National Textile University

Department of Computer Science

Subject: Operating System

Submitted to: Sir Nasir Mahmood

Submitted by: Maha

Reg. number: 23-NTU-CS-1170

Lab no.: lab3 (class task)

Semester:5th

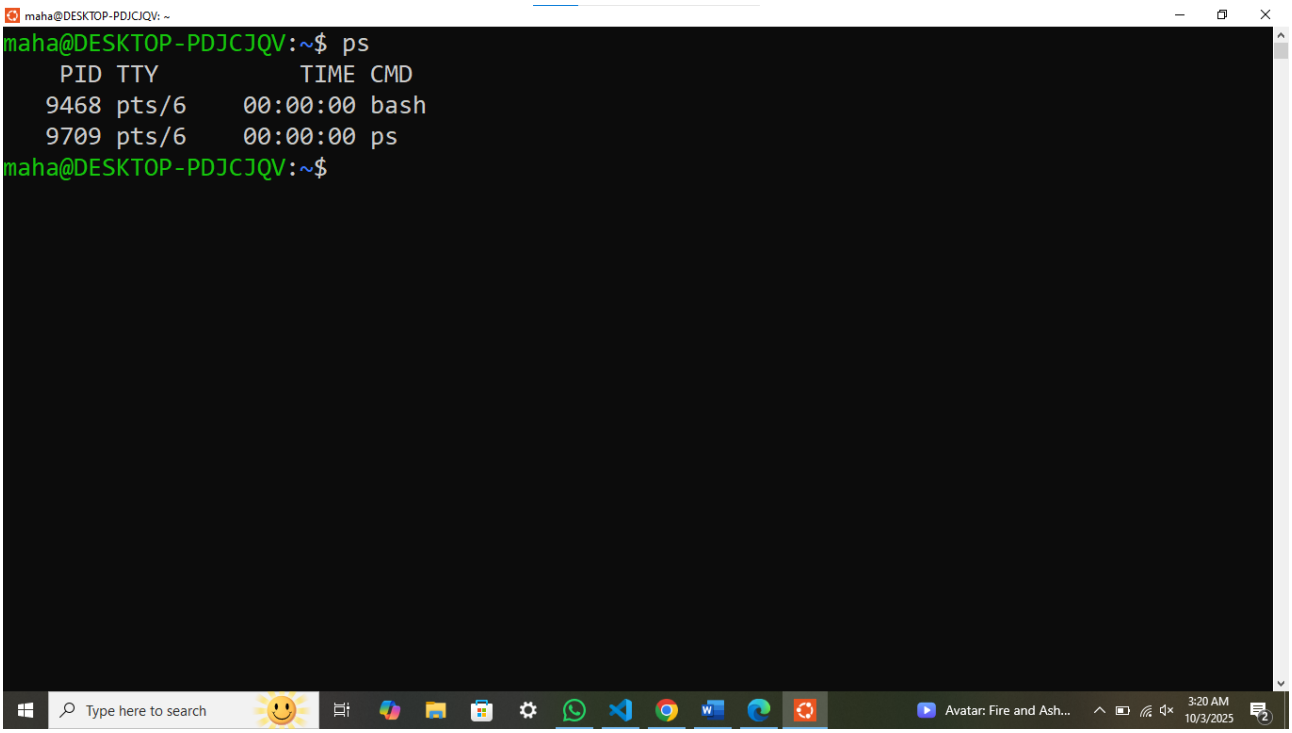
2. Linux Process Commands

2.1 Viewing Processes

`ps` → Process Status

- Shows processes in the current terminal session.

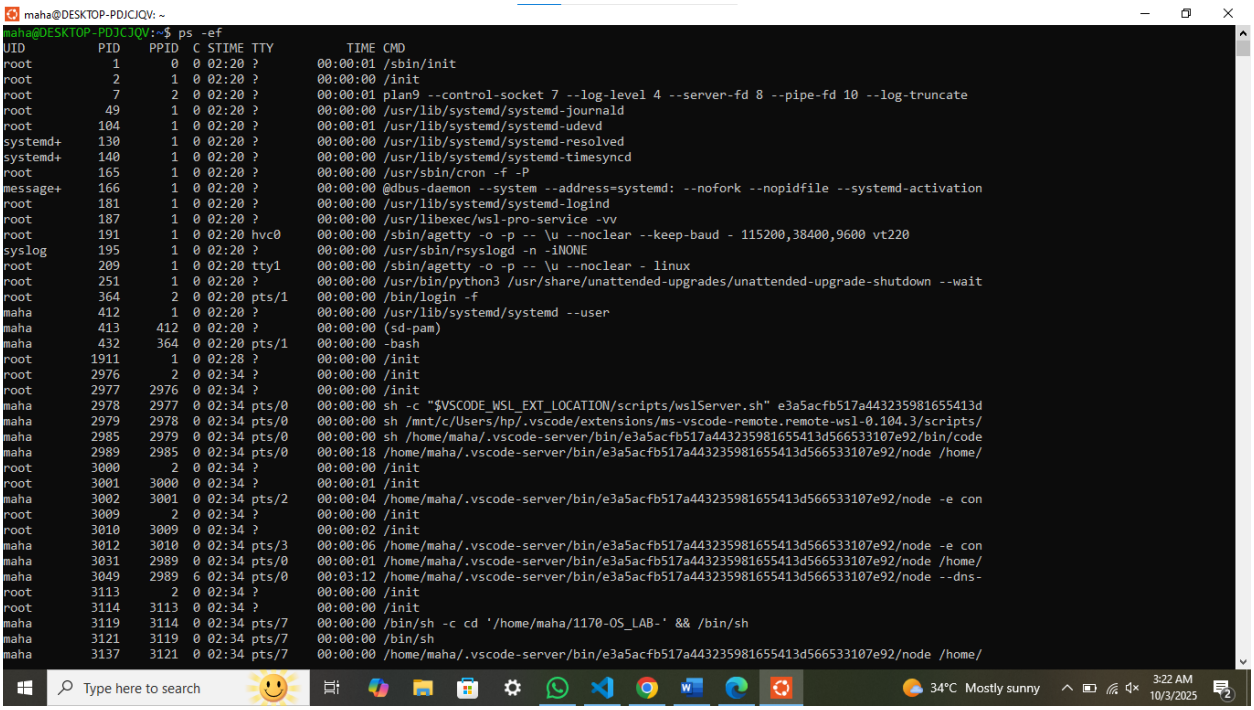
```
ps
```



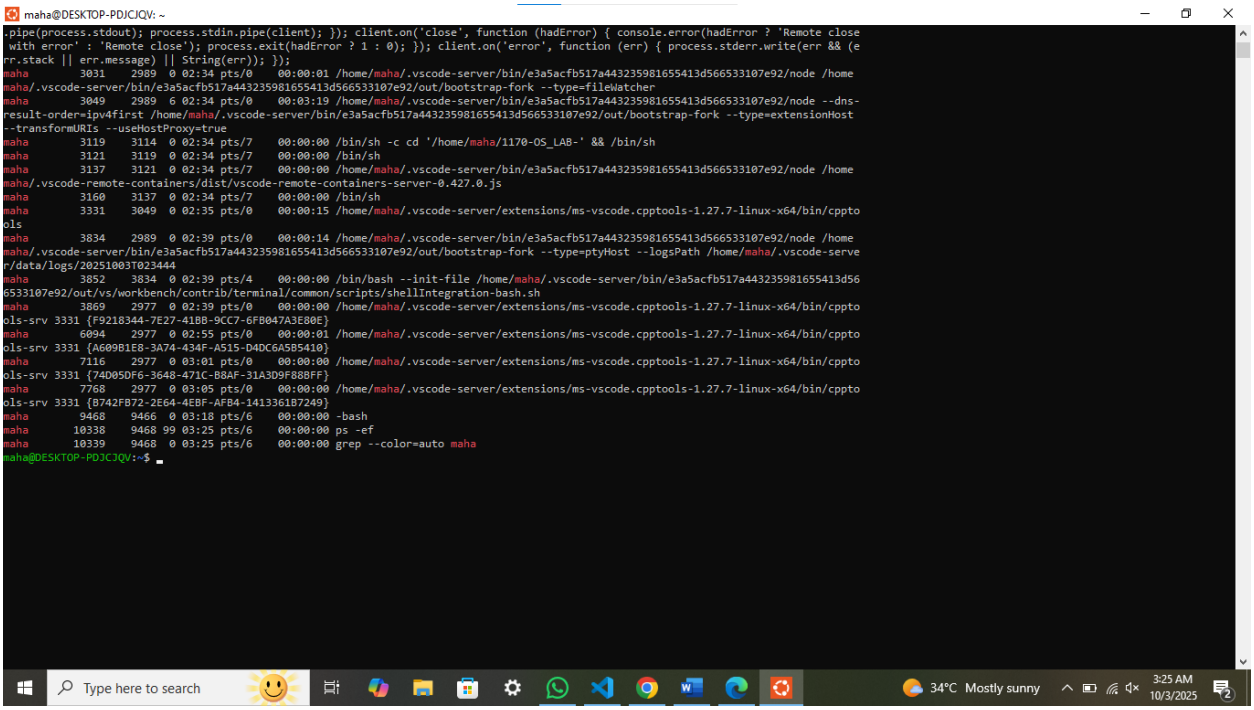
`ps -ef` → Full list of all processes

```
ps -ef
```

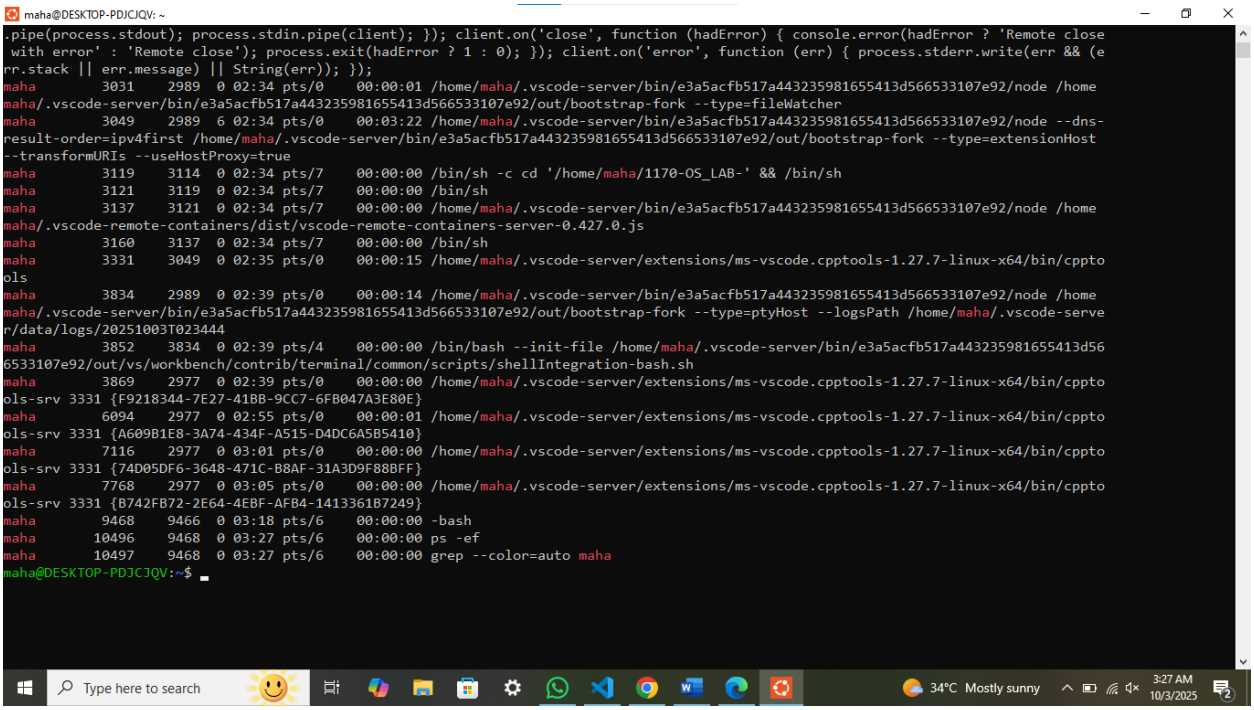
- `-e` → show all processes (not just yours).
- `-f` → full format with UID, PPID, etc.




```
ps -ef | grep bash
```



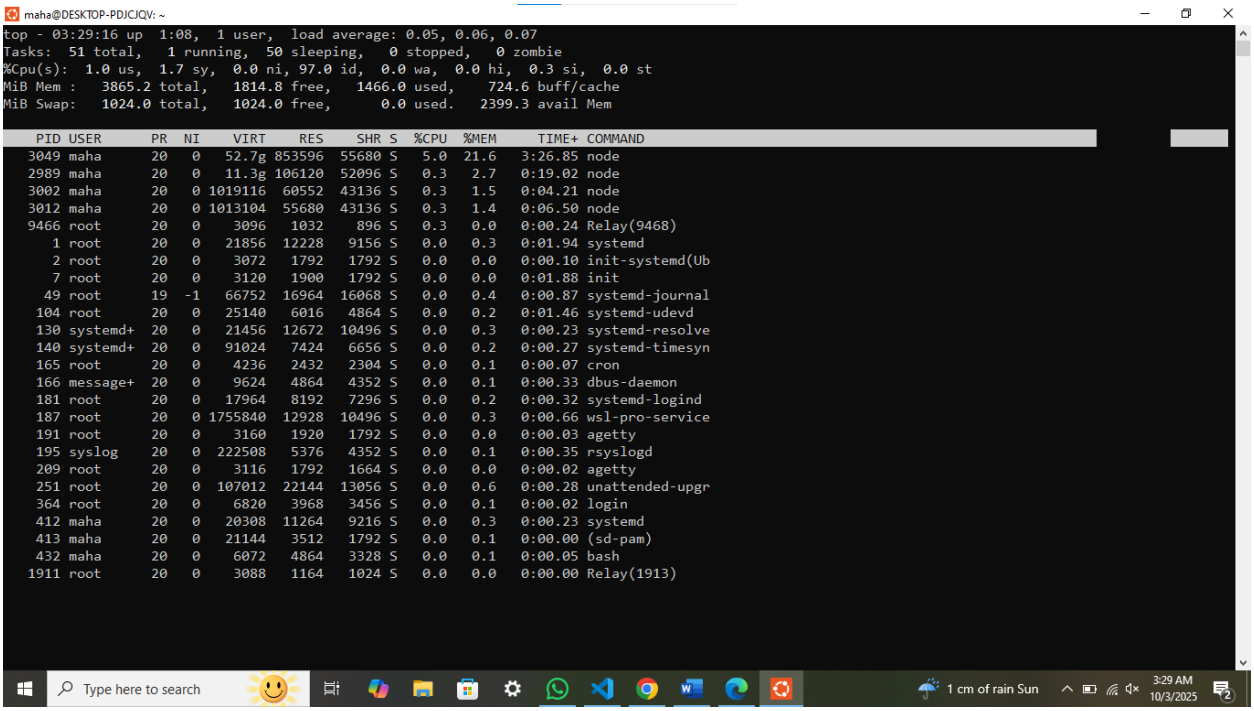
- ps -ef | grep "maha"



2.2 Monitoring Processes Interactively

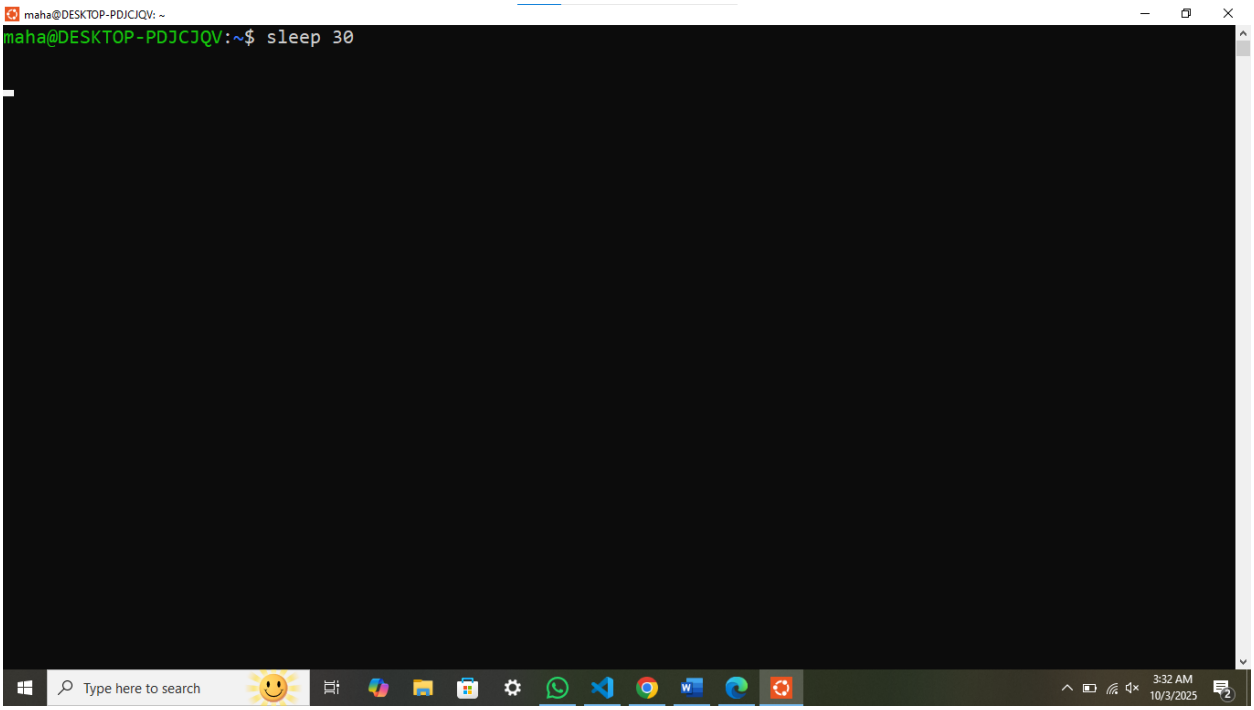
`top` → Dynamic process viewer

```
top
```



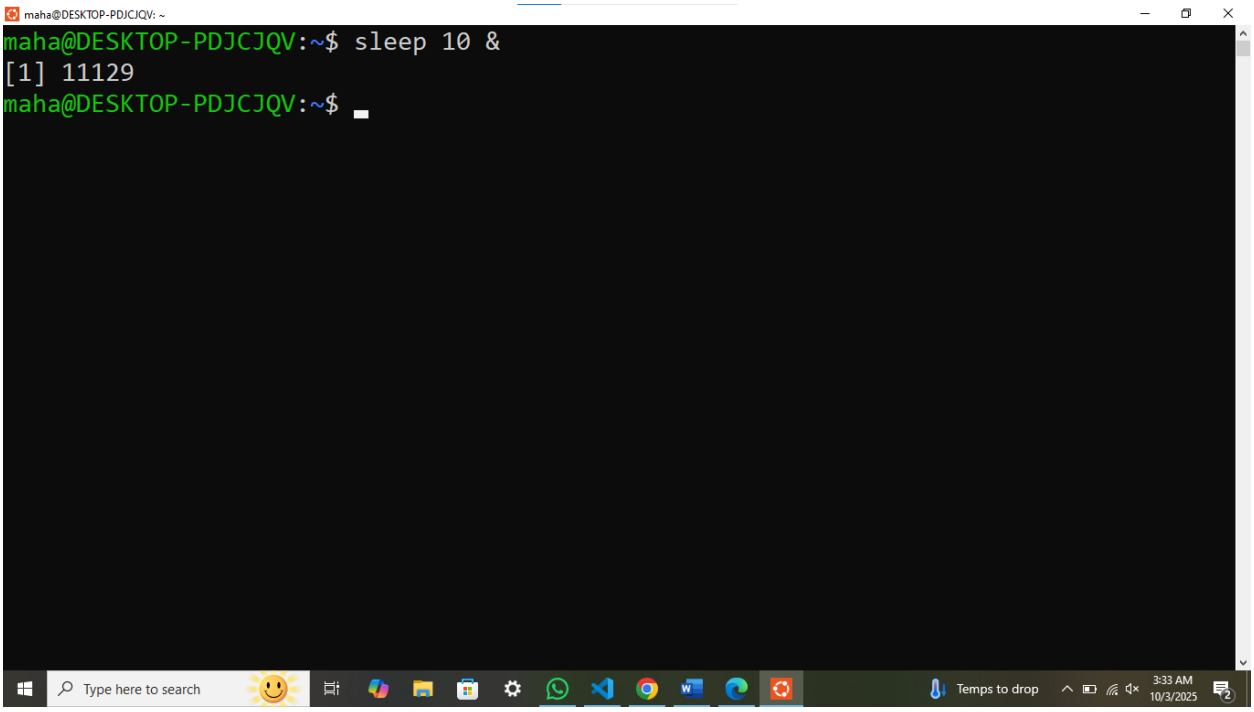
- **Foreground**: A process that takes control of the terminal until it finishes.

```
sleep 30
```



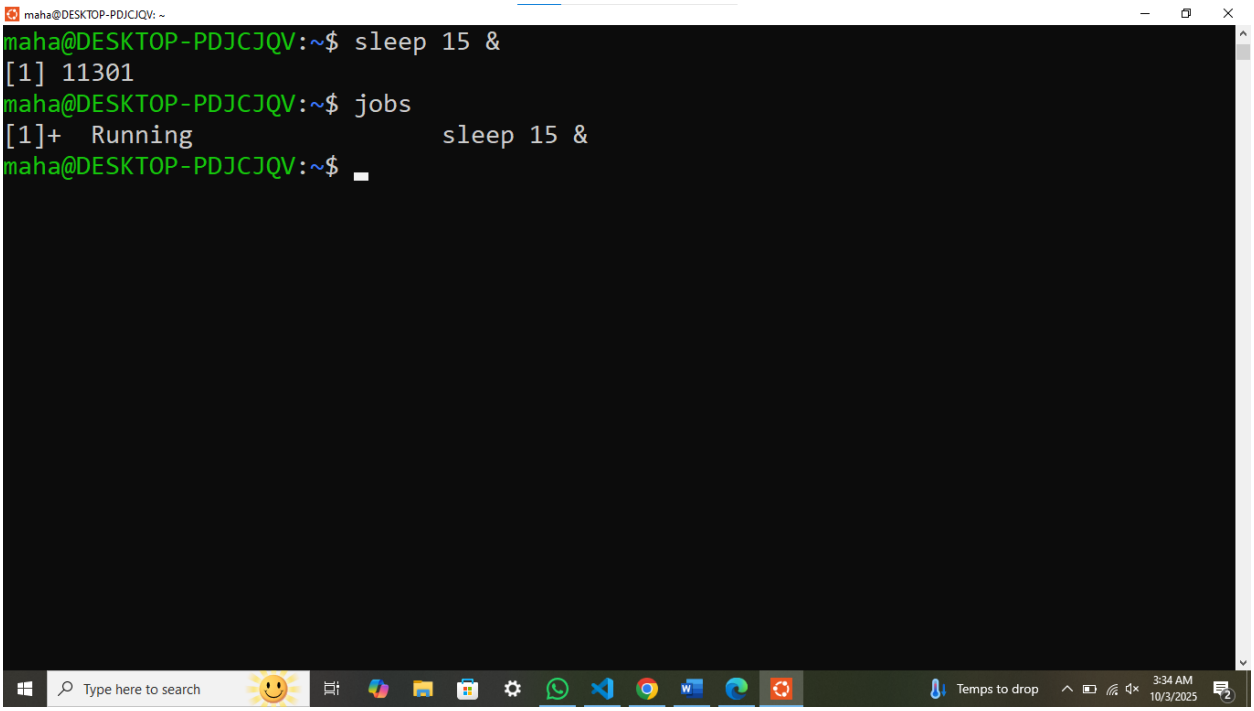
- **Background**: Add `&` to run without blocking.

```
sleep 30 &
```



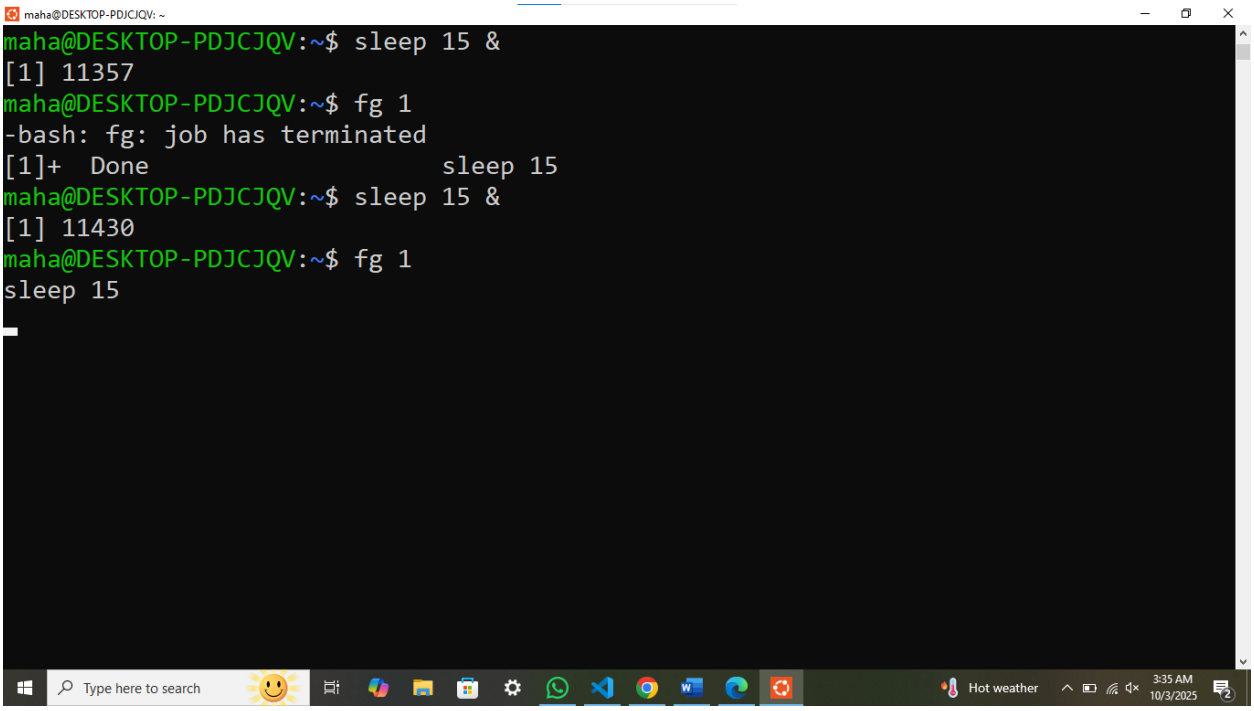
- **Check background jobs**:

```
jobs
```



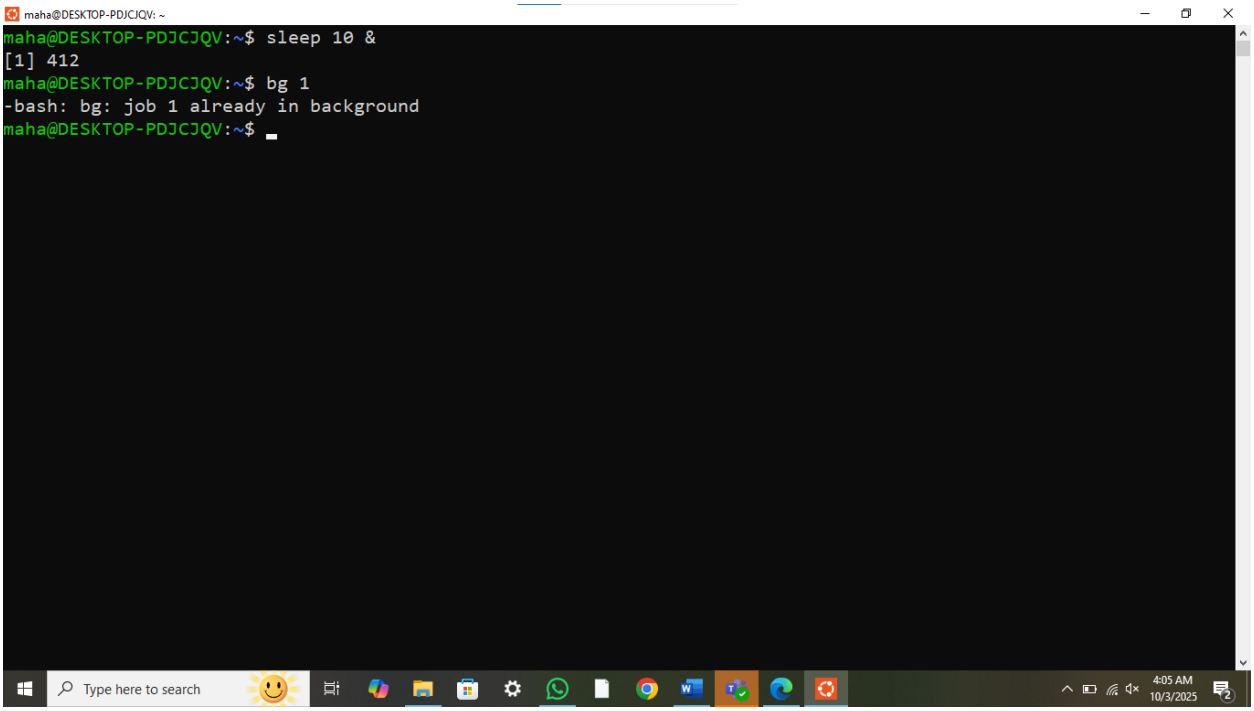
- **Bring a job to foreground**:

```
fg %1
```



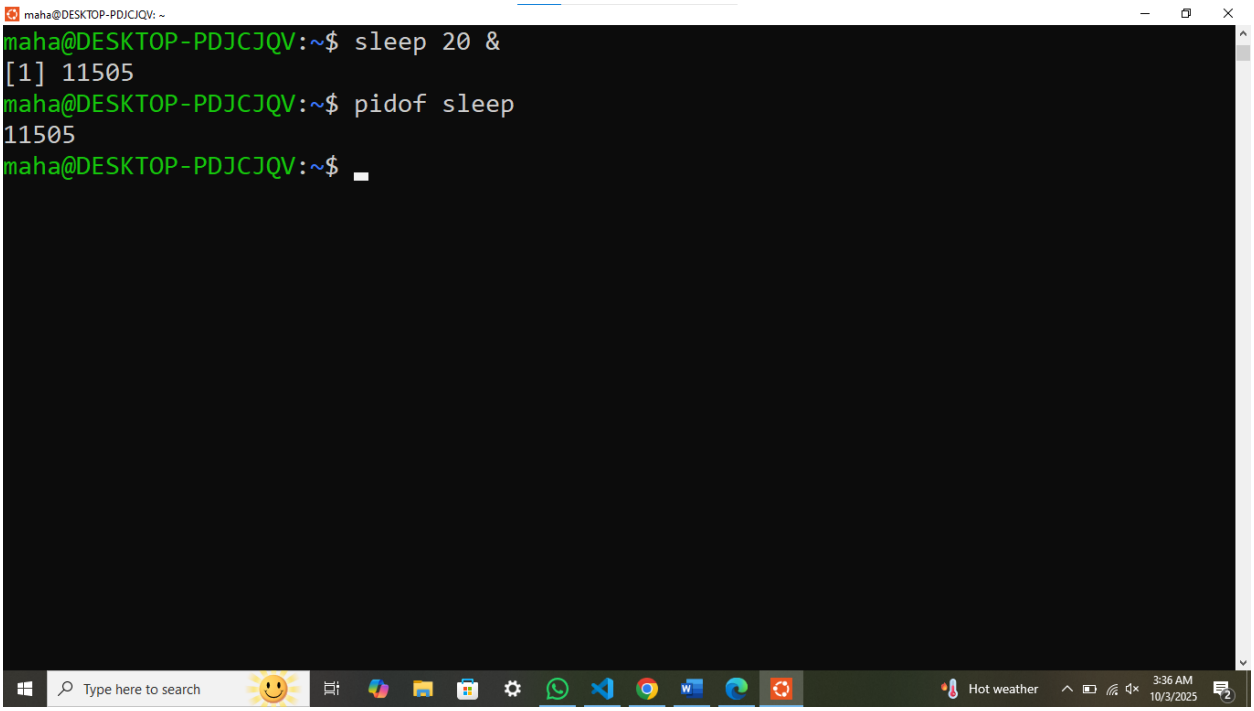
- **Resume suspended job in background:**

```
bg %1
```



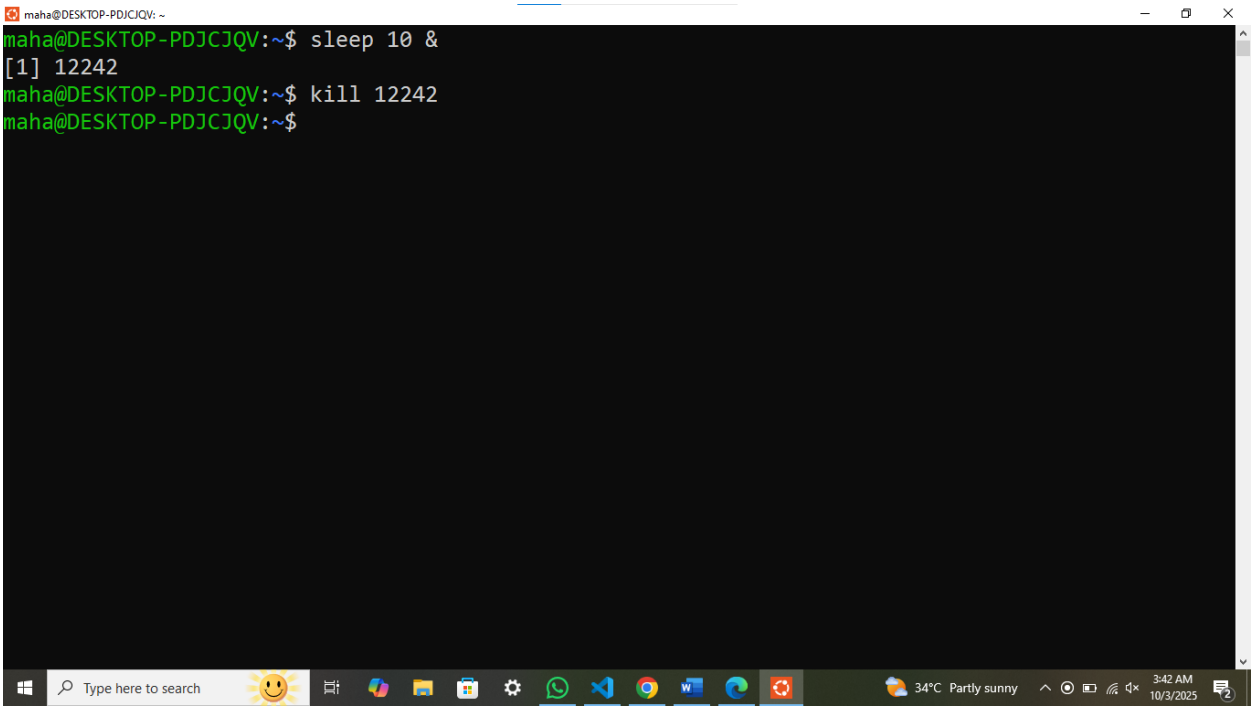
- **Get PID of a process by name:**

```
pidof sleep
```



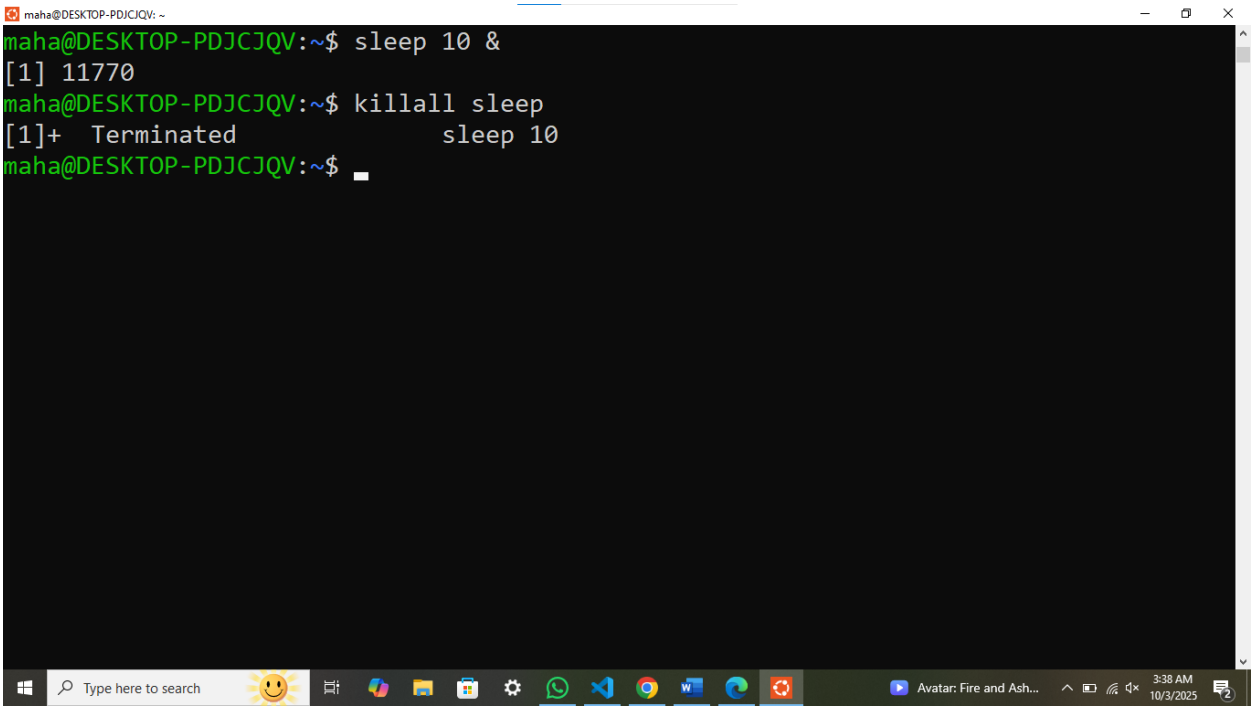
- **Kill by PID:**

```
kill -9 3421
```



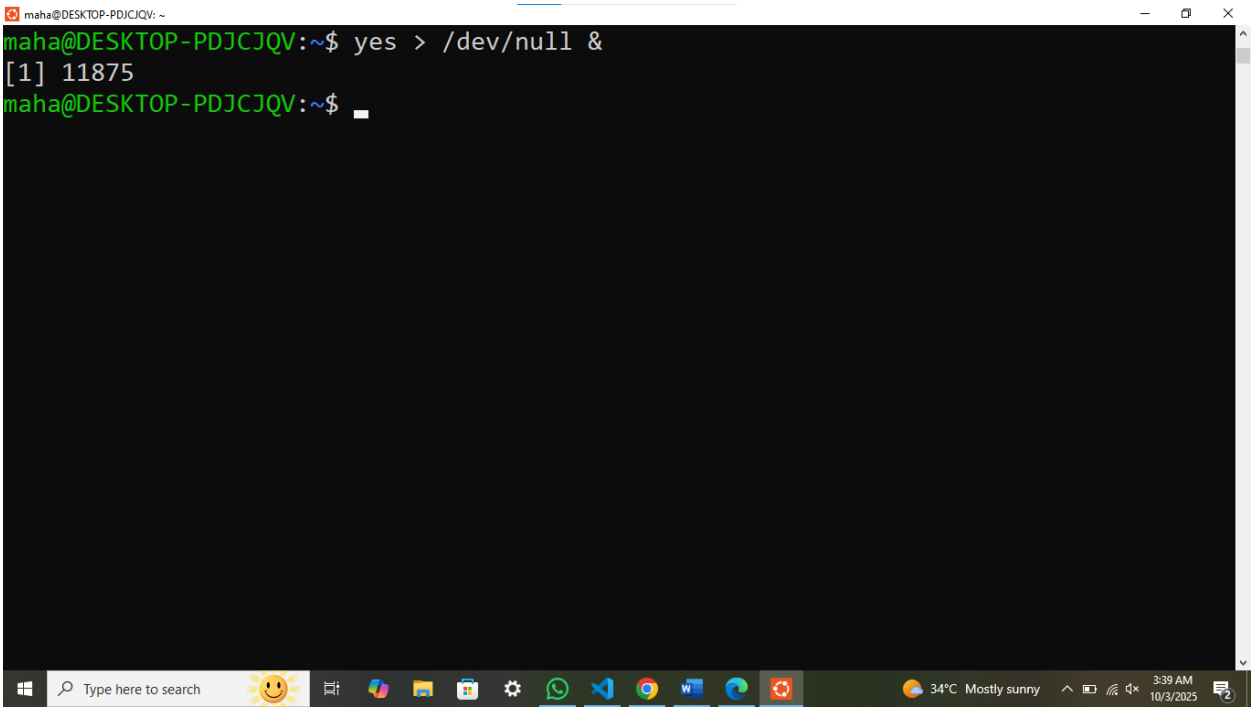
- **Kill all processes by name:**

```
killall sleep
```

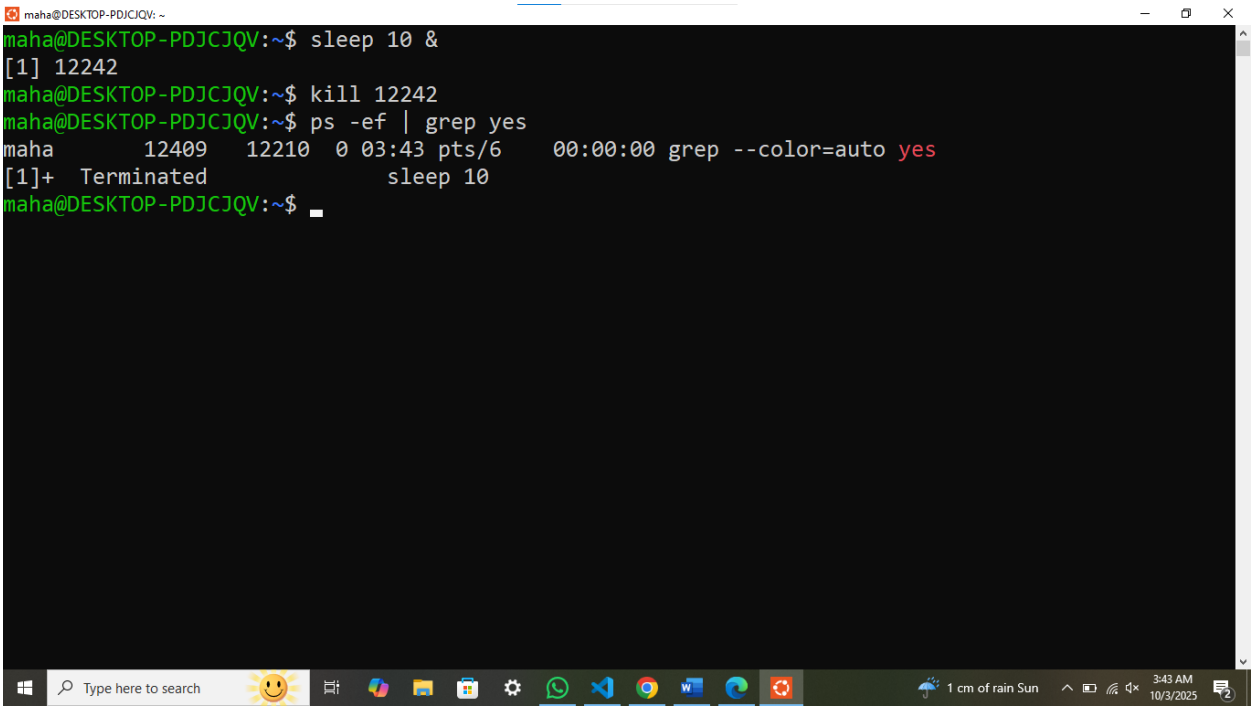


1. Run an infinite process:

```
yes > /dev/null &
```

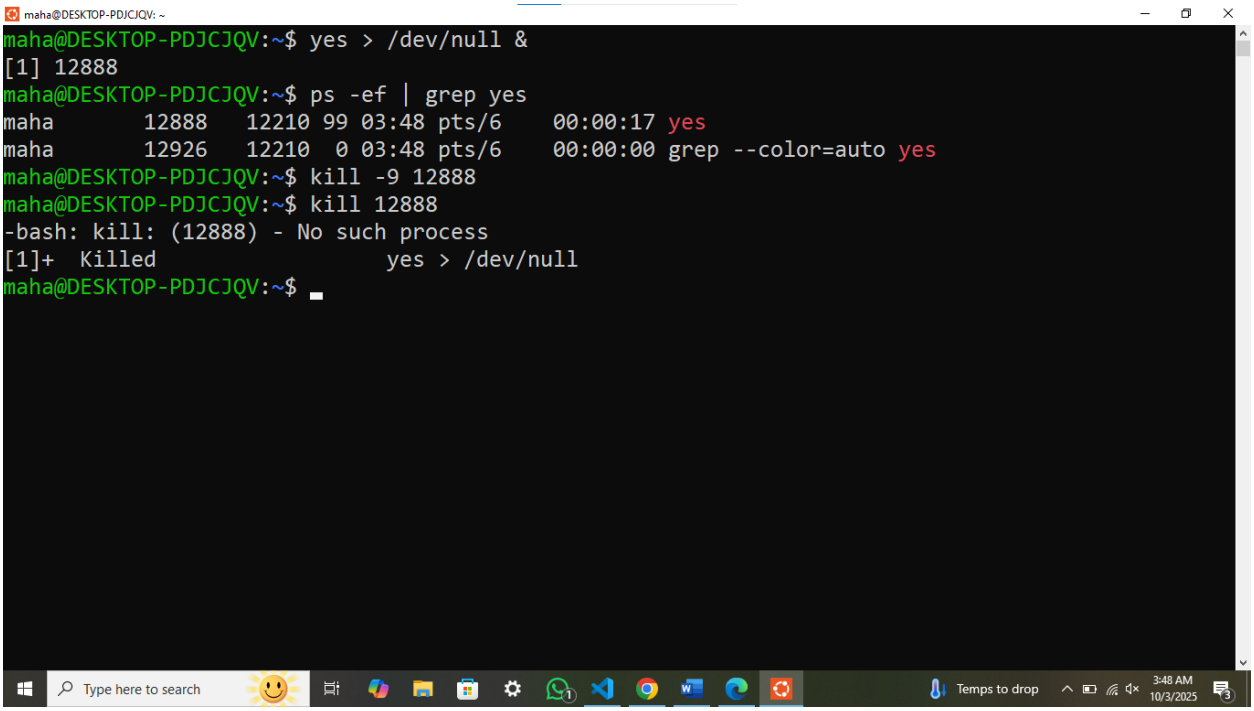



```
ps -ef | grep yes
```



3. Kill it with:

```
kill -9 <PID>
```



3. C Programs on Processes

Program 1: Print PID and PPID

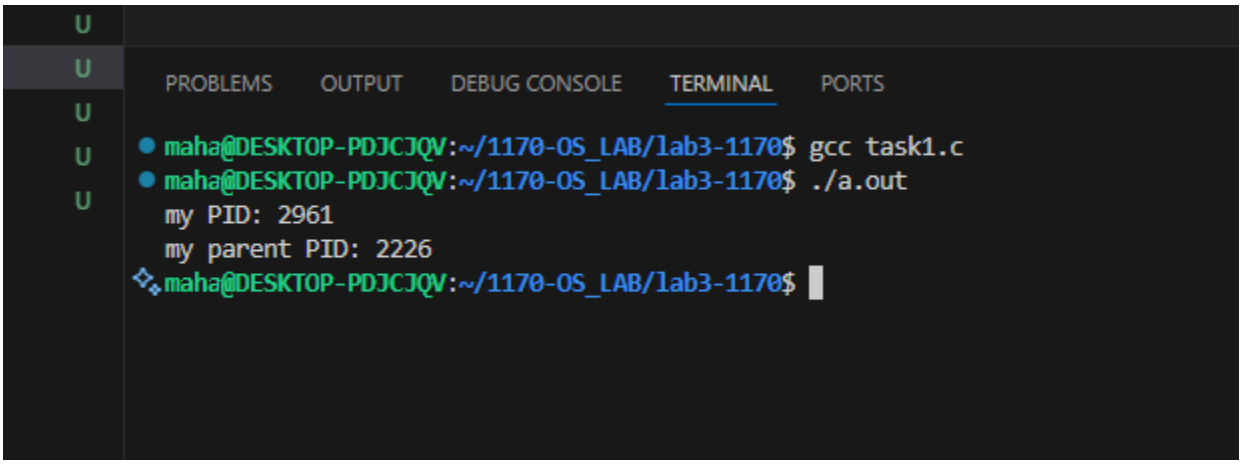
Code:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("my PID: %d\n", getpid());
    printf("my parent PID: %d\n", getppid());
    return 0;
```

```
}
```

Terminal:



Program 2: Fork – Creating Child Process

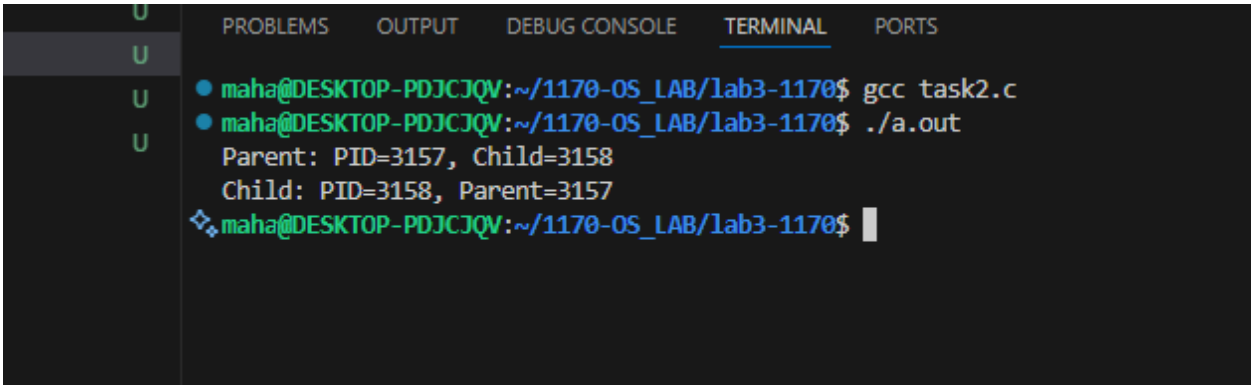
Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>   // add this header for pid_t

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

   if (pid == 0) {
      // This block runs in the child process
      printf("Child: PID=%d, Parent=%d\n", getpid(), getppid());
   } else {
      // This block runs in the parent process
      printf("Parent: PID=%d, Child=%d\n", getpid(), pid);
   }
   return 0;
```

Terminal:



Program 3: Execl – Replacing a Process

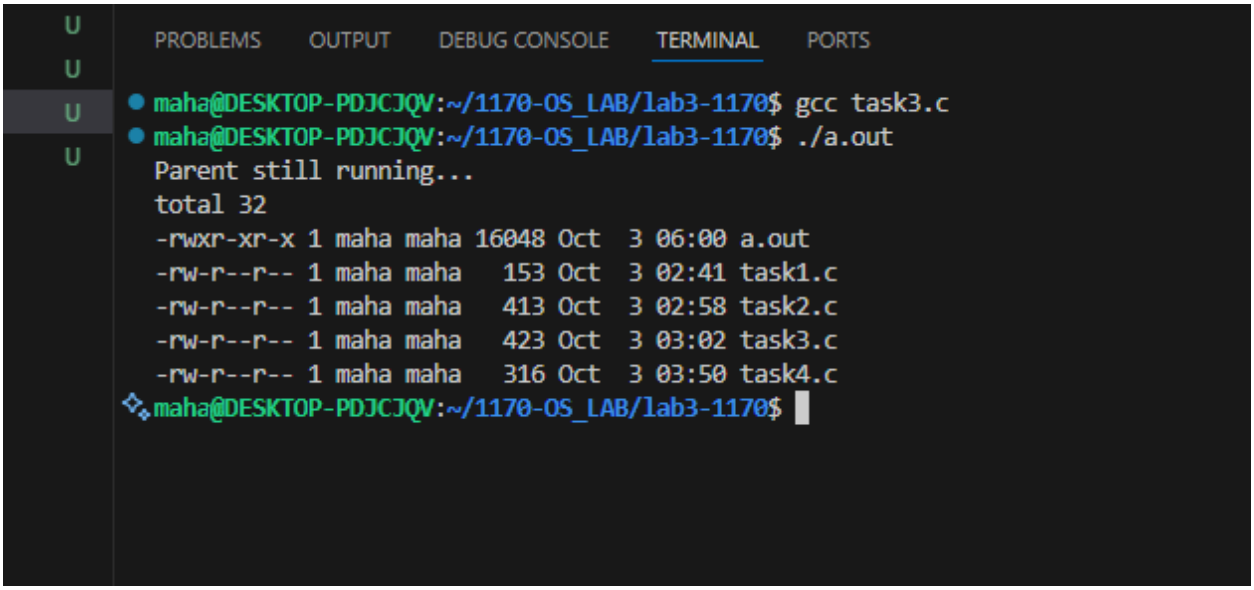
Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>   // required for pid_t

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

   if (pid == 0) {
      // Child process
      execlp("ls", "ls", "-l", NULL);
      // This line only runs if execlp fails
      printf("This will not print if exec succeeds.\n");
   } else {
      // Parent process
      printf("Parent still running...\n");
   }
   return 0;
```

Terminal:

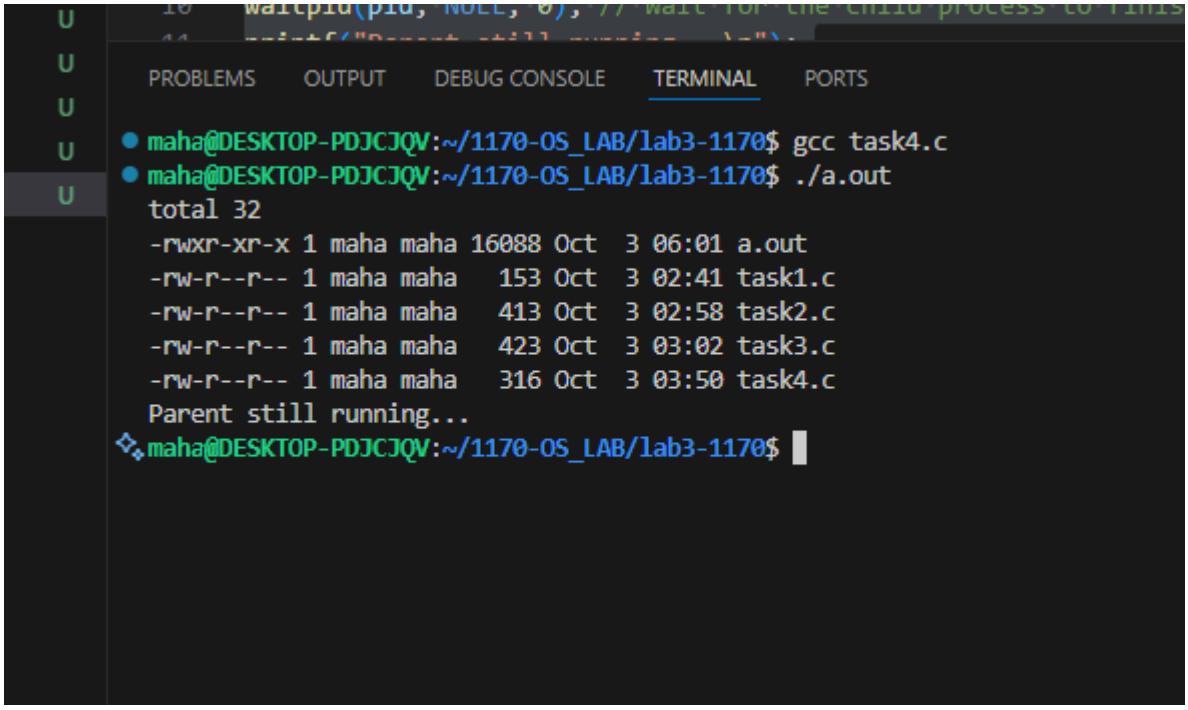


Program 4: Wait – Synchronization

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
pid_t pid = fork();
if (pid == 0) {
execlp("ls", "ls", "-l", NULL);
printf("This will not print if exec succeeds.\n");
} else {
waitpid(pid, NULL, 0); // Wait for the child process to finish
printf("Parent still running...\n");
}
return 0;
}
```

Terminal:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab3-1170$ gcc task4.c
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab3-1170$ ./a.out
total 32
-rwxr-xr-x 1 maha maha 16088 Oct 3 06:01 a.out
-rw-r--r-- 1 maha maha 153 Oct 3 02:41 task1.c
-rw-r--r-- 1 maha maha 413 Oct 3 02:58 task2.c
-rw-r--r-- 1 maha maha 423 Oct 3 03:02 task3.c
-rw-r--r-- 1 maha maha 316 Oct 3 03:50 task4.c
Parent still running...
maha@DESKTOP-PDJCJQV:~/1170-OS_LAB/lab3-1170$

.....