



# National Textile University

## Department of Computer Science

Subject:

Operating system

---

Submitted to:

Sir Nasir Mahmood

---

Submitted by:

Areeba Tariq

---

Reg number:

23-NTU-CS-1139

---

Lab no:

3<sup>rd</sup>

---

Semester:

5<sup>th</sup>

---

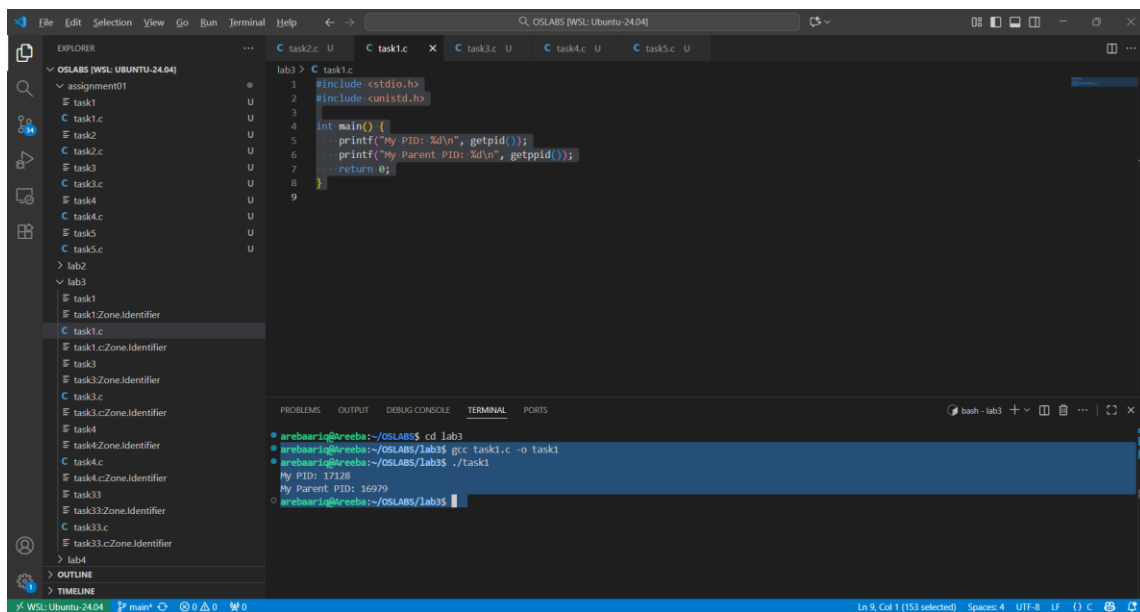
## TASK1:

### CODE:

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

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

### Output:

The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'OSLABS [WSL: Ubuntu-24.04]' with a folder 'lab3' containing files 'task1.c', 'task2.c', 'task3.c', 'task4.c', and 'task5.c'. The 'task1.c' file is open in the editor, showing the code from the previous block. The terminal at the bottom shows the command 'cd lab3' followed by 'gcc task1.c -o task1' and then './task1'. The output of the program is displayed: 'My PID: 17128' and 'My Parent PID: 16979'.

### Remarks:

This program prints its own **Process ID (PID)** and the **Parent Process ID (PPID)** using Linux system calls.

---

## TASK2:

### CODE:

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

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

    if (pid == 0) {
        // Child process
        execlp("ls", "ls", "-l", NULL);

        // Agar exec fail hua to yeh chalega
        printf("This will not print if exec succeeds.\n");
    } else {
        // Parent process
        printf("Parent still running...\n");
    }
}
```

```

    return 0;
}

```

Output:

```

arebaariq@arebaariq:~/OSLABS$ cd lab3
arebaariq@arebaariq:~/OSLABS/lab3$ gcc task1.c -o task1
arebaariq@arebaariq:~/OSLABS/lab3$ ./task1
My PID: 17128
My Parent PID: 16979
arebaariq@arebaariq:~/OSLABS/lab3$ gcc task3.c -o task3
arebaariq@arebaariq:~/OSLABS/lab3$ ./task3
Parent still running...
total 80
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:08 task1
-rw-r--r-- 1 arebaariq arebaariq 153 Oct 26 19:08 task1.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:09 task3
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 26 19:09 task3.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task3.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 5 19:12 task33
-rw-r--r-- 1 arebaariq arebaariq 301 Oct 5 19:19 task33.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16088 Oct 5 19:08 task4
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 4 19:06 task4.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4:Zone.Identifier
arebaariq@arebaariq:~/OSLABS/lab3$

```

Remarks:

The program creates a child process using fork().

Child runs the **ls -l** command using **execlp()**, while the parent prints **"Parent still running..."**.

TASK3:

CODE:

```

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

```

```

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

    if (pid == 0) {
        // Child process
        execlp("ls", "ls", "-l", NULL);
        printf("This will not print if exec succeeds.\n");
    } else {
        // Parent waits for child
        waitpid(pid, NULL, 0);
        printf("Parent still running...\n");
    }

    return 0;
}

```

Output:

Remarks:

After fork(), the child again runs **ls -l**.

The parent uses waitpid() to **wait for the child to finish**, then prints a message.

TASK4:

CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.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;
}
```

Output:

The screenshot shows the Visual Studio Code interface with a C program in `task1.c` and its execution output in the terminal. The program defines several tasks and a main function that prints their identifiers and zone identifiers. The terminal output shows the program's execution, including the printing of task identifiers and zone identifiers for tasks 1, 2, 3, 4, and 5. The output is as follows:

```
arebaariq@areba:~/OSLABS/lab3$ ./task4
total 80
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:08 task1
-rw-r--r-- 1 arebaariq arebaariq 153 Oct 26 19:08 task1.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:09 task3
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 26 19:09 task3.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task3.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 5 19:12 task33
-rw-r--r-- 1 arebaariq arebaariq 301 Oct 5 19:19 task33.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task3:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16088 Oct 26 19:10 task4
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 26 19:10 task4.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4:Zone.Identifier
Parent still running...
arebaariq@areba:~/OSLABS/lab3$ gcc task33.c -o task33
Parent still running...
total 80
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:08 task1
-rw-r--r-- 1 arebaariq arebaariq 153 Oct 26 19:08 task1.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task1:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:09 task3
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 26 19:09 task3.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task3.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16048 Oct 26 19:10 task33
-rw-r--r-- 1 arebaariq arebaariq 301 Oct 26 19:10 task33.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task33:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task3:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 16088 Oct 26 19:10 task4
-rw-r--r-- 1 arebaariq arebaariq 413 Oct 26 19:10 task4.c
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4.c:Zone.Identifier
-rw-r--r-- 1 arebaariq arebaariq 0 Oct 10 14:52 task4:Zone.Identifier
```

Remarks:

Same behavior as Task 2.

Child executes `ls -l`, and the parent prints a message without waiting.