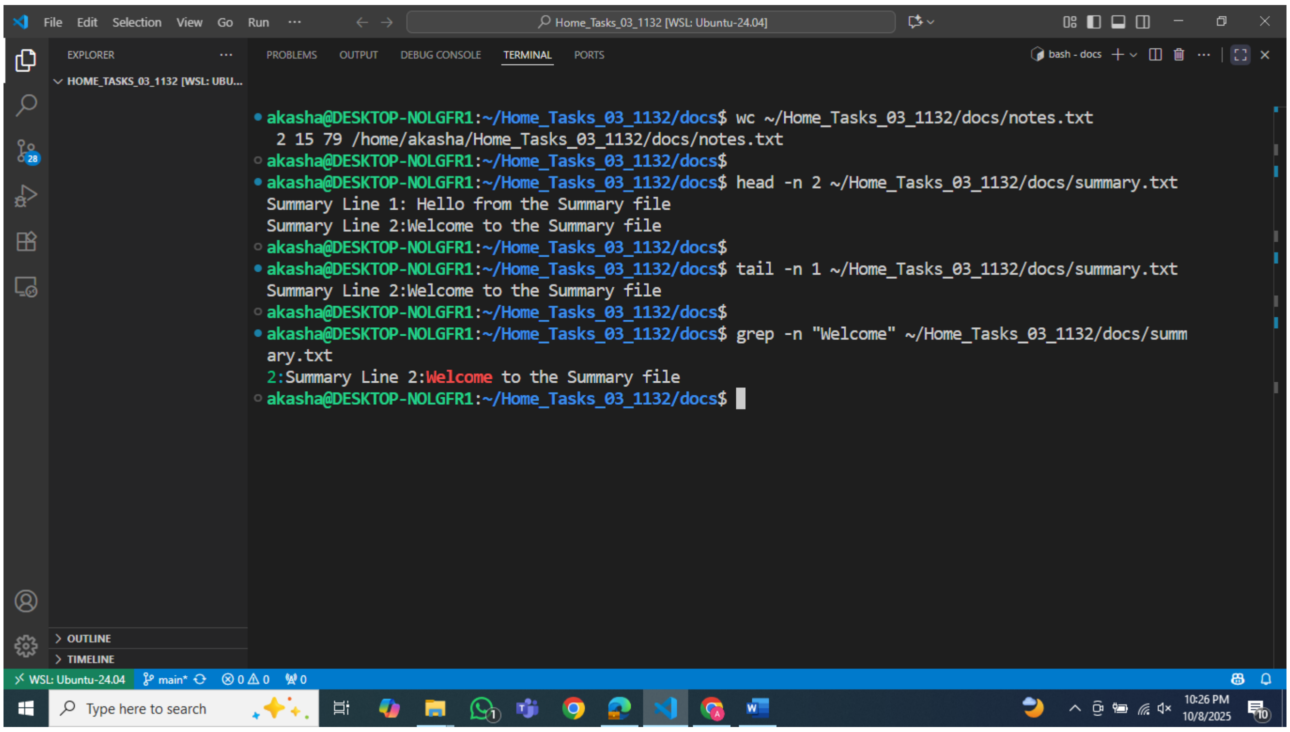
```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ cd ~/Home_Tasks_03_1132/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ls
data  docs  extra.txt  scripts
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ cd ~/Home_Tasks_03_1132/docs/
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ chmod 664 ~/Home_Tasks_03_1132/docs/intro.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ chmod 0  ~/Home_Tasks_03_1132/docs/notes.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ ls -l
total 16
drwxr-xr-x 2 akasha akasha 4096 Oct  8 21:01 drafts
-rw-rw-r-- 1 akasha akasha   79 Oct  8 20:58 intro.txt
---------- 1 akasha akasha   79 Oct  8 20:59 notes.txt
-rw-r--r-- 1 akasha akasha   87 Oct  8 21:00 summary.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
```

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.



```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ wc ~/Home_Tasks_03_1132/docs/notes.txt
  2 15 79 /home/akasha/Home_Tasks_03_1132/docs/notes.txt
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ head -n 2 ~/Home_Tasks_03_1132/docs/summary.txt
Summary Line 1: Hello from the Summary file
Summary Line 2:Welcome to the Summary file
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ tail -n 1 ~/Home_Tasks_03_1132/docs/summary.txt
Summary Line 2:Welcome to the Summary file
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$ grep -n "Welcome" ~/Home_Tasks_03_1132/docs/summ
ary.txt
2:Summary Line 2:Welcome to the Summary file
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132/docs$
```

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**.

Serial No.	PID	PPID	Command(User)
1.	2	1	/init **(root)**
2.	151	1	/usr/lib/systemd/systemd-timesyncd **(systemd+)**
3.	391	315	-bash **(akasha)**

   Run `top` for 20–30 seconds. Write down:

   Which process is consuming the most CPU. **PID:** 2239**, CPU:** 1.0%

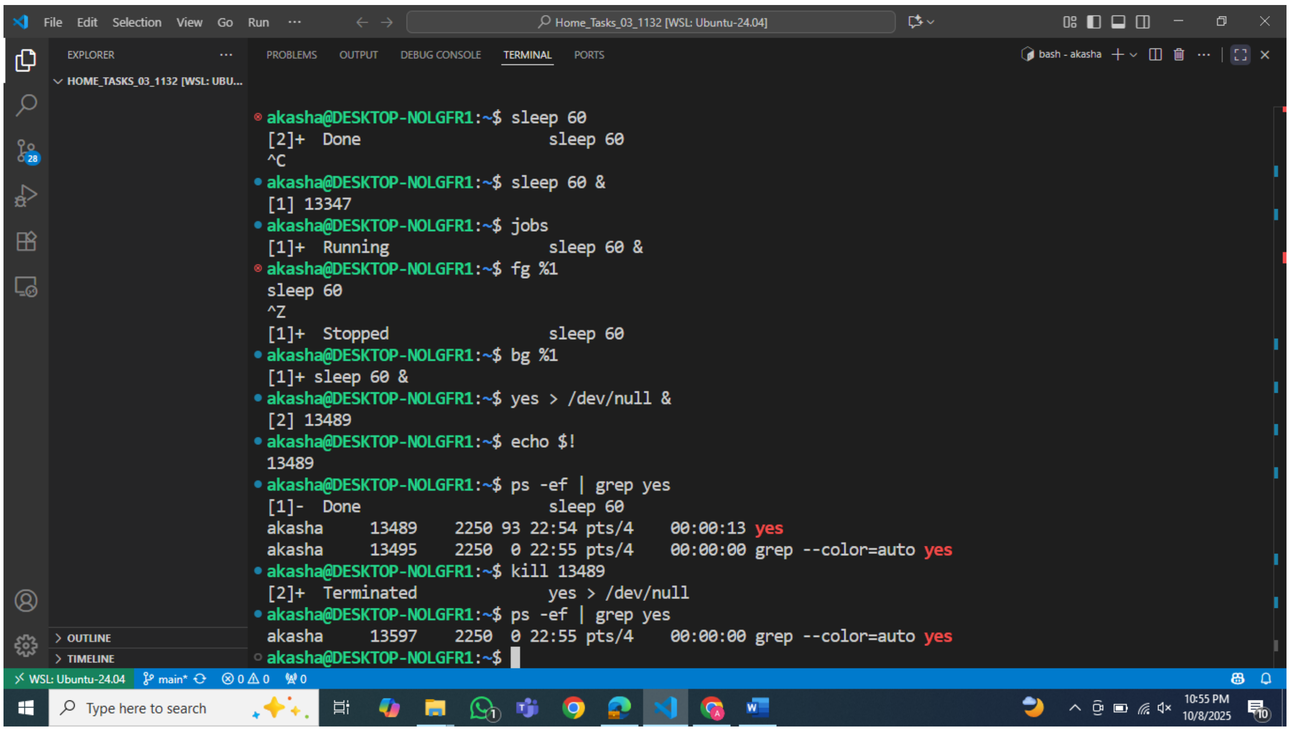
   Which process is consuming the most memory. **PID:** 1339**, CPU:** 39.2%

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`



File Edit Selection View Go Run ···
Home_Tasks_03_1132 [WSL: Ubuntu-24.04]
EXPLORER
HOME_TASKS_03_1132 [WSL: UBU...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - akasha
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
main*
Type here to search
10:55 PM
10/8/2025

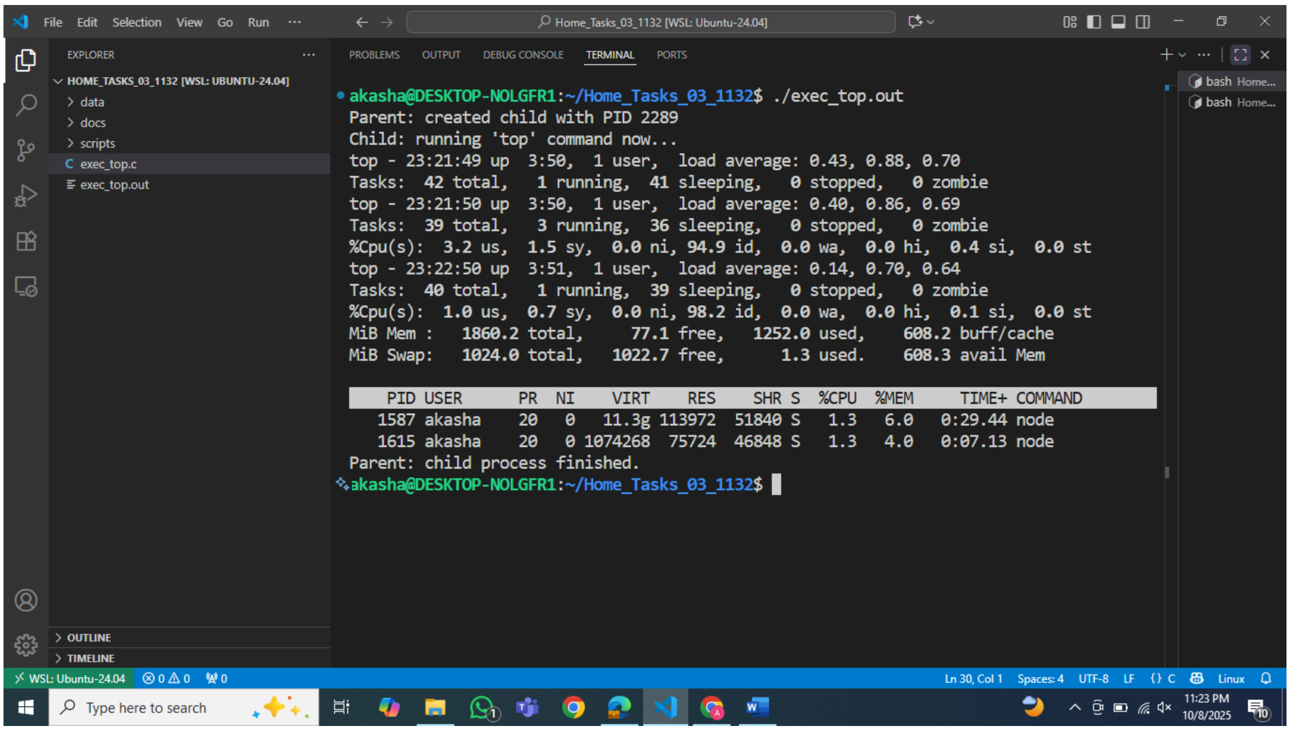
```
akasha@DESKTOP-NOLGFR1:~$ sleep 60
[2]+  Done                    sleep 60
^C
akasha@DESKTOP-NOLGFR1:~$ sleep 60 &
[1] 13347
akasha@DESKTOP-NOLGFR1:~$ jobs
[1]+  Running                 sleep 60 &
akasha@DESKTOP-NOLGFR1:~$ fg %1
sleep 60
^Z
[1]+  Stopped                 sleep 60
akasha@DESKTOP-NOLGFR1:~$ bg %1
[1]+ sleep 60 &
akasha@DESKTOP-NOLGFR1:~$ yes > /dev/null &
[2] 13489
akasha@DESKTOP-NOLGFR1:~$ echo $!
13489
akasha@DESKTOP-NOLGFR1:~$ ps -ef | grep yes
[1]-  Done                    sleep 60
akasha     13489    2250 93 22:54 pts/4    00:00:13 yes
akasha     13495    2250  0 22:55 pts/4    00:00:00 grep --color=auto yes
akasha@DESKTOP-NOLGFR1:~$ kill 13489
[2]+  Terminated              yes > /dev/null
akasha@DESKTOP-NOLGFR1:~$ ps -ef | grep yes
akasha     13597    2250  0 22:55 pts/4    00:00:00 grep --color=auto yes
akasha@DESKTOP-NOLGFR1:~$ 
```

Part 6: C Programs on Processes

Program 1 – Exec with `top`

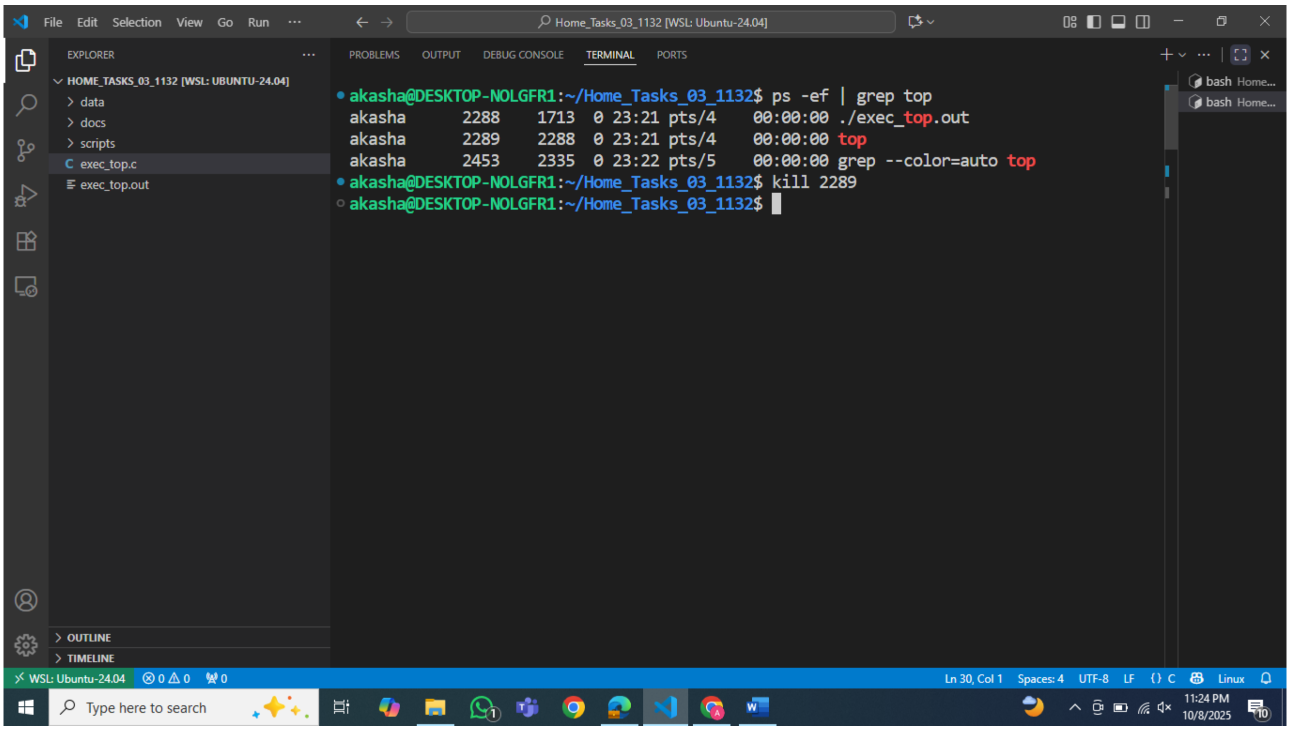
- Modify the exec program so that the child runs `top` instead of `ls -l`.
- Run the program.
- In another terminal, use `ps -ef | grep top` (or run `top`) to find the child's PID.
- Use the child's process ID to kill it manually.

Terminal A:



```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ./exec_top.out
Parent: created child with PID 2289
Child: running 'top' command now...
top - 23:21:49 up  3:50,  1 user,  load average: 0.43, 0.88, 0.70
Tasks:  42 total,   1 running,  41 sleeping,   0 stopped,   0 zombie
top - 23:21:50 up  3:50,  1 user,  load average: 0.40, 0.86, 0.69
Tasks:  39 total,   3 running,  36 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.2 us,  1.5 sy,  0.0 ni, 94.9 id,  0.0 wa,  0.0 hi,  0.4 si,  0.0 st
top - 23:22:50 up  3:51,  1 user,  load average: 0.14, 0.70, 0.64
Tasks:  40 total,   1 running,  39 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.7 sy,  0.0 ni, 98.2 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   1860.2 total,     77.1 free,   1252.0 used,    608.2 buff/cache
MiB Swap:   1024.0 total,   1022.7 free,      1.3 used.    608.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1587 akasha    20   0   11.3g 113972  51840 S   1.3   6.0   0:29.44 node
   1615 akasha    20   0 1074268  75724  46848 S   1.3   4.0   0:07.13 node
Parent: child process finished.
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$
```

Terminal B:



```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ps -ef | grep top
akasha      2288    1713  0 23:21 pts/4    00:00:00 ./exec_top.out
akasha      2289    2288  0 23:21 pts/4    00:00:00 top
akasha      2453    2335  0 23:22 pts/5    00:00:00 grep --color=auto top
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ kill 2289
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$
```

Program 2 – Incomplete Program

```
#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.

CODE:

#include <stdio.h>

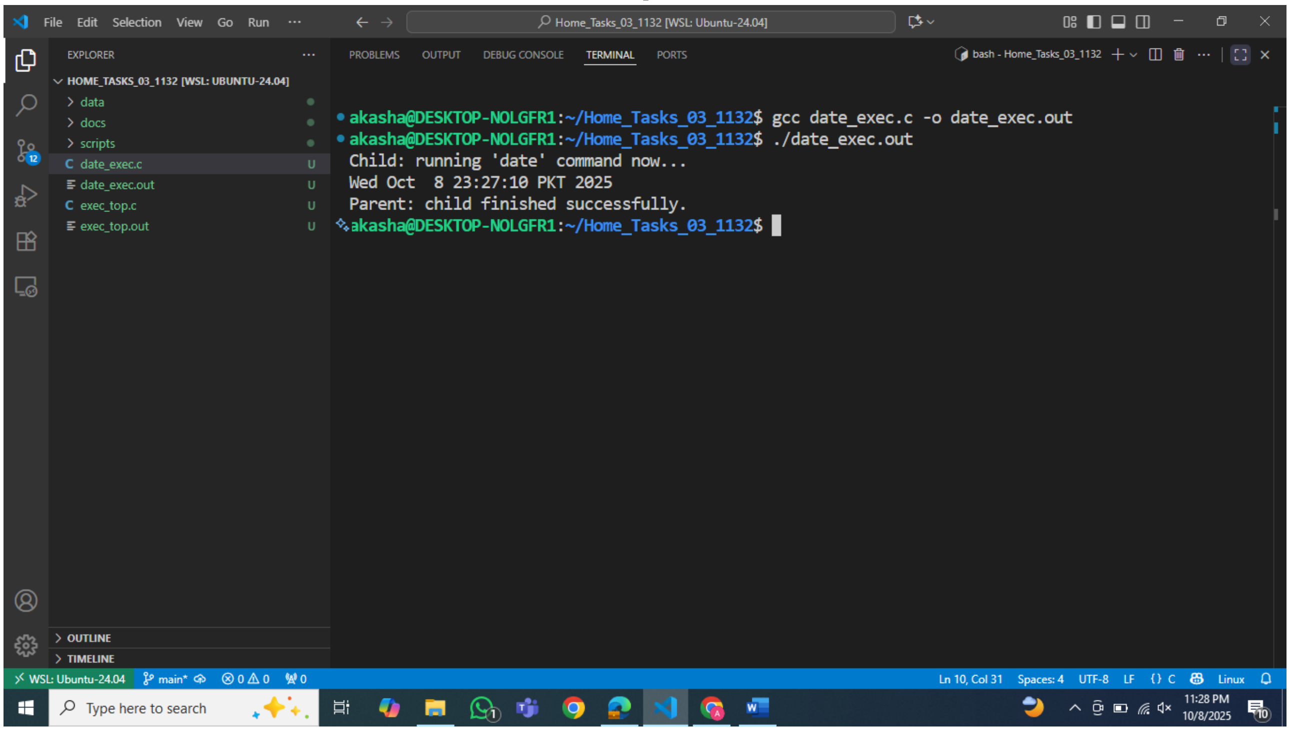
#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

```c
int main(void) {
    pid_t pid = fork();
    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }
    if (pid == 0) {
        // Child process — run "date" command
        printf("Child: running 'date' command now...\n");
        execlp("date", "date", NULL);
        perror("execlp failed");
        exit(1);
    } else {
        // Parent process — wait for child
        waitpid(pid, NULL, 0);
        printf("Parent: child finished successfully.\n");
    }
    return 0;
}
```

Output:



```
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ gcc date_exec.c -o date_exec.out
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$ ./date_exec.out
Child: running 'date' command now...
Wed Oct  8 23:27:10 PKT 2025
Parent: child finished successfully.
akasha@DESKTOP-NOLGFR1:~/Home_Tasks_03_1132$
```

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.