

# 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:

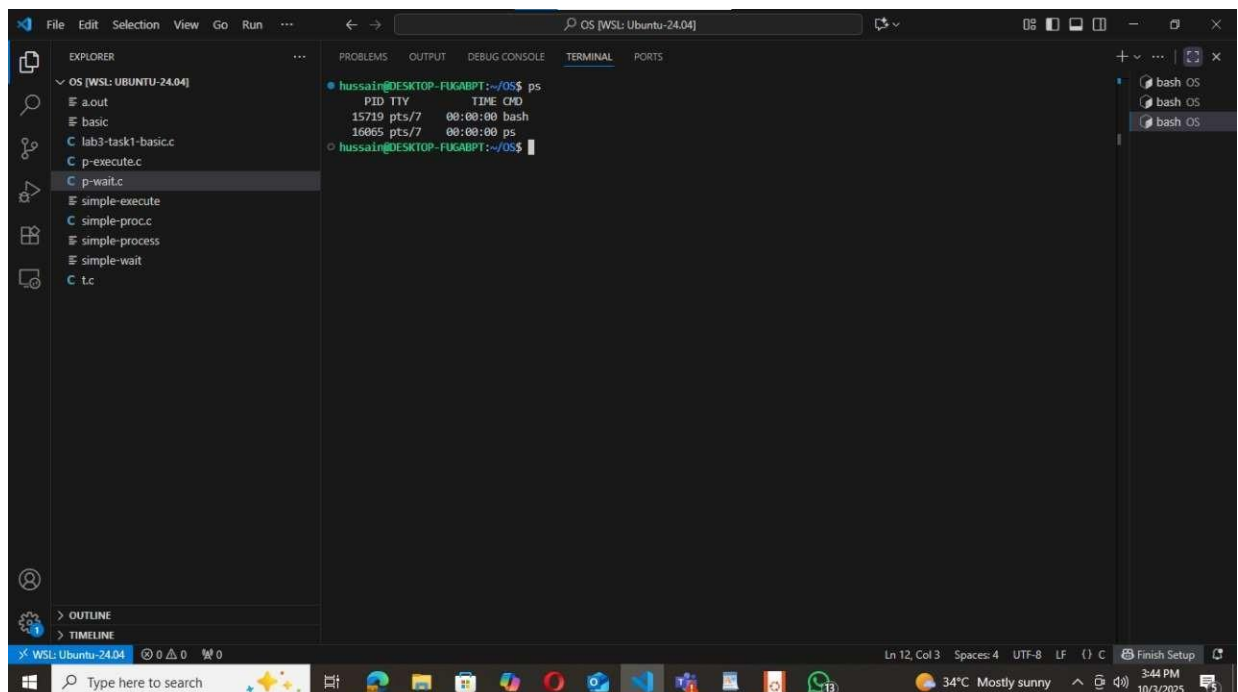
1. Learn Linux commands to monitor and manage processes.
2. Write C programs to create and observe processes.

---

## 2. Linux Process Commands

### 2.1 Viewing Processes

**ps** → Process Status



- ♦ Shows processes in the current terminal session.

```
ps
```

Output example:

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

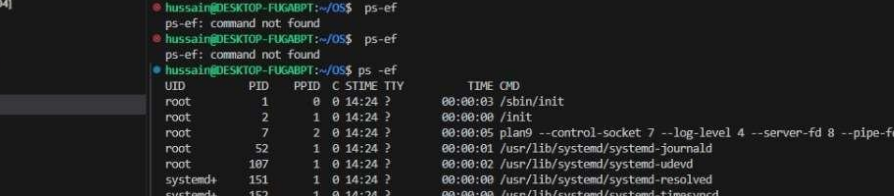
- ♦ **PID** → Process ID
- ♦ **TTY** → terminal
- ♦ **TIME** → CPU time used
- ♦ **CMD** → command name

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

```
ps -ef
```

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

Try:



```
File Edit Selection View Go Run ...
File Explorer
OS [WSL: UBUNTU-24.04]
  a.out
  basic
  lab3-task1-basic.c
  p-execute.c
  p-wait.c
  simple-execute
  simple-procc
  simple-process
  simple-wait
  tc
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
hussain@DESKTOP-FUKABPT:~/OS$ ps -ef
ps-ef: command not found
hussain@DESKTOP-FUKABPT:~/OS$ ps -ef
ps-ef: command not found
hussain@DESKTOP-FUKABPT:~/OS$ ps -ef
  UID          PID     PPID  C  STIME TTY          TIME CMD
root           1      0  0  14:24 ?        00:00:03 /sbin/init
root           2      1  0  14:24 ?        00:00:00 /init
root           7      2  0  14:24 ?        00:00:05 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --log
root          52      1  0  14:24 ?        00:00:01 /usr/lib/systemd/systemd-journald
root         107      1  0  14:24 ?        00:00:02 /usr/lib/systemd/systemd-udev
root         151      1  0  14:24 ?        00:00:00 /usr/lib/systemd/systemd-resolved
systemd+     152      1  0  14:24 ?        00:00:00 /usr/lib/systemd/systemd-timesyncd
root         161      1  0  14:24 ?        00:00:00 /usr/sbin/cron -f -P
message+    162      1  0  14:24 ?        00:00:00 @dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd
root         178      1  0  14:24 ?        00:00:00 /usr/lib/systemd/systemd-logind
root         180      1  0  14:24 ?        00:00:01 /usr/libexec/wnsl-pro-service -vv
syslog      184      1  0  14:24 ?        00:00:00 /usr/sbin/rsyslogd -n -INONE
root        189      1  0  14:24 hvdc0    00:00:00 /sbin/agetty -o -p -- \u --noclear --keep-baud - 115200,38400,9600 vt22
root        203      1  0  14:24 tty1     00:00:00 /sbin/agetty -o -p -- \u --noclear - linux
root        210      1  0  14:24 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shut
root        339      2  0  14:24 pts/1    00:00:00 /bin/login -f
hussain     387      1  0  14:24 ?        00:00:00 /usr/lib/systemd/systemd --user
hussain     388     387  0  14:24 ?        00:00:00 (sd-pam)
hussain     409     339  0  14:24 pts/1    00:00:00 -bash
root        2449      2  0  14:34 ?        00:00:00 /init
root        2450     2449  0  14:34 ?        00:00:00 /init
hussain     2451     2450  0  14:34 pts/0    00:00:00 sh -c "$VSOCODE_WSL_EXT_LOCATION/scripts/wslServer.sh" e3a5acfb517a44323
hussain     2452     2451  0  14:34 pts/0    00:00:00 sh /mnt/c/Users/JK/.vscode/extensions/ms-vscode-remote.remote-wsl-0.104
hussain     2458     2452  0  14:34 pts/0    00:00:00 sh /home/hussain/.vscode-server/bin/e3a5acfb517a443235981655413d566531
hussain     2462     2458  0  14:34 pts/0    00:00:28 /home/hussain/.vscode-server/bin/e3a5acfb517a443235981655413d56653107e
root        2473      2  0  14:34 ?        00:00:00 /init
root        2474     2473  0  14:34 ?        00:00:01 /init
hussain     2475     2474  0  14:34 pts/2    00:00:08 /home/hussain/.vscode-server/bin/e3a5acfb517a443235981655413d56653107e
root        2486      2  0  14:34 ?        00:00:00 /init
root        2489     2486  0  14:34 ?        00:00:01 /init
hussain     2490     2489  0  14:34 pts/3    00:00:07 /home/hussain/.vscode-server/bin/e3a5acfb517a443235981655413d56653107e
WSL: Ubuntu-24.04
Type here to search
3:50 PM 10/3/2025
```

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

This finds all processes related to the shell.

## 2.2 Monitoring Processes Interactively

**top** → Dynamic process viewer

top

```

hussain@DESKTOP-FUGABPT:~/OS$ top
top - 15:56:08 up 1:32, 1 user, load average: 0.14, 0.16, 0.11
Tasks: 42 total, 1 running, 41 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.6 us, 1.8 sy, 0.0 ni, 95.1 id, 0.1 wa, 0.0 hi, 0.5 si, 0.0 st
MiB Mem : 1854.2 total, 507.1 free, 703.9 used, 724.6 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 1150.3 avail Mem

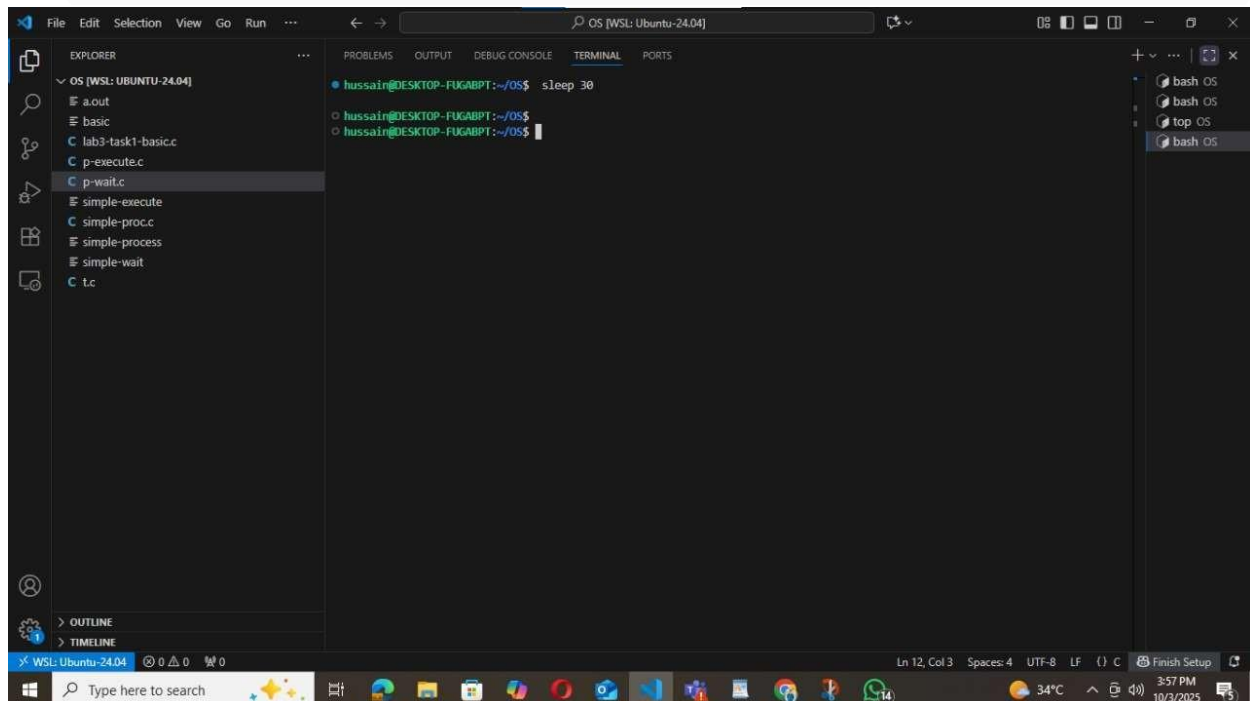
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 3245 hussain   20   0 1153632 82584 48256 S   5.6   4.3   0:32.03 node
 2526 hussain   20   0 31.9g 152520 54016 S   2.6   8.0   2:59.51 node
 2462 hussain   20   0 11.3g 104072 53248 S   0.7   5.5   0:30.51 node
 2475 hussain   20   0 1021296 64532 44032 S   0.7   3.4   0:09.52 node
 2474 root       20   0 3688    1040    896 S   0.3   0.1   0:01.61 Relay(2475)
 2491 hussain   20   0 1262000 59020 47072 S   0.3   3.2   0:02.35 node
18606 hussain   20   0 9284    5632   3456 R   0.3   0.3   0:00.07 top
    1 root       20   0 21664   12584  9304 S   0.7   0.3   0:03.28 systemd
    2 root       20   0 3072    1792   1792 S   0.0   0.1   0:00.11 init-systemd(Ub
    7 root       20   0 3120    1928   1792 S   0.0   0.1   0:05.46 init
   52 root      19  -1 66748  16916 16020 S   0.0   0.9   0:02.06 systemd-journal
  187 root       20   0 25272   6272   4864 S   0.0   0.3   0:02.10 systemd-udevd
  151 systemd+   20   0 21456  12672 10496 S   0.0   0.7   0:00.55 systemd-resolve
  152 systemd+   20   0 91024   7680   6784 S   0.0   0.4   0:00.75 systemd-timesyn
  161 root       20   0 4236    2432   2304 S   0.0   0.1   0:00.50 cron
  162 message+   20   0 9624   4608   4224 S   0.0   0.2   0:00.59 dbus-daemon
  178 root       20   0 17964   8448   7552 S   0.0   0.4   0:00.45 systemd-logind
  180 root       20   0 1829828 13568 11136 S   0.0   0.7   0:01.26 wsl-pro-service
  184 syslog     20   0 222508   5504   4352 S   0.0   0.3   0:00.44 rsyslogd
  189 root       20   0 3160    1920   1792 S   0.0   0.1   0:00.02 agetty
  203 root       20   0 3116    1664   1664 S   0.0   0.1   0:00.00 agetty
  210 root       20   0 107028 22272 13184 S   0.0   1.2   0:00.37 unattended-upgr
  339 root       20   0 6820    4096   3584 S   0.0   0.2   0:00.02 login
  387 hussain   20   0 20312  10880  9088 S   0.0   0.6   0:00.37 systemd
  388 hussain   20   0 21152   3520   1792 S   0.0   0.2   0:00.00 (sd-pam)
  409 hussain   20   0 6072    4992   3456 S   0.0   0.3   0:00.16 bash
 2449 root       20   0 3076    1032    896 S   0.0   0.1   0:00.00 SessionLeader
 2450 root       20   0 3092    1036    896 S   0.0   0.1   0:00.00 Relay(2451)
 2451 hussain   20   0 2800    1664   1664 S   0.0   0.1   0:00.07 sh
  
```

- 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.

## 2.3 Foreground and Background Jobs

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

sleep 30

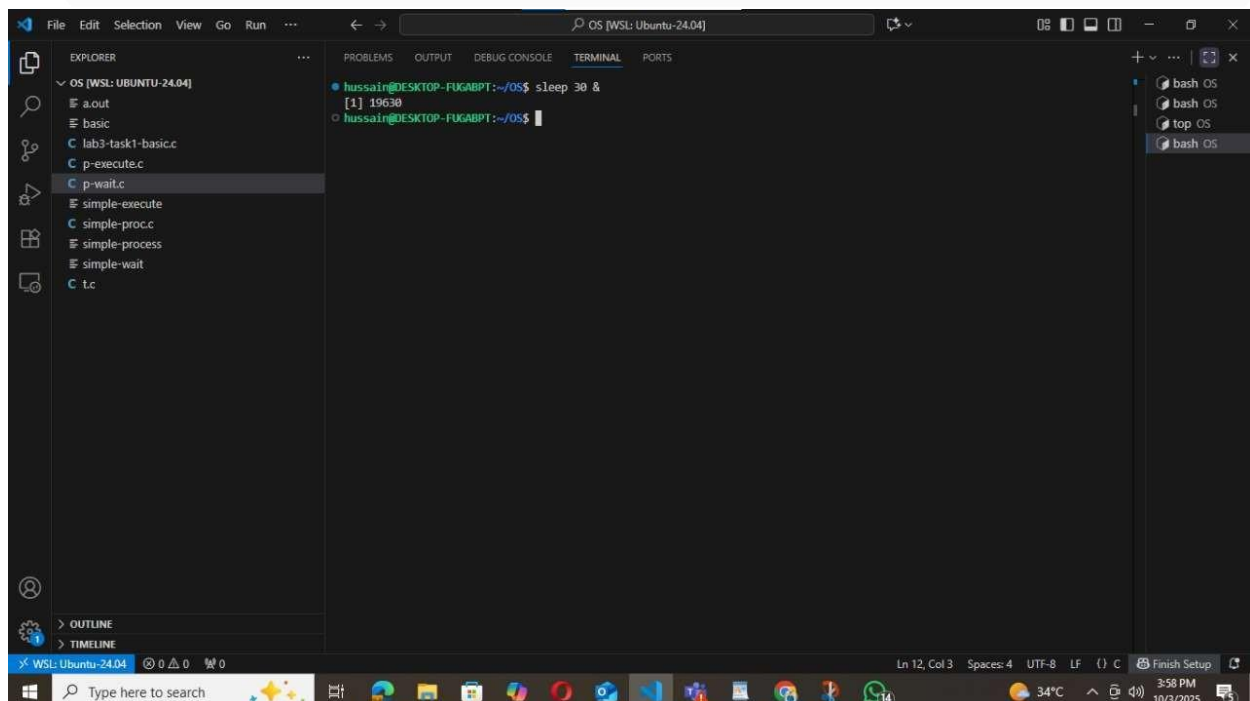


```
hussain@DESKTOP-FUGABPT:~/OS$ sleep 30
hussain@DESKTOP-FUGABPT:~/OS$
```

→ You cannot type new commands until it finishes.

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

sleep 30 &



```
hussain@DESKTOP-FUGABPT:~/OS$ sleep 30 &
[1] 19630
hussain@DESKTOP-FUGABPT:~/OS$
```

→ Terminal is free while the command runs.

♦ Check background jobs:

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
```

---

## 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
```

---

## 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 &
```

- 2.

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

3.

`/dev/null`

4. Find it with:



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

5. Kill it with:

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

---

## 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` .

```
File Edit Selection View Go Run ...
C Lab3-task1.c
1 #include <stdio.h>
2 #include <unistd.h>
3 int main()
4 printf("My PID: %d\\n", getpid());
5 printf("My Parent PID: %d\\n", getppid());
6 return 0;
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ gcc Lab3-task1.c
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ ./a.out
My PID: 1145
My Parent PID: 787
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ gcc Lab3-task2.c
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ ./simple-process
bash: ./simple-process: No such file or directory
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ gcc Lab3-task2.c -o simple-process
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$ ./simple-process
Parent: PID=1896, Child=1897
Child: PID=1897, Parent=1896
hussain@DESKTOP-NNR4PJ:~/OS/Lab3$

WSL: Ubuntu-24.04 0 0 0
Type here to search
21°C Clear
10:59 AM
10/24/2025
```

## 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);
    }
}
```

FileEditSelectionViewGoRun...Lab3 [WSL: Ubuntu-24.04]

EXPLORER

LAB3 [WSL: UBUNTU-24.04]

a.outLab3-task1.cLab3-task2.cLab3-task3.cLab3-task4.csimple-process

Lab3-task2.c 1

```
1 #include <stdio.h>
2 #include <unistd.h>
3 int main() {
4     pid_t pid = fork();
5     if (pid == 0) {
6         // This block runs in the child process
7         printf("Child: PID=%d, Parent=%d\n", getpid(), getppid());
8     } else {
9         // This block runs in the parent process
10        printf("Parent: PID=%d, Child=%d\n", getpid(), pid);
11    }
12    return 0;
13 }
```

CHAT

Ask about your code

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Lab3-task2.c

Add context (#), extensions (@), commands

Ask

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

bash - Lab3

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ gcc Lab3-task1.c

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ ./a.out

My PID: 1145

My Parent PID: 787

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ gcc Lab3-task2.c

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ ./simple-process

bash: ./simple-process: No such file or directory

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ gcc Lab3-task2.c -o simple-process

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$ ./simple-process

Parent: PID-1896, Child-1897

Child: PID-1897, Parent-1896

hussain@DESKTOP-MNNR4MJ:~/OS/Lab3\$

WSL: Ubuntu-24.04

1 0 0

Ln 1, Col 1

Spaces: 4

UTF-8

LF

{ } C

Signed out

Linux

Type here to search

21°C

10:59 AM

10/24/2025

```

    }
    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** .

---

## Program 3: Execl – 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.
- ♦ After `exec()` , the child **no longer runs our C code** – it becomes `ls` .

- ◆ Parent is unaffected and continues normally.

The screenshot shows the Visual Studio Code editor with a C program named `Lab3-task3.c`. The program uses `fork()` to create a child process. The parent process prints "Parent still running..." and then returns 0. The child process prints "This will not print if exec succeeds.\n" and then returns 0. The terminal output shows the parent process running and the child process executing `ls`.

```
#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;
}
```

Terminal Output:

```
total 48
-rw-r--r-- 1 hussain hussain 146 Oct 3 02:47 Lab3-task1.c
-rw-r--r-- 1 hussain hussain 315 Oct 3 03:49 Lab3-task2.c
-rw-r--r-- 1 hussain hussain 242 Oct 3 03:49 Lab3-task3.c
-rwxr-xr-x 1 hussain hussain 16048 Oct 24 11:01 a.out
-rwxr-xr-x 1 hussain hussain 16096 Oct 24 10:58 simple-process
hussain@DESKTOP-NNNR4M3:~/OS/Lab3$ gcc Lab3-task3.c -o without-wait
hussain@DESKTOP-NNNR4M3:~/OS/Lab3$ ./without-wait
Parent still running...
total 64
-rw-r--r-- 1 hussain hussain 146 Oct 3 02:47 Lab3-task1.c
-rw-r--r-- 1 hussain hussain 315 Oct 3 03:49 Lab3-task2.c
```

## Program 4: Wait – Synchronization

```
    } else {  
  
    }  
}
```

`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**.

FileEditSelectionViewGoRun...Lab3 [WSL: Ubuntu-24.04]

EXPLORER

Lab3 [WSL: UBUNTU-24.04]

- a.out
- Lab3-task1.c
- Lab3-task2.c
- Lab3-task3.c
- Lab3-task4.c
- simple-process
- wait
- without-wait

Lab3-task4.c

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4 int main() {
5     pid_t pid = fork();
6     if (pid == 0) {
7         execlp("ls", "ls", "-l", NULL);
8         printf("This will not print if exec succeeds.\n");
9     } else {
10        waitpid(pid, NULL, 0); // Wait for the child process to
11        printf("Parent still running...\n");
12    }
13    return 0;
14 }
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

bash - Lab3

```
hussain@DESKTOP-MNNR4MM:~/OS/Lab3$ ./wait
total 80
-rw-r--r-- 1 hussain hussain 146 Oct 3 02:47 Lab3-task1.c
-rw-r--r-- 1 hussain hussain 315 Oct 3 03:49 Lab3-task2.c
-rw-r--r-- 1 hussain hussain 242 Oct 24 11:04 Lab3-task3.c
-rw-r--r-- 1 hussain hussain 328 Oct 3 03:52 Lab3-task4.c
-rwxr-xr-x 1 hussain hussain 10096 Oct 24 11:05 a.out
-rwxr-xr-x 1 hussain hussain 10096 Oct 24 10:58 simple-process
-rwxr-xr-x 1 hussain hussain 10096 Oct 24 11:05 wait
-rwxr-xr-x 1 hussain hussain 16048 Oct 24 11:01 without-wait
Parent still running...
```

CHAT

Ask about your code

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Lab3-task4.c

Add context (#), extensions (@), commands

Ask

WSL: Ubuntu-24.04

0 0 0

Activating Extensions...

Ln 9, Col 10

Spaces: 4

UTF-8

LF

Signed out

Linux

21°C

11:06 AM

10/24/2025

