

# Operating Systems – COC 3071L

SE 5th A – Fall 2025

## 1. Introduction

A **process** is simply a program in execution.

- When you type a command in Linux (like `ls`), the OS creates a process for it.
- Every process has:
  - **PID (Process ID)** → unique number for each process.
  - **PPID (Parent Process ID)** → ID of the process that created it.

**State** → running, sleeping, stopped, zombie, etc.

In this lab, you will:

1. Learn Linux commands to monitor and manage processes.
2. Write C programs to create and observe processes.

---

## 2. Linux Process Commands

### 2.1 Viewing Processes

**ps** → **Process Status**

- Shows processes in the current terminal session.

```
ps
```

Output example:

```
ayez@Ayeza: ~$ ps
  PID TTY          TIME CMD
   310 pts/0    00:00:00 bash
   431 pts/0    00:00:00 ps
ayez@Ayeza: ~$
```

PID	TTY	TIME	CMD
1234	pts/0	00:00:00	bash
1256	pts/0	00:00:00	ps

- **PID** → Process ID
- **TTY** → terminal
- **TIME** → CPU time used
- **CMD** → command name

**ps -ef** → Full list of all processes

```
ps -ef
```

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

```
ayez@Ayeza:~$ ps -ef
UID          PID     PPID  C  STIME TTY          TIME CMD
root           1         0  0  15:43 ?        00:00:01 /sbin/init
root           2         1  0  15:43 ?        00:00:00 /init
root           7         2  0  15:43 ?        00:00:00 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe
root          46         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd-journald
root          93         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd-udevd
systemd+     111         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd-resolved
systemd+     112         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd-timesyncd
root        159         1  0  15:43 ?        00:00:00 /usr/sbin/cron -f -P
message+    160         1  0  15:43 ?        00:00:00 @dbus-daemon --system --address=systemd: --nofork --nopidfi
root        170         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd-logind
root        172         1  0  15:43 ?        00:00:00 /usr/libexec/wsl-pro-service -vv
root        174         1  0  15:43 hvcd0 00:00:00 /sbin/agetty -o -p -- \u --noclear --keep-baud - 115200,3840
syslog      178         1  0  15:43 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root        181         1  0  15:43 tty1    00:00:00 /sbin/agetty -o -p -- \u --noclear - linux
root        199         1  0  15:43 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-u
root        307         2  0  15:43 ?        00:00:00 /init
root        308        307  0  15:43 ?        00:00:00 /init
ayez@        310        308  0  15:43 pts/0    00:00:00 -bash
root        311         2  0  15:43 pts/1    00:00:00 /bin/login -f
ayez@        358         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd --user
ayez@        359        358  0  15:43 ?        00:00:00 (sd-pam)
ayez@        384        311  0  15:43 pts/1    00:00:00 -bash
root        434         2  0  15:45 ?        00:00:00 /init
root        435        434  0  15:45 ?        00:00:00 /init
ayez@        440        435  0  15:45 pts/2    00:00:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslServer.sh" e3a5a
root        441        440  0  15:45 pts/2    00:00:00 sh -c "code-server --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-1252369057 --use-host-proxy --without-b
--accept-server-license-terms --telemetry-level=all
ayez@        441        440  0  15:45 pts/2    00:00:00 sh /mnt/c/Users/Dell/.vscode/extensions/ms-vscode-remote.remote-wsl-0.104.
5413d566533107e92/stable/code-server .vscode-server --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-
ser-env-var --disable-websocket-compression --accept-server-license-terms --telemetry-level=all
ayez@        520        441  0  15:45 pts/2    00:00:00 sh /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/
connection-token=1362810641-1704448701-3371422609-1252369057 --use-host-proxy --without-browser-env-var --disable-websocket-com
etry-level=all
ayez@        524        520  12 15:45 pts/2    00:00:09 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
443235981655413d566533107e92/out/server-main.js --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-
env-var --disable-websocket-compression --accept-server-license-terms --telemetry-level=all
ayez@        541        537  0  15:45 pts/3    00:00:00 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
n.pause(); const client = net.createConnection({ host: '127.0.0.1', port: 36751 }, () => { client.pipe(process.stdout); proces
function (hadError) { console.error(hadError ? 'Remote close with error' : 'Remote close'); process.exit(hadError ? 1 : 0); });
tderr.write(err && (err.stack || err.message) || String(err)); });
ayez@        558        555  0  15:45 pts/4    00:00:00 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
n.pause(); const client = net.createConnection({ host: '127.0.0.1', port: 36751 }, () => { client.pipe(process.stdout); proces
function (hadError) { console.error(hadError ? 'Remote close with error' : 'Remote close'); process.exit(hadError ? 1 : 0); });
tderr.write(err && (err.stack || err.message) || String(err)); });
ayez@        578        524  13 15:45 pts/2    00:00:10 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
code-server/bin/e3a5acfb517a443235981655413d566533107e92/out/bootstrap-fork --type=extensionHost --transformURIs --useHostPro
ayez@        819        310  0  15:46 pts/0    00:00:00 ps -ef
ayez@        820        310  0  15:46 pts/0    00:00:00 grep --color=auto ayeza
ayez@Ayeza:~$
```

Try:

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

This finds all processes related to the `bash` shell.

```
ayez@Ayeza:~$ ps -ef | grep ayeza
ayez@        310        308  0  15:43 pts/0    00:00:00 -bash
ayez@        358         1  0  15:43 ?        00:00:00 /usr/lib/systemd/systemd --user
ayez@        359        358  0  15:43 ?        00:00:00 (sd-pam)
ayez@        384        311  0  15:43 pts/1    00:00:00 -bash
ayez@        440        435  0  15:45 pts/2    00:00:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslServer.sh" e3a5acfb517a443235981655413d566533107e92
e-server --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-1252369057 --use-host-proxy --without-b
--accept-server-license-terms --telemetry-level=all
ayez@        441        440  0  15:45 pts/2    00:00:00 sh /mnt/c/Users/Dell/.vscode/extensions/ms-vscode-remote.remote-wsl-0.104.
5413d566533107e92/stable/code-server .vscode-server --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-
ser-env-var --disable-websocket-compression --accept-server-license-terms --telemetry-level=all
ayez@        520        441  0  15:45 pts/2    00:00:00 sh /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/
connection-token=1362810641-1704448701-3371422609-1252369057 --use-host-proxy --without-browser-env-var --disable-websocket-com
etry-level=all
ayez@        524        520  12 15:45 pts/2    00:00:09 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
443235981655413d566533107e92/out/server-main.js --host=127.0.0.1 --port=0 --connection-token=1362810641-1704448701-3371422609-
env-var --disable-websocket-compression --accept-server-license-terms --telemetry-level=all
ayez@        541        537  0  15:45 pts/3    00:00:00 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
n.pause(); const client = net.createConnection({ host: '127.0.0.1', port: 36751 }, () => { client.pipe(process.stdout); proces
function (hadError) { console.error(hadError ? 'Remote close with error' : 'Remote close'); process.exit(hadError ? 1 : 0); });
tderr.write(err && (err.stack || err.message) || String(err)); });
ayez@        558        555  0  15:45 pts/4    00:00:00 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
n.pause(); const client = net.createConnection({ host: '127.0.0.1', port: 36751 }, () => { client.pipe(process.stdout); proces
function (hadError) { console.error(hadError ? 'Remote close with error' : 'Remote close'); process.exit(hadError ? 1 : 0); });
tderr.write(err && (err.stack || err.message) || String(err)); });
ayez@        578        524  13 15:45 pts/2    00:00:10 /home/ayez/.vscode-server/bin/e3a5acfb517a443235981655413d566533107e92/no
code-server/bin/e3a5acfb517a443235981655413d566533107e92/out/bootstrap-fork --type=extensionHost --transformURIs --useHostPro
ayez@        819        310  0  15:46 pts/0    00:00:00 ps -ef
ayez@        820        310  0  15:46 pts/0    00:00:00 grep --color=auto ayeza
ayez@Ayeza:~$
```

## 2.2 Monitoring Processes Interactively

**top** → Dynamic process viewer

**top**

- Displays running processes with CPU and memory usage.
- Press **q** to quit.
- Press **k** inside **top** to kill a process (enter PID).
- Press **h** for help.

```
top - 15:47:08 up 3 min, 1 user, load average: 0.20, 0.14, 0.05
Tasks: 36 total, 1 running, 35 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.6 us, 0.5 sy, 0.0 ni, 98.6 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
Mem Mem : 3857.2 total, 2698.6 free, 675.4 used, 630.3 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used, 3181.9 avail Mem

  PID USER   PR   NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 578 ayeza   20    0 31.9g 153844 53888 S   1.5   3.9   0:11.22 node
    1 root    20    0 21848 12516 9316 S   0.0   0.3   0:01.52 systemd
    2 root    20    0 3072 1664 1664 S   0.0   0.0   0:00.02 init-systemd(ub
    7 root    20    0 3104 2060 1920 S   0.0   0.1   0:00.02 init
   46 root    19   -1 42164 15616 14720 S   0.0   0.4   0:00.43 systemd-journal
   93 root    20    0 25908 6016 4864 S   0.0   0.2   0:00.40 systemd-udevd
  111 systemd+ 20    0 21456 12416 10240 S   0.0   0.3   0:00.28 systemd-resolve
  112 systemd+ 20    0 91024 7424 6656 S   0.0   0.2   0:00.22 systemd-timesyn
  159 root    20    0 4236 2560 2432 S   0.0   0.1   0:00.02 cron
  160 message+ 20    0 9632 4864 4480 S   0.0   0.1   0:00.11 dbus-daemon
  170 root    20    0 17960 8320 7424 S   0.0   0.2   0:00.14 systemd-logind
  172 root    20    0 1756096 12544 10624 S   0.0   0.3   0:00.17 wsl-pro-service
  174 root    20    0 3160 1664 1664 S   0.0   0.0   0:00.01 agetty
  178 syslog 20    0 222508 5760 4480 S   0.0   0.1   0:00.13 rsyslogd
  181 root    20    0 3116 1664 1664 S   0.0   0.0   0:00.02 agetty
  199 root    20    0 107032 22400 13184 S   0.0   0.6   0:00.44 unattended-upgr
  307 root    20    0 3076 896 896 S   0.0   0.0   0:00.00 SessionLeader
  308 root    20    0 3092 1824 896 S   0.0   0.0   0:00.01 Relay(318)
  310 ayeza   20    0 6072 5248 3584 S   0.0   0.1   0:00.07 bash
  311 root    20    0 6824 4124 3712 S   0.0   0.1   0:00.02 login
  358 ayeza   20    0 20112 10880 9088 S   0.0   0.3   0:00.28 systemd
  359 ayeza   20    0 21152 3520 1792 S   0.0   0.1   0:00.00 (sd-pam)
  384 ayeza   20    0 6072 4864 3456 S   0.0   0.1   0:00.03 bash
  434 root    20    0 3076 1828 896 S   0.0   0.0   0:00.00 SessionLeader
  435 root    20    0 3092 1160 1824 S   0.0   0.0   0:00.01 Relay(448)
  440 ayeza   20    0 2800 1536 1536 S   0.0   0.0   0:00.01 sh
  441 ayeza   20    0 2800 1664 1664 S   0.0   0.0   0:00.00 sh
  520 ayeza   20    0 2800 1792 1792 S   0.0   0.0   0:00.00 sh
  524 ayeza   20    0 11.3g 113172 52864 S   0.0   2.9   0:09.69 node
  535 root    20    0 3088 1028 896 S   0.0   0.0   0:00.00 SessionLeader
  537 root    20    0 3088 1164 1824 S   0.0   0.0   0:00.18 Relay(541)
  541 ayeza   20    0 953032 59264 43264 S   0.0   1.5   0:00.50 node
  554 root    20    0 3088 900 768 S   0.0   0.0   0:00.00 SessionLeader
  555 root    20    0 3088 1164 1824 S   0.0   0.0   0:00.17 Relay(558)
```

## 2.3 Foreground and Background Jobs

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

**sleep 30**

→ You cannot type new commands until it finishes.

•

**sleep 30 &**

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

→ Terminal is free while the command runs.

•

**jobs**

**Check background jobs:**

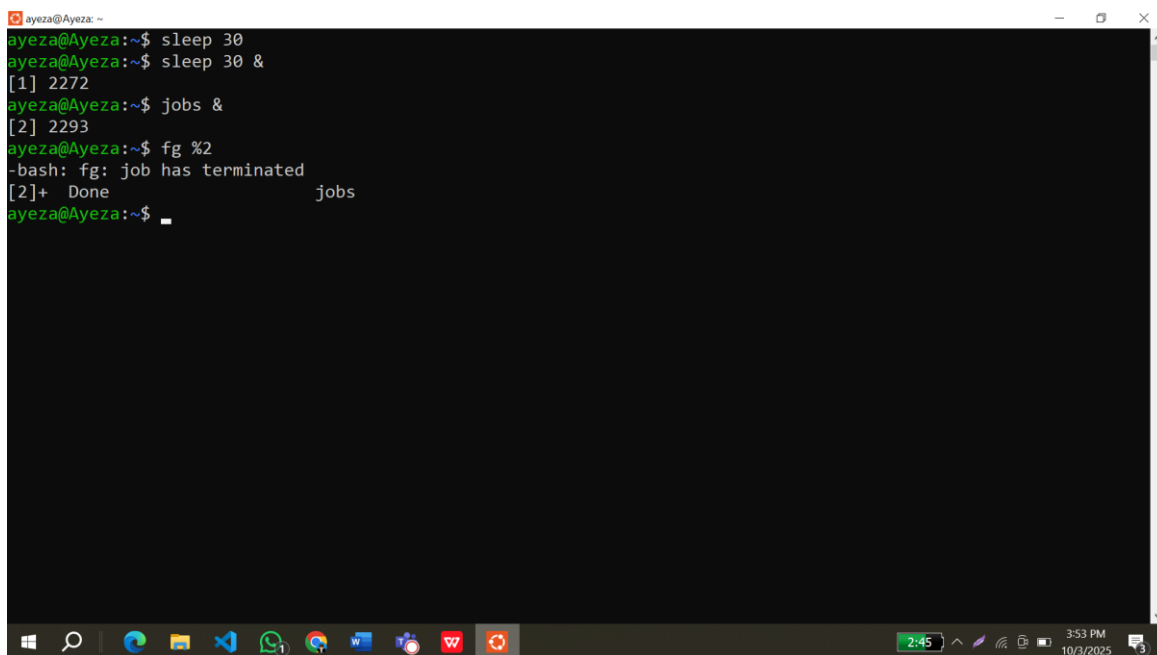
- **Bring a job to foreground:**

```
fg %1
```

%1 means job number 1 (from `jobs` output).

- **Suspend a job:** Press **Ctrl + Z** while it runs.
- **Resume suspended job in background:**

```
bg %1
```



```
ayez@Ayeza: ~$ sleep 30
ayez@Ayeza: ~$ sleep 30 &
[1] 2272
ayez@Ayeza: ~$ jobs &
[2] 2293
ayez@Ayeza: ~$ fg %2
-bash: fg: job has terminated
[2]+  Done                  jobs
ayez@Ayeza: ~$
```

## 2.4 Process Identification

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

```
pidof sleep
```

Example output: 3421 (PID of sleep command).

- **Search using `ps` and `grep`:**

```
ps -ef | grep firefox
```

```
ayez@Ayeza: ~  
ayez@Ayeza:~$ pidof sleep  
ayez@Ayeza:~$ ps -ef | grep firefox  
ayez      2519      310  0 15:54 pts/0    00:00:00 grep --color=auto firefox  
ayez@Ayeza:~$
```

## 2.5 Killing Processes

- Kill by PID:

```
kill -9 3421
```

- `-9` → force kill (SIGKILL).

- Kill all processes by name:

```
killall sleep
```

### Practice Task

1. Run an infinite process:

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

( `yes` prints “y” forever; redirected to `/dev/null` to hide output).

2. Find it with:

3. Kill it with:

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

---

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

### 3. C Programs on Processes

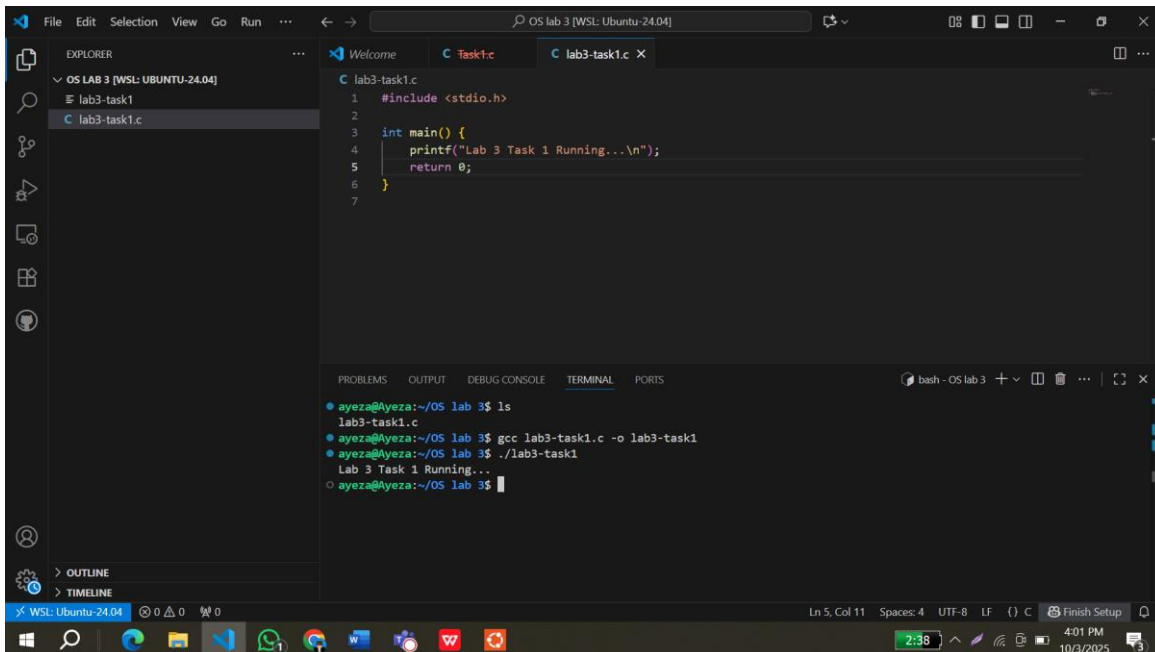
#### Program 1: Print PID and PPID

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

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

- `#include <unistd.h>` → contains process-related functions like `getpid()` and `getppid()`.
- `getpid()` → returns the unique **process ID** of the current process.
- `getppid()` → returns the **parent's PID**.
- Every process in Linux has a parent (except the very first process, usually `init` or `systemd`).

Run and compare with `ps -ef`.



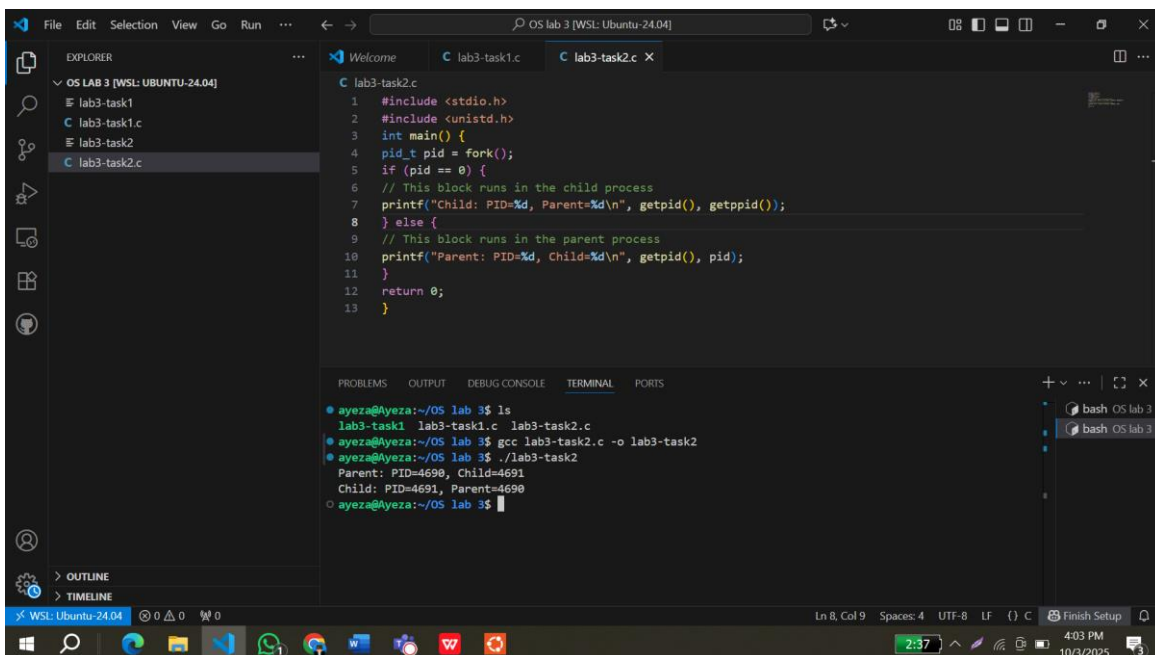
## Program 2: Fork – Creating Child Process

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

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;
}
```

- `fork()` creates a new process by duplicating the current one.
- Return value of `fork()` :
  - 0 → you are inside the **child** process.
  - Positive number (child PID) → you are in the **parent** process.
- After `fork()` , both parent and child run **the same code**, but in different branches of the **if**.



The screenshot shows a Visual Studio Code window with the following components:

- EXPLORER:** Lists files in the 'OS LAB 3 [WSL: UBUNTU-24.04]' workspace, including 'lab3-task1.c' and 'lab3-task2.c'.
- EDITOR:** Displays the source code for 'lab3-task2.c', which is identical to the code provided in the first block.
- TERMINAL:** Shows the execution of the program. The user runs 'ls' and 'gcc lab3-task2.c -o lab3-task2'. Then, they run 'lab3-task2', which produces the following output:

```
Parent: PID=4690, Child=4691
Child: PID=4691, Parent=4690
```

The status bar at the bottom indicates the file is at 'Ln 8, Col 9' with 'Spaces: 4' and 'UTF-8' encoding.



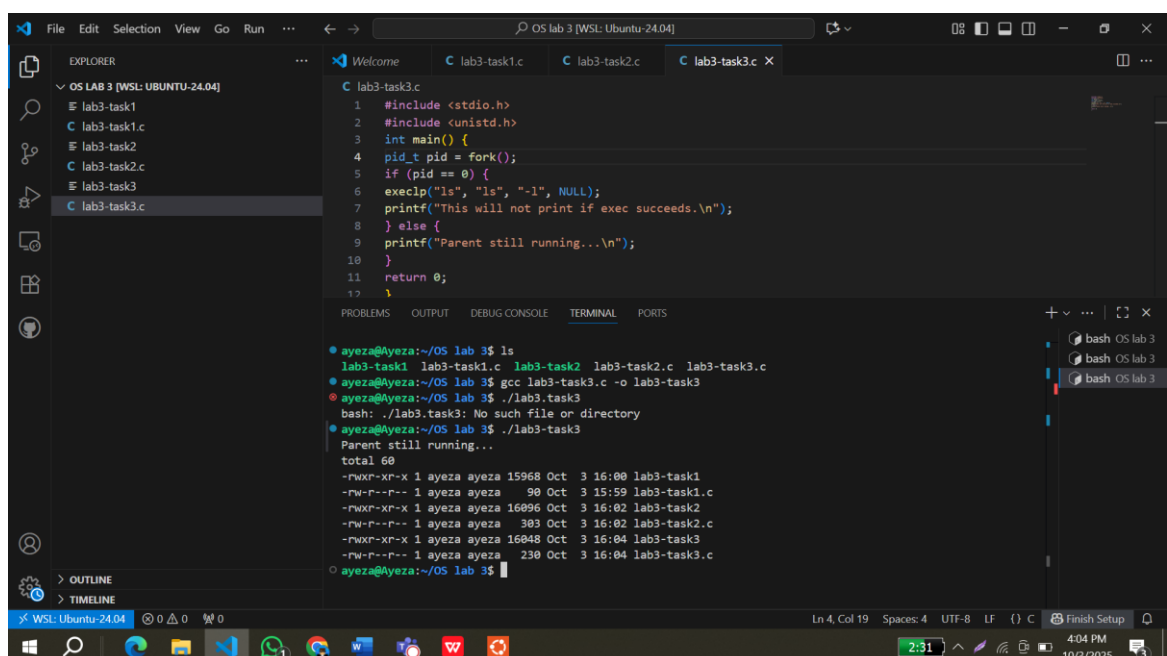
## Program 3: Exec1 – Replacing a Process

```
#include <stdio.h>
#include <unistd.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 {
        printf("Parent still running...\n");
    }
    return 0;
}
```

- `fork()` → creates child.
  - In the child:
    - `execlp("ls", "ls", "-l", NULL);`
      - Replaces the **current process image** with the `ls` program.
      - First `"ls"` = name of the program, second `"ls"` = argument 0 (how program sees itself).
      - `"-l"` = argument for `ls`.
      - `NULl` marks end of arguments.
  - Parent is unaffected and continues normally.
- After `exec()`, the child **no longer runs our C code** – it becomes `ls`.



The screenshot shows the Visual Studio Code interface with the file `lab3-task3.c` open. The code is as follows:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 int main() {
4     pid_t pid = fork();
5     if (pid == 0) {
6         execlp("ls", "ls", "-l", NULL);
7         printf("This will not print if exec succeeds.\n");
8     } else {
9         printf("Parent still running...\n");
10    }
11    return 0;
12 }
```

The terminal output shows the execution of the program:

```
ayeza@ayeza:~/OS lab 3$ ls
lab3-task1  lab3-task1.c  lab3-task2  lab3-task2.c  lab3-task3.c
ayeza@ayeza:~/OS lab 3$ gcc lab3-task3.c -o lab3-task3
ayeza@ayeza:~/OS lab 3$ ./lab3-task3
bash: ./lab3-task3: No such file or directory
ayeza@ayeza:~/OS lab 3$ ./lab3-task3
Parent still running...
total 68
-rwxr-xr-x 1 ayeza ayeza 15968 Oct 3 16:00 lab3-task1
-rw-r--r-- 1 ayeza ayeza 90 Oct 3 15:59 lab3-task1.c
-rwxr-xr-x 1 ayeza ayeza 16096 Oct 3 16:02 lab3-task2
-rw-r--r-- 1 ayeza ayeza 303 Oct 3 16:02 lab3-task2.c
-rwxr-xr-x 1 ayeza ayeza 16048 Oct 3 16:04 lab3-task3
-rw-r--r-- 1 ayeza ayeza 230 Oct 3 16:04 lab3-task3.c
ayeza@ayeza:~/OS lab 3$
```

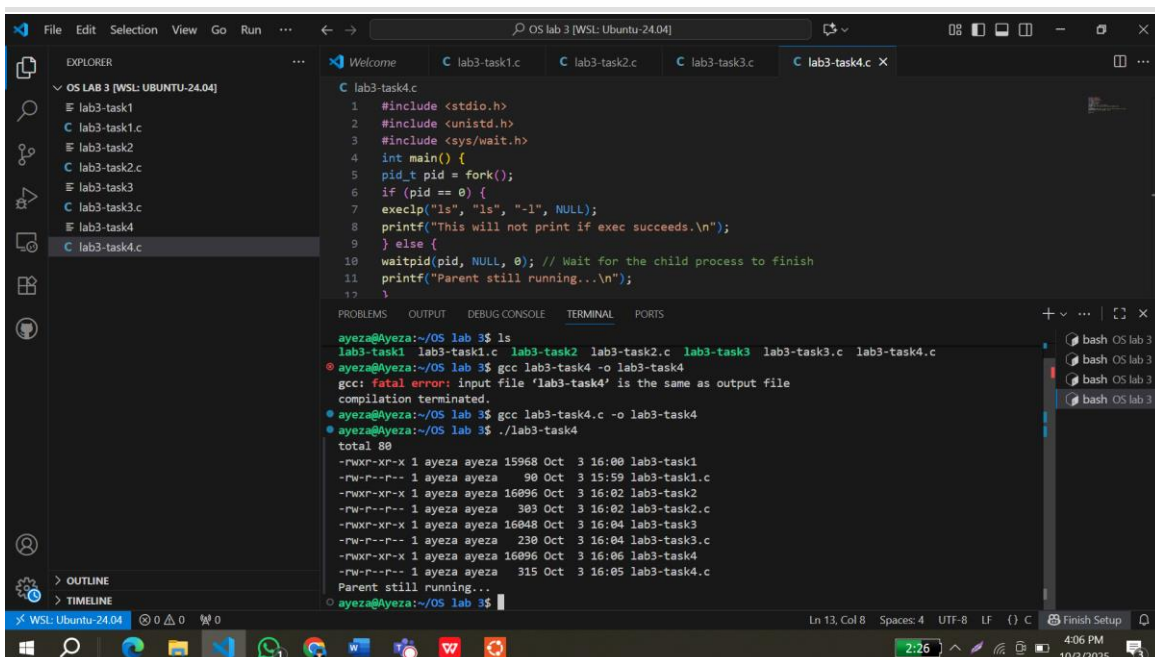
## Program 4: Wait – Synchronization

```
#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;
}
```

- - `fork()` → creates child.
- `sleep(3)` → child "works" for 3 seconds.
- `wait(NULL)` → parent pauses until child exits.
- Without `wait()`, parent may finish early and child could become a **zombie process**.



```
ayez@Ayeza:~/OS lab 3$ ls
lab3-task1  lab3-task1.c  lab3-task2  lab3-task3  lab3-task3.c  lab3-task4.c
ayez@Ayeza:~/OS lab 3$ gcc lab3-task4 -o lab3-task4
gcc: fatal error: input file 'lab3-task4' is the same as output file
compilation terminated.
ayez@Ayeza:~/OS lab 3$ gcc lab3-task4.c -o lab3-task4
ayez@Ayeza:~/OS lab 3$ ./lab3-task4
total 80
-rwxr-xr-x 1 ayeza ayeza 15968 Oct 3 16:00 lab3-task1
-rw-r--r-- 1 ayeza ayeza 90 Oct 3 15:59 lab3-task1.c
-rwxr-xr-x 1 ayeza ayeza 16096 Oct 3 16:02 lab3-task2
-rw-r--r-- 1 ayeza ayeza 303 Oct 3 16:02 lab3-task2.c
-rwxr-xr-x 1 ayeza ayeza 16048 Oct 3 16:04 lab3-task3
-rw-r--r-- 1 ayeza ayeza 238 Oct 3 16:04 lab3-task3.c
-rwxr-xr-x 1 ayeza ayeza 16096 Oct 3 16:06 lab3-task4
-rw-r--r-- 1 ayeza ayeza 315 Oct 3 16:05 lab3-task4.c
Parent still running...
```