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

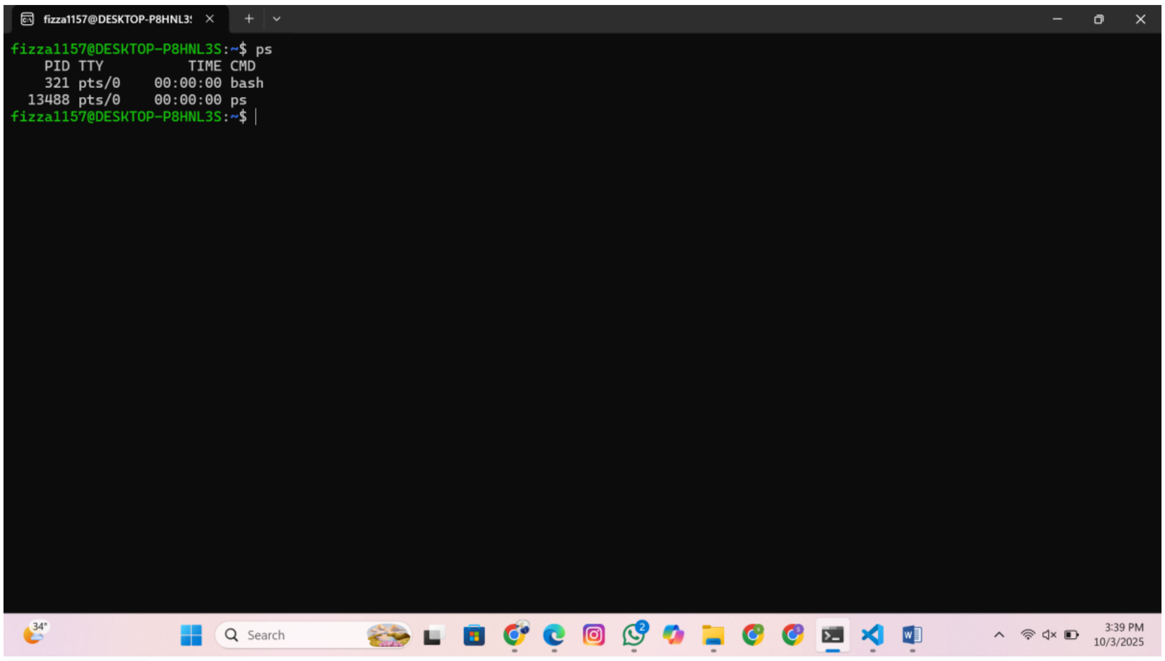
- Shows processes in the current terminal session.

```
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

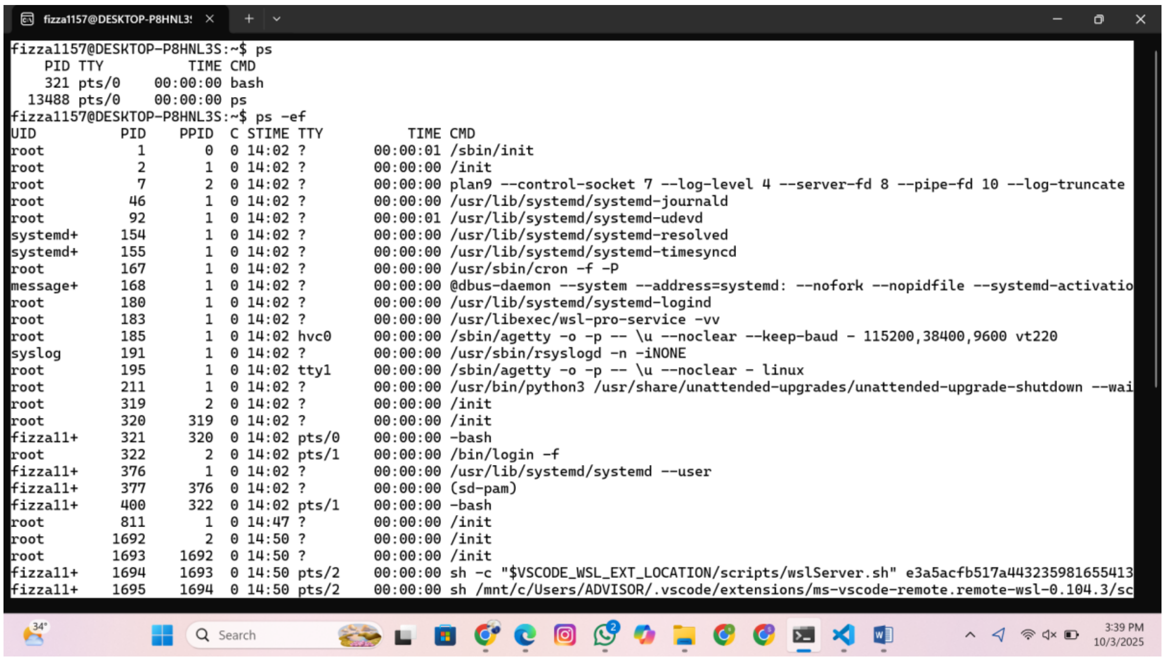


```
fizza1157@DESKTOP-P8HNL3S:~$ ps
    PID TTY          TIME CMD
    321 pts/0    00:00:00 bash
  13488 pts/0    00:00:00 ps
fizza1157@DESKTOP-P8HNL3S:~$
```

`ps -ef` → Full list of all processes

```
ps -ef
```

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

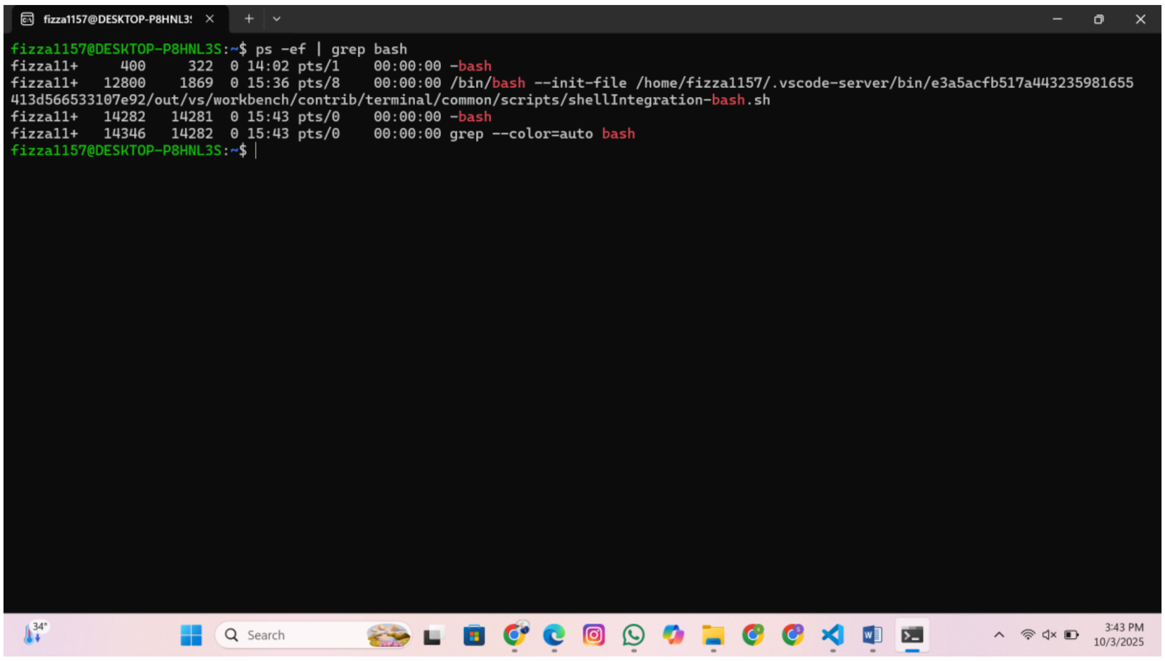


```
fizza1157@DESKTOP-P8HNL3S:~$ ps
    PID TTY          TIME CMD
    321 pts/0    00:00:00 bash
  13488 pts/0    00:00:00 ps
fizza1157@DESKTOP-P8HNL3S:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 14:02 ?        00:00:01 /sbin/init
root           2       1  0 14:02 ?        00:00:00 /init
root           7       2  0 14:02 ?        00:00:00 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --log-truncate
root          46       1  0 14:02 ?        00:00:00 /usr/lib/systemd/systemd-journald
root          92       1  0 14:02 ?        00:00:01 /usr/lib/systemd/systemd-udevd
systemd+     154       1  0 14:02 ?        00:00:00 /usr/lib/systemd/systemd-resolved
systemd+     155       1  0 14:02 ?        00:00:00 /usr/lib/systemd/systemd-timesyncd
root         167       1  0 14:02 ?        00:00:00 /usr/sbin/cron -f -P
message+     168       1  0 14:02 ?        00:00:00 @dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activatio
root         180       1  0 14:02 ?        00:00:00 /usr/lib/systemd/systemd-logind
root         183       1  0 14:02 ?        00:00:00 /usr/libexec/wsl-pro-service -vv
root         185       1  0 14:02 hvc0     00:00:00 /sbin/agetty -o -p -- \u --noclear --keep-baud - 115200,38400,9600 vt220
syslog       191       1  0 14:02 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root         195       1  0 14:02 tty1     00:00:00 /sbin/agetty -o -p -- \u --noclear - linux
root         211       1  0 14:02 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wai
root         319       2  0 14:02 ?        00:00:00 /init
root         320     319  0 14:02 ?        00:00:00 /init
fizza11+     321     320  0 14:02 pts/0    00:00:00 -bash
root         322       2  0 14:02 pts/1    00:00:00 /bin/login -f
fizza11+     376       1  0 14:02 ?        00:00:00 /usr/lib/systemd/systemd --user
fizza11+     377     376  0 14:02 ?        00:00:00 (sd-pam)
fizza11+     400     322  0 14:02 pts/1    00:00:00 -bash
root         811       1  0 14:47 ?        00:00:00 /init
root        1692       2  0 14:50 ?        00:00:00 /init
root        1693    1692  0 14:50 ?        00:00:00 /init
fizza11+    1694    1693  0 14:50 pts/2    00:00:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslServer.sh" e3a5acfb517a443235981655413
fizza11+    1695    1694  0 14:50 pts/2    00:00:00 sh /mnt/c/Users/ADVISOR/.vscode/extensions/ms-vscode-remote.remote-wsl-0.104.3/sc
```

Try:

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

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



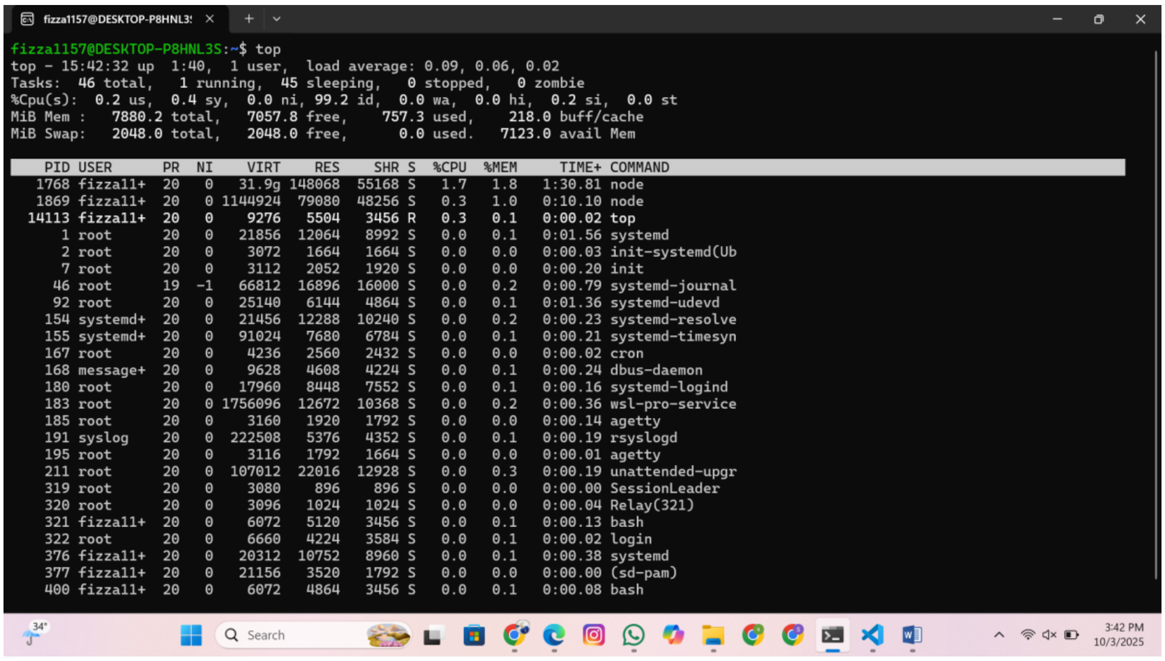
```
fizza1157@DESKTOP-P8HNL3S:~$ ps -ef | grep bash
fizza11+     400     322  0 14:02 pts/1    00:00:00 -bash
fizza11+   12800    1869  0 15:36 pts/8    00:00:00 /bin/bash --init-file /home/fizza1157/.vscode-server/bin/e3a5acfb517a443235981655
413d566533107e92/out/vs/workbench/contrib/terminal/common/scripts/shellIntegration-bash.sh
fizza11+   14282   14281  0 15:43 pts/0    00:00:00 -bash
fizza11+   14346   14282  0 15:43 pts/0    00:00:00 grep --color=auto bash
fizza1157@DESKTOP-P8HNL3S:~$
```

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.



```
fizza1157@DESKTOP-P8HNL3S:~$ top
top - 15:42:32 up  1:40,  1 user,  load average: 0.09, 0.06, 0.02
Tasks:  46 total,   1 running,  45 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.4 sy,  0.0 ni, 99.2 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7880.2 total,   7057.8 free,    757.3 used,    218.0 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7123.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1768 fizza11+  20   0   31.9g 148068  55168 S   1.7   1.8   1:30.81 node
   1869 fizza11+  20   0 1144924  79080  48256 S   0.3   1.0   0:10.10 node
  14113 fizza11+  20   0    9276   5504   3456 R   0.3   0.1   0:00.02 top
      1 root      20   0   21856  12064   8992 S   0.0   0.1   0:01.56 systemd
      2 root      20   0    3072   1664   1664 S   0.0   0.0   0:00.03 init-systemd(Ub
      7 root      20   0    3112   2052   1920 S   0.0   0.0   0:00.20 init
     46 root      19  -1   66812  16896  16000 S   0.0   0.2   0:00.79 systemd-journal
     92 root      20   0   25140   6144   4864 S   0.0   0.1   0:01.36 systemd-udevd
    154 systemd+  20   0   21456  12288  10240 S   0.0   0.2   0:00.23 systemd-resolve
    155 systemd+  20   0   91024   7680   6784 S   0.0   0.1   0:00.21 systemd-timesyn
    167 root      20   0    4236   2560   2432 S   0.0   0.0   0:00.02 cron
    168 message+  20   0    9628   4608   4224 S   0.0   0.1   0:00.24 dbus-daemon
    180 root      20   0   17960   8448   7552 S   0.0   0.1   0:00.16 systemd-logind
    183 root      20   0 1756096  12672  10368 S   0.0   0.2   0:00.36 wsl-pro-service
    185 root      20   0    3160   1920   1792 S   0.0   0.0   0:00.14 agetty
    191 syslog    20   0  222508   5376   4352 S   0.0   0.1   0:00.19 rsyslogd
    195 root      20   0    3116   1792   1664 S   0.0   0.0   0:00.01 agetty
    211 root      20   0  107012  22016  12928 S   0.0   0.3   0:00.19 unattended-upgr
    319 root      20   0    3080    896    896 S   0.0   0.0   0:00.00 SessionLeader
    320 root      20   0    3096   1024   1024 S   0.0   0.0   0:00.04 Relay(321)
    321 fizza11+  20   0    6072   5120   3456 S   0.0   0.1   0:00.13 bash
    322 root      20   0    6660   4224   3584 S   0.0   0.1   0:00.02 login
    376 fizza11+  20   0   20312  10752   8960 S   0.0   0.1   0:00.38 systemd
    377 fizza11+  20   0   21156   3520   1792 S   0.0   0.0   0:00.00 (sd-pam)
    400 fizza11+  20   0    6072   4864   3456 S   0.0   0.1   0:00.08 bash
```

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:

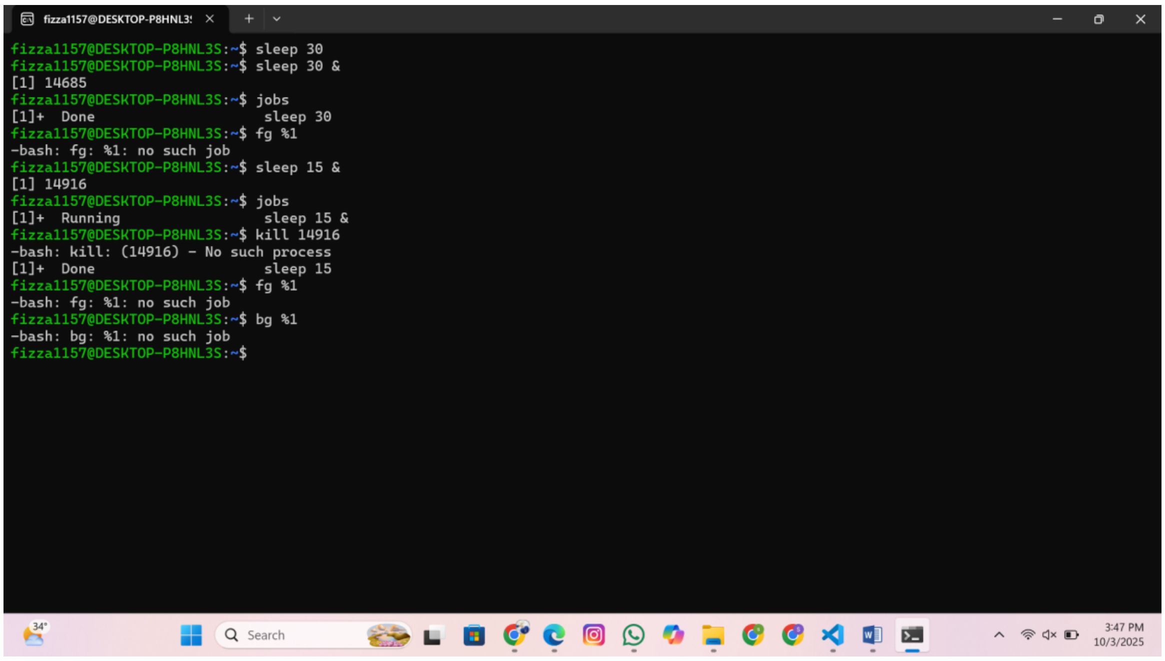
- **Bring a job to foreground**:

  ```
  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
  ```



```
fizza1157@DESKTOP-P8HNL3S:~$ sleep 30
fizza1157@DESKTOP-P8HNL3S:~$ sleep 30 &
[1] 14685
fizza1157@DESKTOP-P8HNL3S:~$ jobs
[1]+  Done                    sleep 30
fizza1157@DESKTOP-P8HNL3S:~$ fg %1
-bash: fg: %1: no such job
fizza1157@DESKTOP-P8HNL3S:~$ sleep 15 &
[1] 14916
fizza1157@DESKTOP-P8HNL3S:~$ jobs
[1]+  Running                 sleep 15 &
fizza1157@DESKTOP-P8HNL3S:~$ kill 14916
-bash: kill: (14916) - No such process
[1]+  Done                    sleep 15
fizza1157@DESKTOP-P8HNL3S:~$ fg %1
-bash: fg: %1: no such job
fizza1157@DESKTOP-P8HNL3S:~$ bg %1
-bash: bg: %1: no such job
fizza1157@DESKTOP-P8HNL3S:~$
```

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:

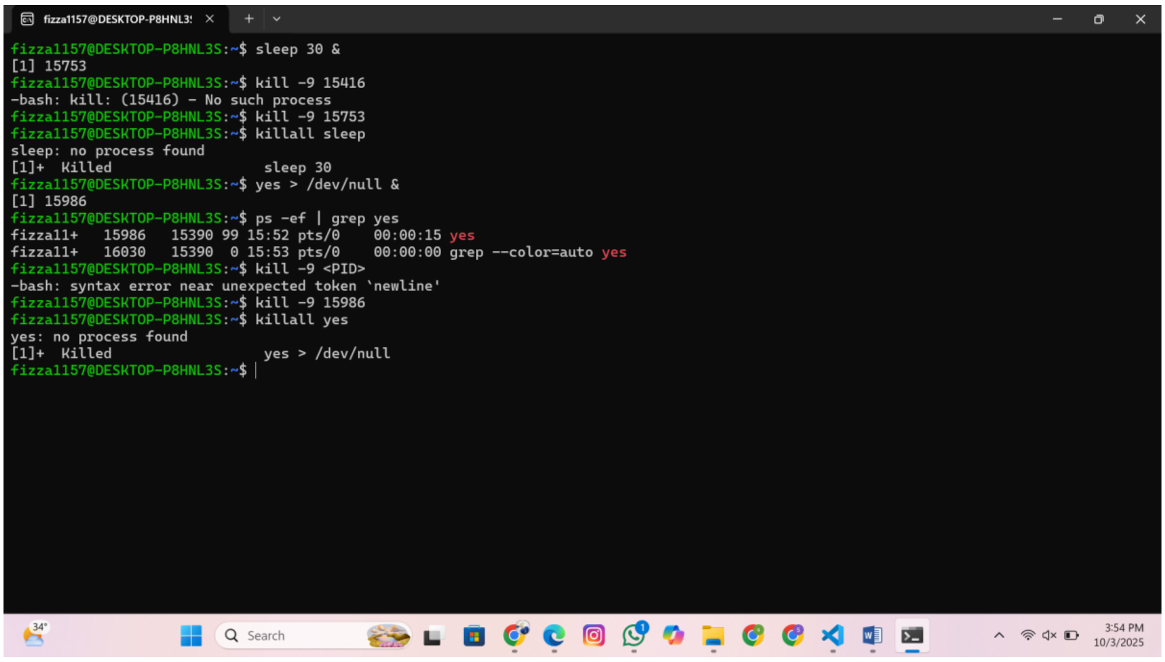
   ```
   yes > /dev/null &
   ```

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

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

2. Find it with:
3. Kill it with:

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



```
fizza1157@DESKTOP-P8HNL3S:~$ sleep 30 &
[1] 15753
fizza1157@DESKTOP-P8HNL3S:~$ kill -9 15416
-bash: kill: (15416) - No such process
fizza1157@DESKTOP-P8HNL3S:~$ kill -9 15753
fizza1157@DESKTOP-P8HNL3S:~$ killall sleep
sleep: no process found
[1]+  Killed                  sleep 30
fizza1157@DESKTOP-P8HNL3S:~$ yes > /dev/null &
[1] 15986
fizza1157@DESKTOP-P8HNL3S:~$ ps -ef | grep yes
fizza11+   15986   15390 99 15:52 pts/0    00:00:15 yes
fizza11+   16030   15390  0 15:53 pts/0    00:00:00 grep --color=auto yes
fizza1157@DESKTOP-P8HNL3S:~$ kill -9 <PID>
-bash: syntax error near unexpected token `newline'
fizza1157@DESKTOP-P8HNL3S:~$ kill -9 15986
fizza1157@DESKTOP-P8HNL3S:~$ killall yes
yes: no process found
[1]+  Killed                  yes > /dev/null
fizza1157@DESKTOP-P8HNL3S:~$
```

---

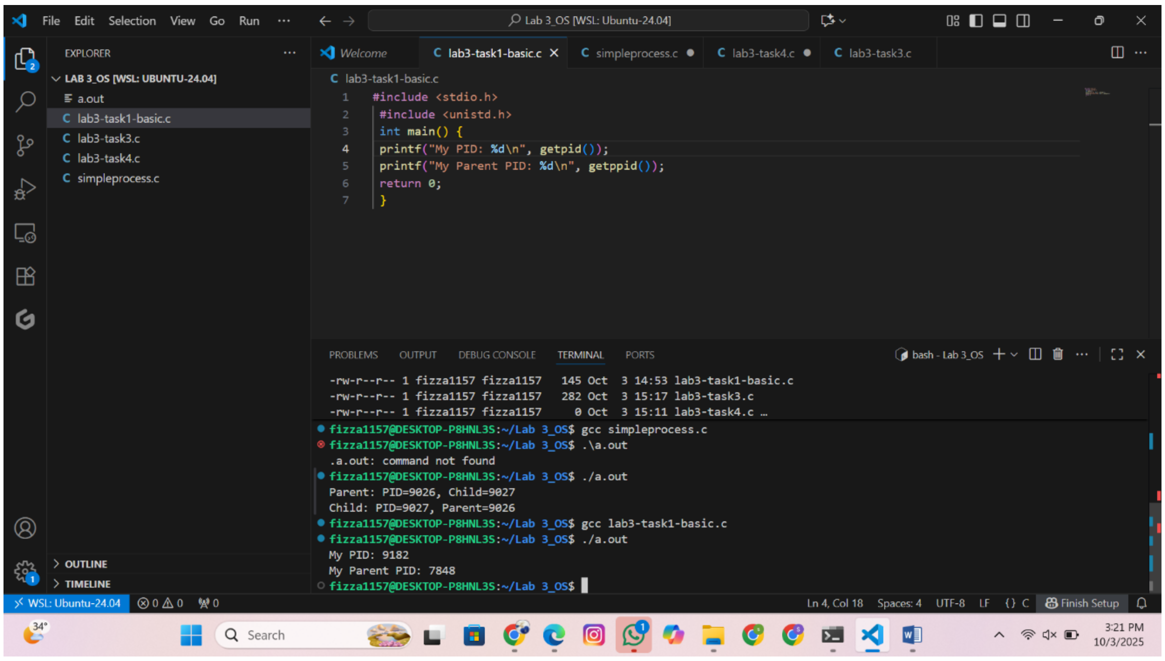
3. C Programs on Processes

Program 1: Print PID and PPID

```c
#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`).



```c
#include <stdio.h>
 #include <unistd.h>
 int main() {
 printf("My PID: %d\n", getpid());
 printf("My Parent PID: %d\n", getppid());
 return 0;
 }
```

```
-rw-r--r-- 1 fizza1157 fizza1157   145 Oct  3 14:53 lab3-task1-basic.c
-rw-r--r-- 1 fizza1157 fizza1157   282 Oct  3 15:17 lab3-task3.c
-rw-r--r-- 1 fizza1157 fizza1157     0 Oct  3 15:11 lab3-task4.c …
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc simpleprocess.c
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ .\a.out
.a.out: command not found
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ ./a.out
Parent: PID=9026, Child=9027
Child: PID=9027, Parent=9026
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc lab3-task1-basic.c
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ ./a.out
My PID: 9182
My Parent PID: 7848
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$
```

Run and compare with `ps -ef`.

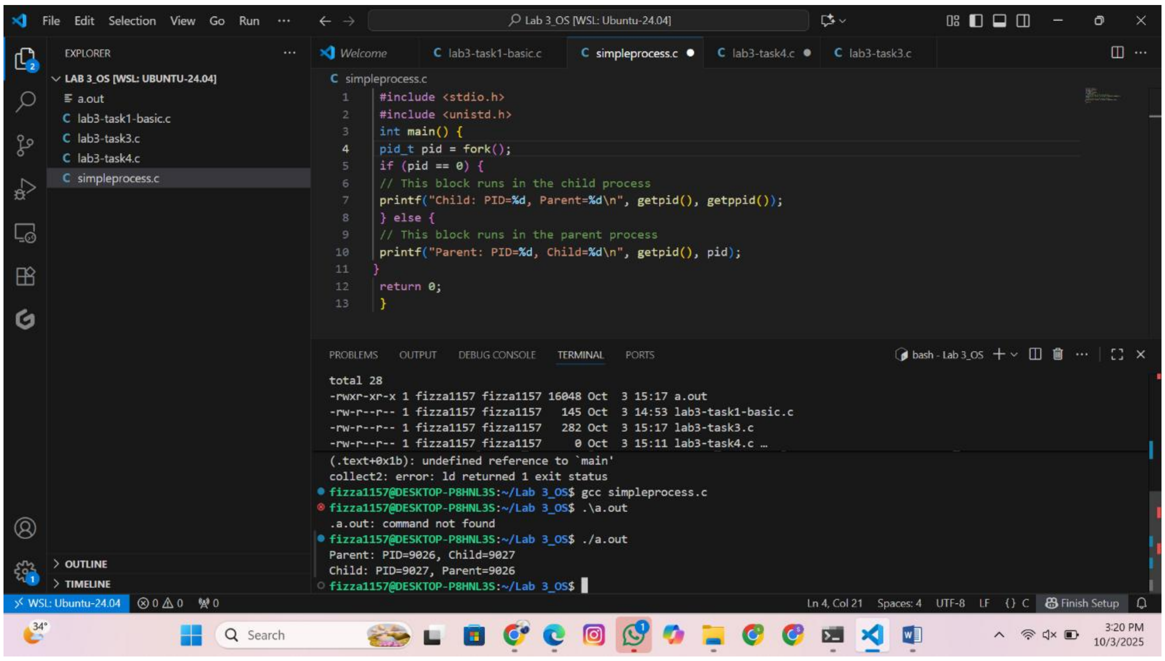
Program 2: Fork – Creating Child Process

```c
#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`.



```c
#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;
}
```

```
total 28
-rwxr-xr-x 1 fizza1157 fizza1157 16048 Oct  3 15:17 a.out
-rw-r--r-- 1 fizza1157 fizza1157   145 Oct  3 14:53 lab3-task1-basic.c
-rw-r--r-- 1 fizza1157 fizza1157   282 Oct  3 15:17 lab3-task3.c
-rw-r--r-- 1 fizza1157 fizza1157     0 Oct  3 15:11 lab3-task4.c …
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc simpleprocess.c
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ .\a.out
.a.out: command not found
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ ./a.out
Parent: PID=9026, Child=9027
Child: PID=9027, Parent=9026
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$
```

Program 3: Execl – Replacing a Process

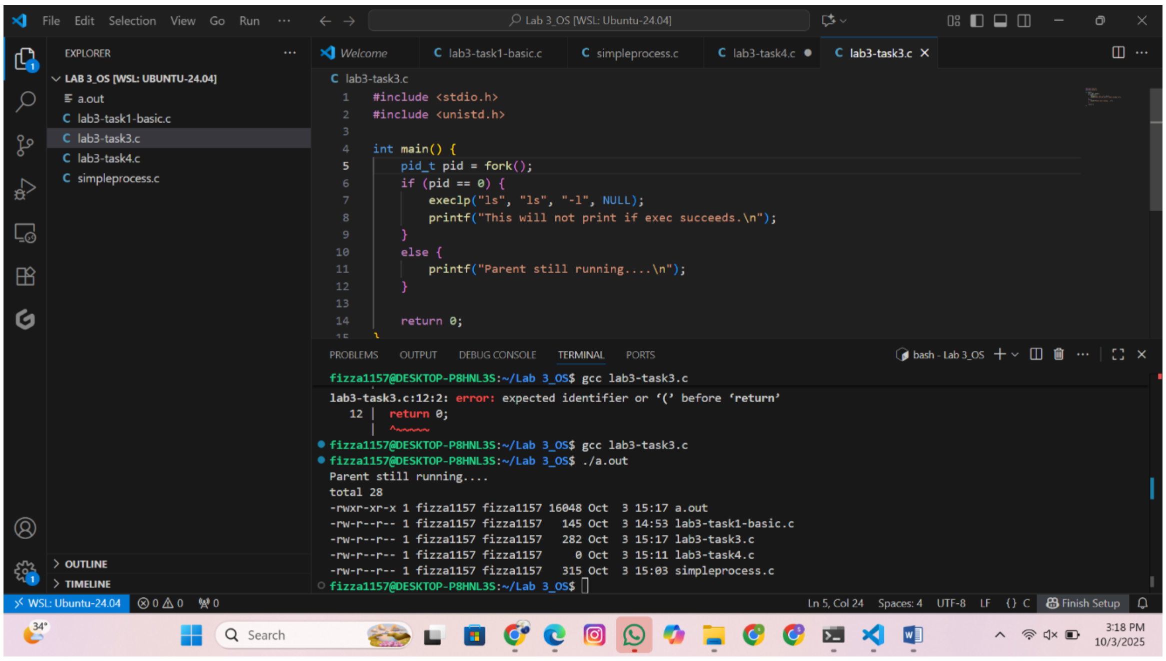
```c
#include <stdio.h>
#include <unistd.h>

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`.



```c
#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;
}
```

```
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc lab3-task3.c
lab3-task3.c:12:2: error: expected identifier or ‘(’ before ‘return’
   12 |  return 0;
      |  ^~~~~~
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc lab3-task3.c
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ ./a.out
Parent still running....
total 28
-rwxr-xr-x 1 fizza1157 fizza1157 16048 Oct  3 15:17 a.out
-rw-r--r-- 1 fizza1157 fizza1157   145 Oct  3 14:53 lab3-task1-basic.c
-rw-r--r-- 1 fizza1157 fizza1157   282 Oct  3 15:17 lab3-task3.c
-rw-r--r-- 1 fizza1157 fizza1157     0 Oct  3 15:11 lab3-task4.c
-rw-r--r-- 1 fizza1157 fizza1157   315 Oct  3 15:03 simpleprocess.c
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$
```

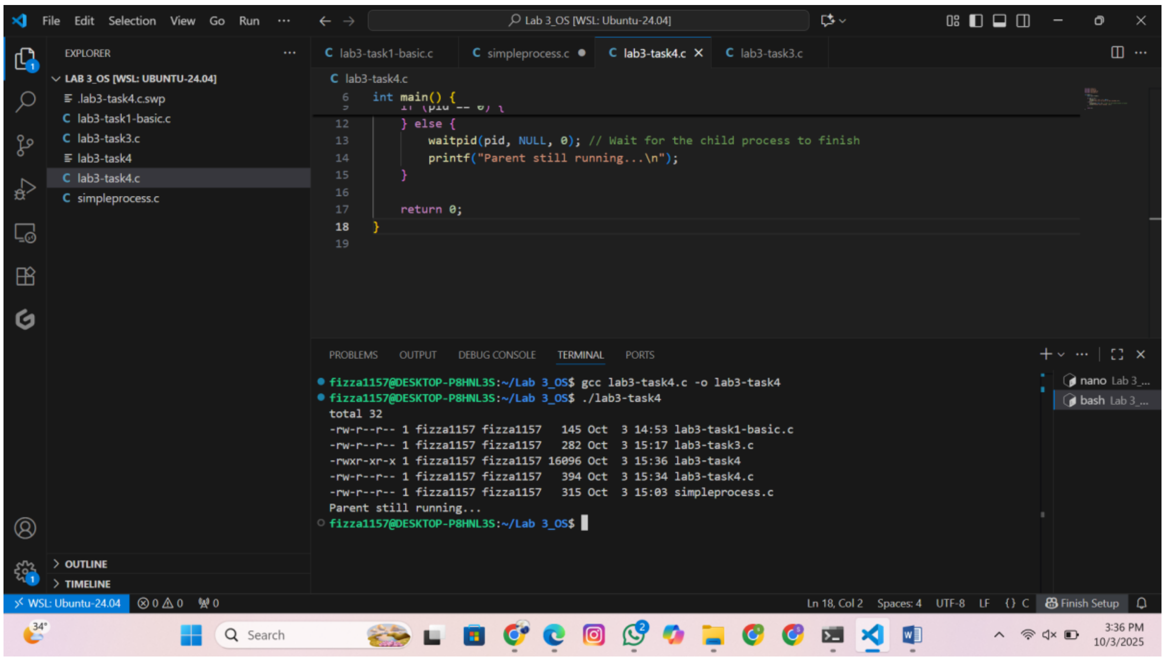
Program 4: Wait – Synchronization

```c
#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**.



```c
int main() {
    } else {
        waitpid(pid, NULL, 0); // Wait for the child process to finish
        printf("Parent still running...\n");
    }

    return 0;
}
```

```
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ gcc lab3-task4.c -o lab3-task4
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$ ./lab3-task4
total 32
-rw-r--r-- 1 fizza1157 fizza1157   145 Oct  3 14:53 lab3-task1-basic.c
-rw-r--r-- 1 fizza1157 fizza1157   282 Oct  3 15:17 lab3-task3.c
-rwxr-xr-x 1 fizza1157 fizza1157 16096 Oct  3 15:36 lab3-task4
-rw-r--r-- 1 fizza1157 fizza1157   394 Oct  3 15:34 lab3-task4.c
-rw-r--r-- 1 fizza1157 fizza1157   315 Oct  3 15:03 simpleprocess.c
Parent still running...
fizza1157@DESKTOP-P8HNL3S:~/Lab 3_OS$
```