

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:

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

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

```
shanzai209@DESKTOP-L1S8TT: ~
shanzai209@DESKTOP-L1S8TT:~$ ps
  PID TTY          TIME CMD
 25216 pts/12    00:00:00 bash
 22672 pts/12    00:00:00 ps
shanzai209@DESKTOP-L1S8TT:~$ ps -ef
UID          PID    PPID  C TIME   TTY          TIME CMD
root           1        0  0 18:25 ?        00:00:02 /sbin/init
root           2        1  0 18:25 ?        00:00:00 /init
root           7        2  0 18:25 ?        00:00:00 plan9 --control-socket 7 --log-level 4 --server-fd 8root
root        108        1  0 18:25 ?        00:00:01 /usr/lib/systemd/systemd-udevd
systemd+    124        1  0 18:25 ?        00:00:00 /usr/lib/systemd/systemd-resolved
systemd+    125        1  0 18:25 ?        00:00:00 /usr/lib/systemd/systemd-timesyncd
root        175        1  0 18:25 ?        00:00:00 /usr/sbin/cron -f -p
message+   176        1  0 18:25 ?        00:00:00 @dbus-daemon --system --address=systemd: --nofork --root
root        191        1  0 18:25 ?        00:00:00 /usr/libexec/ssl-proc-service -vv
root       194        1  0 18:25 hvc0    00:00:00 /sbin/agetty -o -p -- \u --noclear --keep-baud - 115root
syslog     215        1  0 18:25 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root       223        1  0 18:25 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unatroot
shanzai+   370        1  0 18:25 ?        00:00:00 /usr/lib/systemd/systemd --user
shanzai+   371       370  0 18:25 ?        00:00:00 (sd-pam)
shanzai+   397       324  0 18:25 pts/1    00:00:00 -bash
root       446        2  0 18:25 ?        00:00:00 /init
root       447       446  0 18:25 ?        00:00:00 /init
shanzai+   448       447  0 18:25 pts/0    00:00:00 sh -c "VSCODE_WSL_EXT_LOCATION/scripts/wslServer.shshanzai+
shanzai+   455       449  0 18:25 pts/0    00:00:00 sh /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4shanzai+
shanzai+   997       461  0 18:27 pts/0    00:00:37 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4shanzai+
root      1466       461  2 18:29 pts/0    00:04:29 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
root      1510      1509  0 18:29 ?        00:00:00 /init
shanzai+  1511      1510  0 18:29 pts/8    00:00:00 /bin/sh -c cd "/home/shanzai209/Lab06" && /bin/sh
shanzai+  1512      1511  0 18:29 pts/8    00:00:00 /bin/sh
shanzai+  1522      1512  0 18:29 pts/8    00:00:00 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4shanzai+
shanzai+  2412       997  0 18:33 pts/5    00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-serverpolkitd
shanzai+  22591      997  0 20:20 pts/6    00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-servershanzai+
shanzai+  24262      461  0 20:26 pts/0    00:00:22 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
shanzai+  24311      24310  0 20:26 ?        00:00:00 /init
shanzai+  24312      24311  0 20:26 pts/10   00:00:00 /bin/sh -c cd "/home/shanzai209/Labs_05" && /bin/sh
shanzai+  24323      24313  0 20:26 pts/10   00:00:00 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4shanzai+
shanzai+  24366       997  0 20:26 pts/11   00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-serverroot
root      26067        2  0 20:44 ?        00:00:00 /init
root      26069      26067  0 20:44 ?        00:00:00 /init
shanzai+  26072      26069  0 20:44 pts/2    00:00:01 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
root      26080      26079  0 20:44 ?        00:00:00 /init
shanzai+  26081      26080  0 20:44 pts/3    00:00:02 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
root      26089      26088  0 20:44 ?        00:00:00 /init
shanzai+  26090      26089  0 20:44 pts/4    00:00:00 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
root      26098      26097  0 20:44 ?        00:00:00 /init
shanzai+  26099      26098  0 20:44 pts/9    00:00:00 /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4root
shanzai209@DESKTOP-L1S8TT:~$
```

```
shanzai209@DESKTOP-L58TUD:~$ ps -ef | grep bash
shanzai+ 397 324 0 18:25 pts/1 00:00:00 -bash
shanzai+ 2412 997 0 18:33 pts/5 00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4a8e4d7e040d2625abbf7f084/out/vs/workbench/contrib/terminal/common/scripts/shellIntegr
shanzai+ 22591 997 0 20:20 pts/6 00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4a8e4d7e040d2625abbf7f084/out/vs/workbench/contrib/terminal/common/scripts/shellIntegr
shanzai+ 24366 997 0 20:26 pts/11 00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4a8e4d7e040d2625abbf7f084/out/vs/workbench/contrib/terminal/common/scripts/shellIntegr
shanzai+ 26216 26212 0 20:47 pts/12 00:00:00 -bash
shanzai+ 27884 997 0 21:04 pts/13 00:00:00 /bin/bash --init-file /home/shanzai209/.vscode-server/bin/7d842fb85a0275a4a8e4d7e040d2625abbf7f084/out/vs/workbench/contrib/terminal/common/scripts/shellIntegr
shanzai+ 27698 26216 0 21:11 pts/12 00:00:00 grep --color=auto bash
shanzai@DESKTOP-L58TUD:~$ top
top - 21:11:18 up 2:47, 1 user, load average: 0.01, 0.01, 0.00
Tasks: 64 total, 1 running, 63 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2 us, 0.3 sy, 0.0 ni, 99.2 id, 0.1 wa, 0.0 hi, 0.1 si, 0.0 st
Mem: 3896.2 total, 2883.5 free, 982.0 used, 211.4 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used, 2868.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR   S  %CPU  %MEM     TIME+ COMMAND
 1466 shanzai+  20   0  31.3g 158768 68092  S   2.0   4.0   4:29.88 node
 188 root      20   0  25272   6144   6864  S   0.3   0.2   0:01.92 systemd-udev
 461 shanzai+  20   0  11.3g 150192 51328  S   0.3   3.8   0:41.80 node
 997 shanzai+  20   0 1097476 82312 47360  S   0.3   2.1   0:37.61 node
24262 shanzai+  20   0  31.9g 146116 55424  S   0.3   3.7   0:23.07 node
27699 shanzai+  20   0   9388   5594   3328  R   0.3   0.1   0:00.04 top
    1 root      20   0  21780 12340  9140  S   0.0   0.3   0:02.14 systemd
    2 root      20   0  3072   1664   1664  S   0.0   0.0   0:00.04 init-systemd(Ub
    7 root      20   0  3528   2236   1792  S   0.0   0.1   0:06.57 init
   59 root      19  -1 50436 15616 14720  S   0.0   0.4   0:01.41 systemd-journal
  124 systemd+  20   0 21456 12416 10240  S   0.0   0.3   0:00.20 systemd-resolve
  125 systemd+  20   0 91024 7424  6656  S   0.0   0.2   0:00.38 systemd-timesyn
  175 root      20   0  4236   2568   2432  S   0.0   0.1   0:00.04 cron
  176 message+  20   0   9632   4726   4352  S   0.0   0.1   0:00.44 dbus-daemon
  189 root      20   0 17964  8192  7296  S   0.0   0.2   0:00.28 systemd-logind
  191 root      20   0 1755840 12800 10112  S   0.0   0.3   0:00.67 wsl-pro-service
  194 root      20   0  3160  1920  1792  S   0.0   0.0   0:00.01 agetty
  214 root      20   0  3116  1792  1664  S   0.0   0.0   0:00.00 agetty
  215 syslog    20   0 222588 5632  4352  S   0.0   0.1   0:00.46 rsyslogd
  223 root      20   0 107028 22272 13056  S   0.0   0.6   0:00.18 unattended-upgr
  324 root      20   0  6664  4352  3712  S   0.0   0.1   0:00.01 login
  378 shanzai+  20   0 20188 11800  9216  S   0.0   0.3   0:00.39 systemd
  371 shanzai+  20   0 21156 3520  1792  S   0.0   0.1   0:00.00 (sd-pam)
  397 shanzai+  20   0  6072  4992  3456  S   0.0   0.1   0:00.05 bash
  446 root      20   0  3076   908   768  S   0.0   0.0   0:00.00 SessionLeader
  447 root      20   0  3092  1160  1024  S   0.0   0.0   0:00.00 Relay(446)
  448 shanzai+  20   0  2800  1536  1536  S   0.0   0.0   0:00.00 sh
  449 shanzai+  20   0  2800  1664  1664  S   0.0   0.0   0:00.00 sh
  455 shanzai+  20   0  2800  1664  1664  S   0.0   0.0   0:00.00 sh
```

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.

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
```

2.4 Process Identification

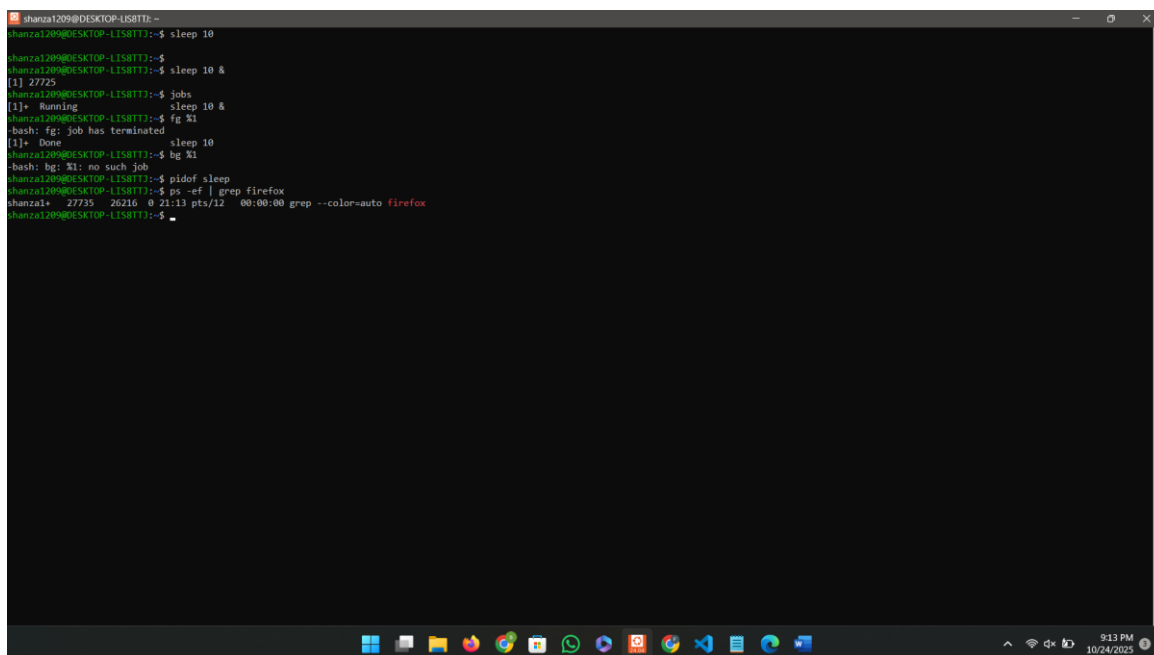
- Get PID of a process by name:

```
pidof sleep
```

Example output: 3421 (PID of sleep command).

- Search using `ps` and `grep`:

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



```
shanzai209@DESKTOP-1158T71:~$ sleep 10
shanzai209@DESKTOP-1158T71:~$
shanzai209@DESKTOP-1158T71:~$ sleep 10 &
[1] 27725
shanzai209@DESKTOP-1158T71:~$ jobs
[1]+  Running                  sleep 10 &
shanzai209@DESKTOP-1158T71:~$ fg %1
-bash: fg: job has terminated
[1]+  Done                      sleep 10
shanzai209@DESKTOP-1158T71:~$ bg %1
-bash: bg: %1: no such job
shanzai209@DESKTOP-1158T71:~$ pidof sleep
shanzai209@DESKTOP-1158T71:~$ ps -ef | grep firefox
shanzai+  27735  26216  0 21:13 pts/12    00:00:00 grep --color=auto firefox
shanzai209@DESKTOP-1158T71:~$
```

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:

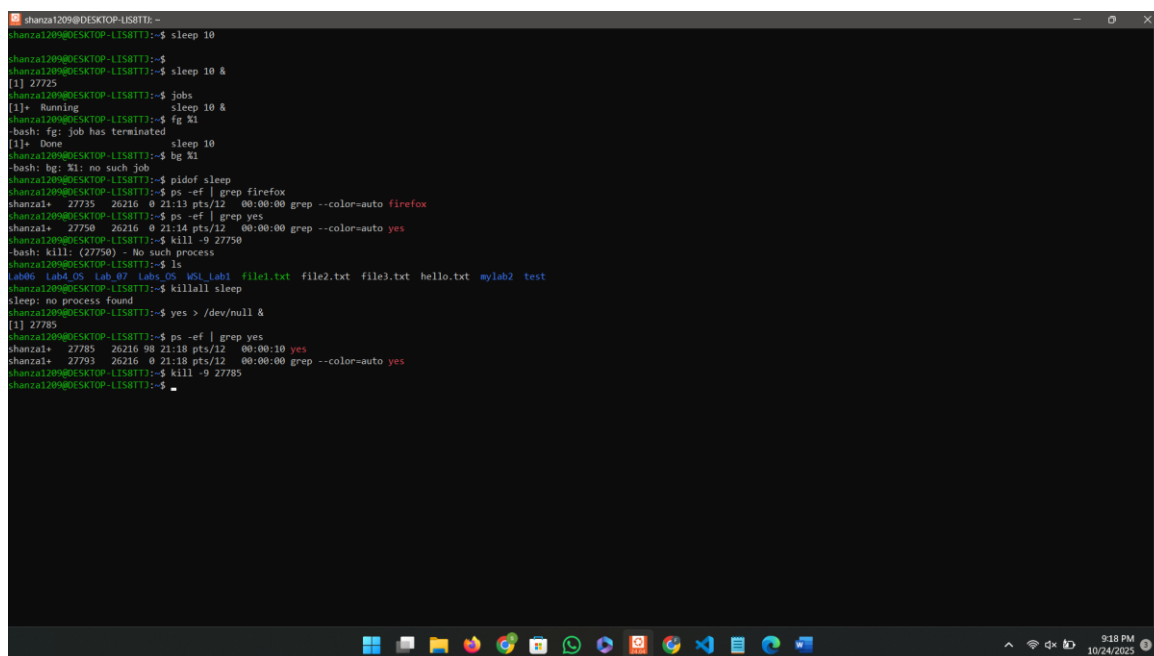
```
yes > /dev/null &
```

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

2. Find it with:

3. Kill it with:

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



```
shanza1209@DESKTOP-LS8T7L:~$ sleep 10
shanza1209@DESKTOP-LS8T7L:~$
shanza1209@DESKTOP-LS8T7L:~$ sleep 10 &
[1] 27725
shanza1209@DESKTOP-LS8T7L:~$ jobs
[1]+  Running                  sleep 10 &
shanza1209@DESKTOP-LS8T7L:~$ fg %1
-bash: fg: job has terminated
[1]+  Done                      sleep 10
shanza1209@DESKTOP-LS8T7L:~$ bg %1
-bash: bg: %1: no such job
shanza1209@DESKTOP-LS8T7L:~$ pidof sleep
shanza1209@DESKTOP-LS8T7L:~$ ps -ef | grep firefox
shanza1+  27735  26216  0 21:13 pts/12   00:00:00 grep --color=auto firefox
shanza1209@DESKTOP-LS8T7L:~$ ps -ef | grep yes
shanza1+  27750  26216  0 21:14 pts/12   00:00:00 grep --color=auto yes
shanza1209@DESKTOP-LS8T7L:~$ kill -9 27750
-bash: kill: (27750) - No such process
shanza1209@DESKTOP-LS8T7L:~$ ls
Lab06  Lab4_05  Lab_07  Labs_05  WS1_Lab1  file1.txt  file2.txt  file3.txt  hello.txt  mylab2  test
shanza1209@DESKTOP-LS8T7L:~$ killall sleep
sleep: no process found
shanza1209@DESKTOP-LS8T7L:~$ yes > /dev/null &
[1] 27785
shanza1209@DESKTOP-LS8T7L:~$ ps -ef | grep yes
shanza1+  27785  26216  98 21:18 pts/12   00:00:10 yes
shanza1+  27793  26216  0 21:18 pts/12   00:00:00 grep --color=auto yes
shanza1209@DESKTOP-LS8T7L:~$ kill -9 27785
shanza1209@DESKTOP-LS8T7L:~$
```

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

3. C Programs on Processes

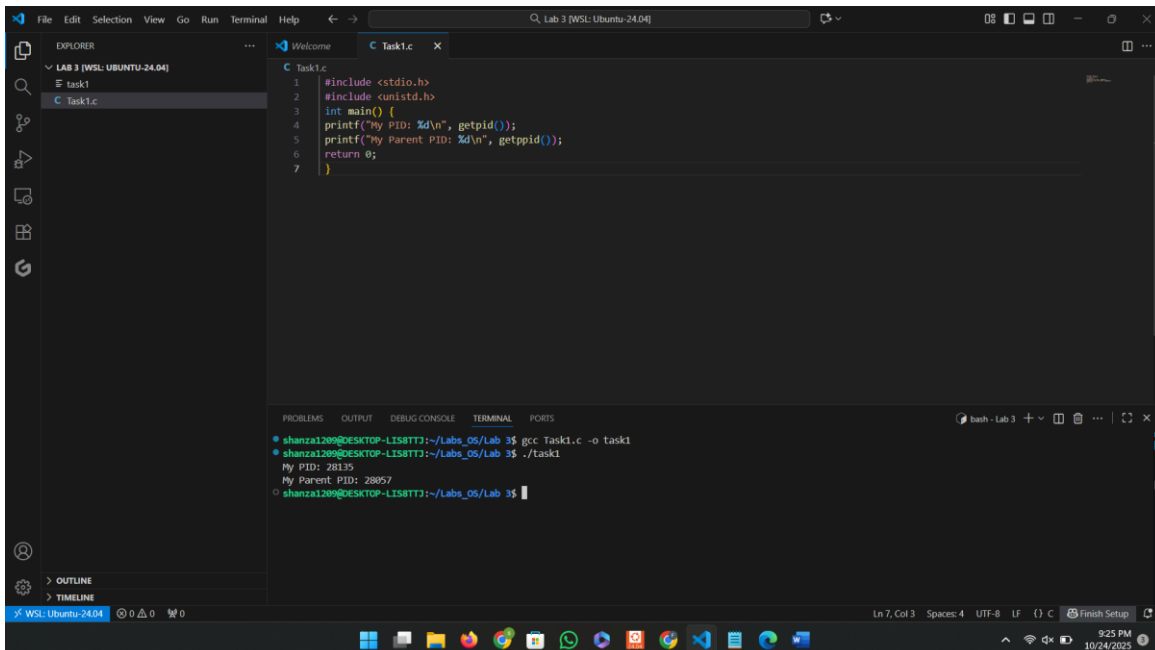
Program 1: Print PID and PPID

```
#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`.



The screenshot shows a Visual Studio Code window with a C program named `Task1.c` and its execution output in the terminal. The program prints the PID and PPID of the current process and its parent.

```
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 }
```

The terminal output shows the execution of the program:

```
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$ gcc Task1.c -o task1
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$ ./task1
My PID: 28135
My Parent PID: 28057
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$
```

Program 2: Fork – Creating Child Process

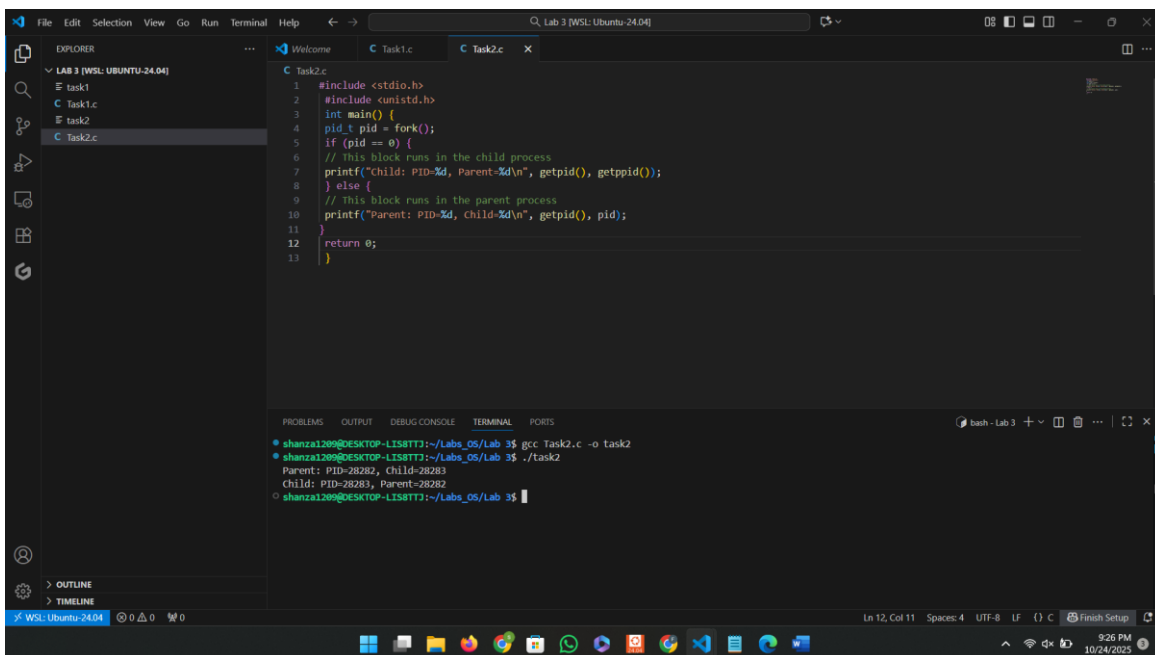
```
#include <stdio.h>
#include <unistd.h>

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**.



```
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 }
```

```
shanzai209@DESKTOP-LIS8TT3:~/Labs_O5/Lab 3$ gcc Task2.c -o task2
shanzai209@DESKTOP-LIS8TT3:~/Labs_O5/Lab 3$ ./task2
Parent: PID=28282, Child=28283
Child: PID=28283, Parent=28282
shanzai209@DESKTOP-LIS8TT3:~/Labs_O5/Lab 3$
```

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.
- 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 C program code in the editor and the terminal output below it. The code is the same as shown in the previous block. The terminal output shows the execution of the program, with the parent process still running and the child process replaced by the `ls` command.

```
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$ gcc Task3.c -o task3
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$ ./task3
Parent still running...
total 60
-rw-r--r-- 1 shanzai209 shanzai209 146 Oct 24 21:25 Task1.c
-rw-r--r-- 1 shanzai209 shanzai209 314 Oct 24 21:26 Task2.c
-rw-r--r-- 1 shanzai209 shanzai209 244 Oct 24 21:28 Task3.c
-rwxr-xr-x 1 shanzai209 shanzai209 16048 Oct 24 21:25 task1
-rwxr-xr-x 1 shanzai209 shanzai209 16088 Oct 24 21:26 task2
-rwxr-xr-x 1 shanzai209 shanzai209 16048 Oct 24 21:28 task3
shanzai209@DESKTOP-LIS8TTJ:~/Labs_OS/Lab 3$
```

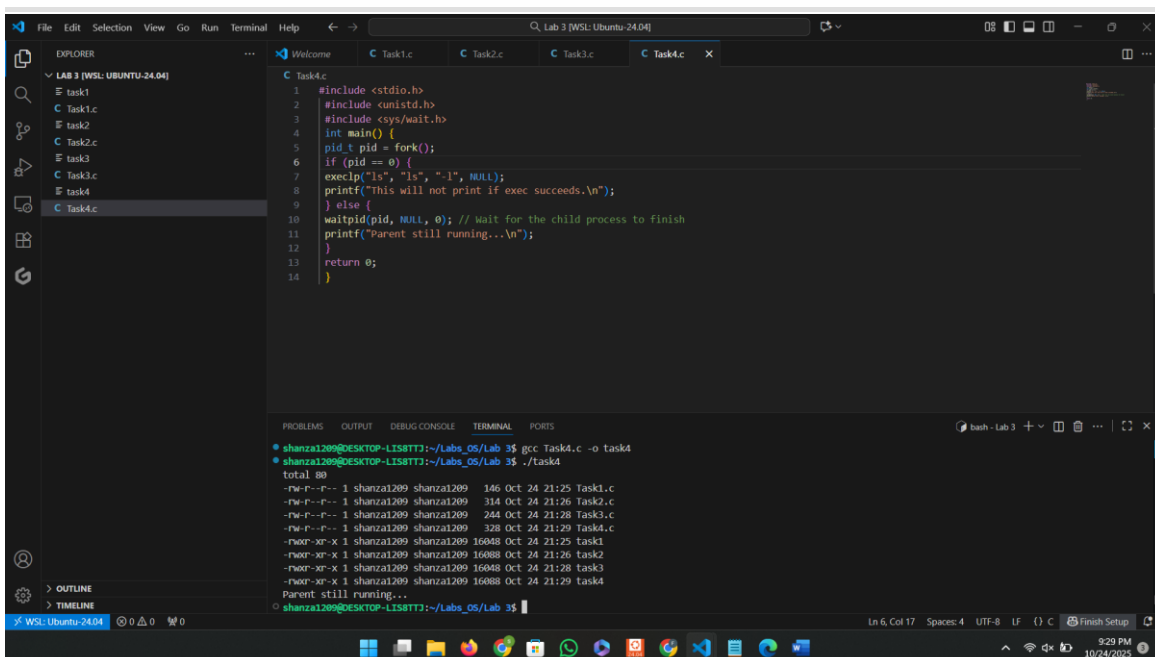

Program 4: Wait – Synchronization

```
#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**.



The screenshot shows a Visual Studio Code editor window with a C program named `Task4.c` and its execution output in the terminal. The program is a simple fork-and-wait example. The terminal output shows the execution of `gcc Task4.c -o task4` and the running of `./task4`. The output lists the parent and child processes, their PIDs, and the parent's wait status. The parent process (PID 16088) is shown waiting for the child process (PID 16048) to finish, and the output "Parent still running..." is printed.

```
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 finish
11        printf("Parent still running...\n");
12    }
13    return 0;
14 }
```

```
shanzai209@DESKTOP-LI58TTJ:~/Lab_05/Lab_5$ gcc Task4.c -o task4
shanzai209@DESKTOP-LI58TTJ:~/Lab_05/Lab_5$ ./task4
total 80
-rw-r--r-- 1 shanzai209 shanzai209 146 Oct 24 21:25 Task1.c
-rw-r--r-- 1 shanzai209 shanzai209 314 Oct 24 21:26 Task2.c
-rw-r--r-- 1 shanzai209 shanzai209 244 Oct 24 21:28 Task3.c
-rw-r--r-- 1 shanzai209 shanzai209 328 Oct 24 21:29 Task4.c
-rwxr-xr-x 1 shanzai209 shanzai209 16048 Oct 24 21:25 task1
-rwxr-xr-x 1 shanzai209 shanzai209 16088 Oct 24 21:26 task2
-rwxr-xr-x 1 shanzai209 shanzai209 16048 Oct 24 21:28 task3
-rwxr-xr-x 1 shanzai209 shanzai209 16088 Oct 24 21:29 task4
Parent still running...
```