LAB #05

Process Creation and Execution



Spring 2023

**CSE-204L Operating Systems Lab**

Submitted by: MUHAMMAD SADEEQ

Registration No.: 21PWCSE2028

Section: C

“On my honor, as a student of the University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work”

Submitted to:

Engr. Madiha Sher

(31 Mar 2023)

Department of Computer systems engineering

University of Engineering and Technology, Peshawar

**CSE 302L: Operating Systems Lab**

**LAB ASSESSMENT RUBRICS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Marking**  **Criteria** | **Exceeds expectation**  **(2.5)** | **Meets expectation**  **(1.5)** | **Does not meet expectation**  **(0)** | **Score** |
| **1.**  **Correctness** | Program compiles (no errors and no warnings).  Program always works correctly and meets the specification(s).  Completed between 81-  100% of the requirements. | Program compiles (no errors and some warnings). Some details of the program specification are violated, program functions incorrectly for some inputs.  Completed between 4180% of the requirements. | Program fails to or compile with lots of warnings.  Program only functions correctly in very limited cases or not at all.  Completed less than 40% of the requirements. |  |
| **2.**  **Delivery** | Delivered on time, and in correct format (disk, email, hard copy etc.) | Not delivered on time, or slightly incorrect format. | Not delivered on time or not in correct format. |  |
| **3.**  **Coding**  **Standards** | Proper indentation, whitespace, line length, wrapping, comments and references. | Missing some of whitespace, line length, wrapping, comments or references. | Poor use of whitespace, line length, wrapping, comments and references. |  |
| **4.**  **Presentation of document** | Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting and excellently organized. | Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting. | No name, date, or assignment title included. No task titles, no objectives, no output screenshots, poor formatting. |  |

**Instructor:**

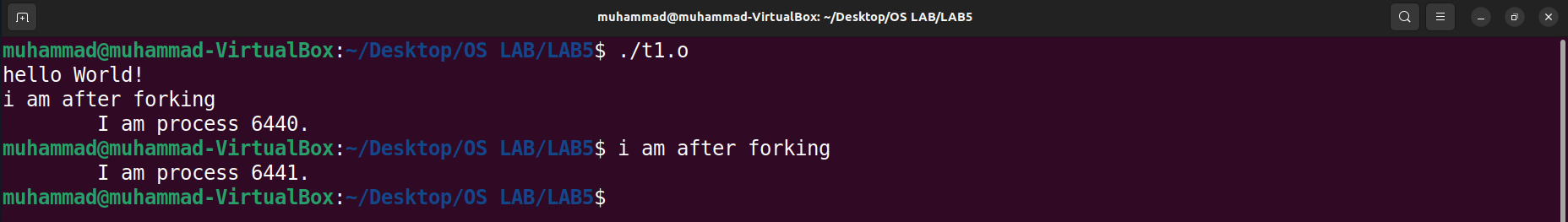
Name: \_\_Engr. Madiha Sher\_\_\_ Signature: MUHAMMAD SADEEQ

**Example 1**

**CODE:**

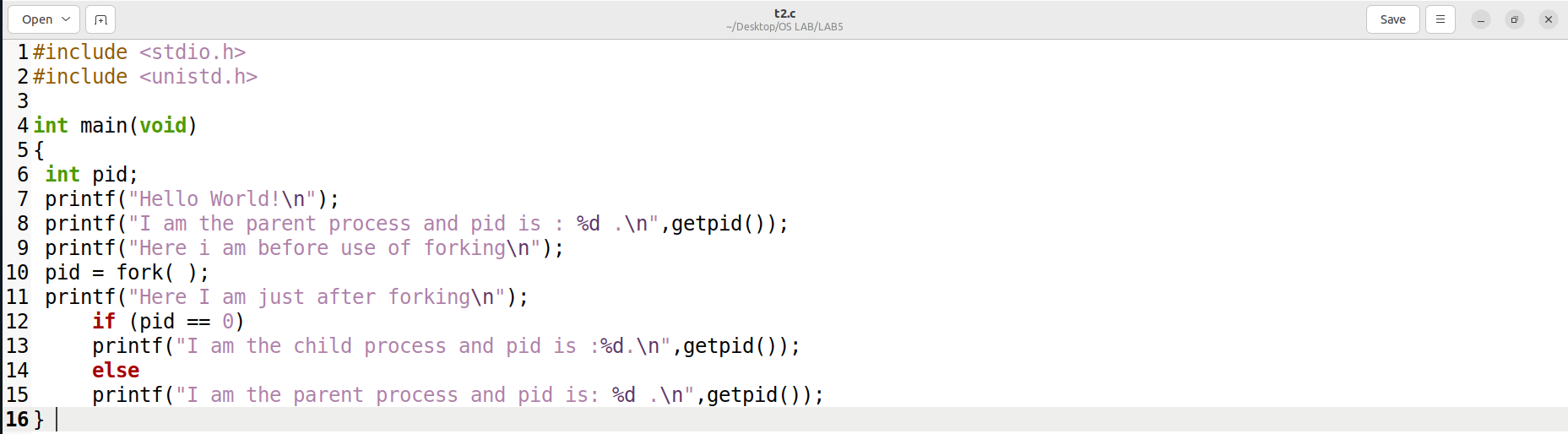
****

**Output:**

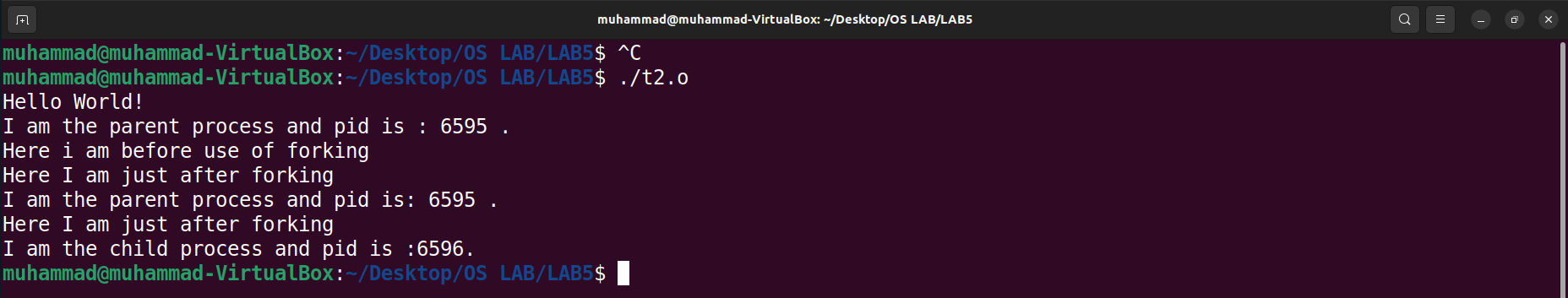
****

**Example 2**

**CODE:**

****

**Output:**

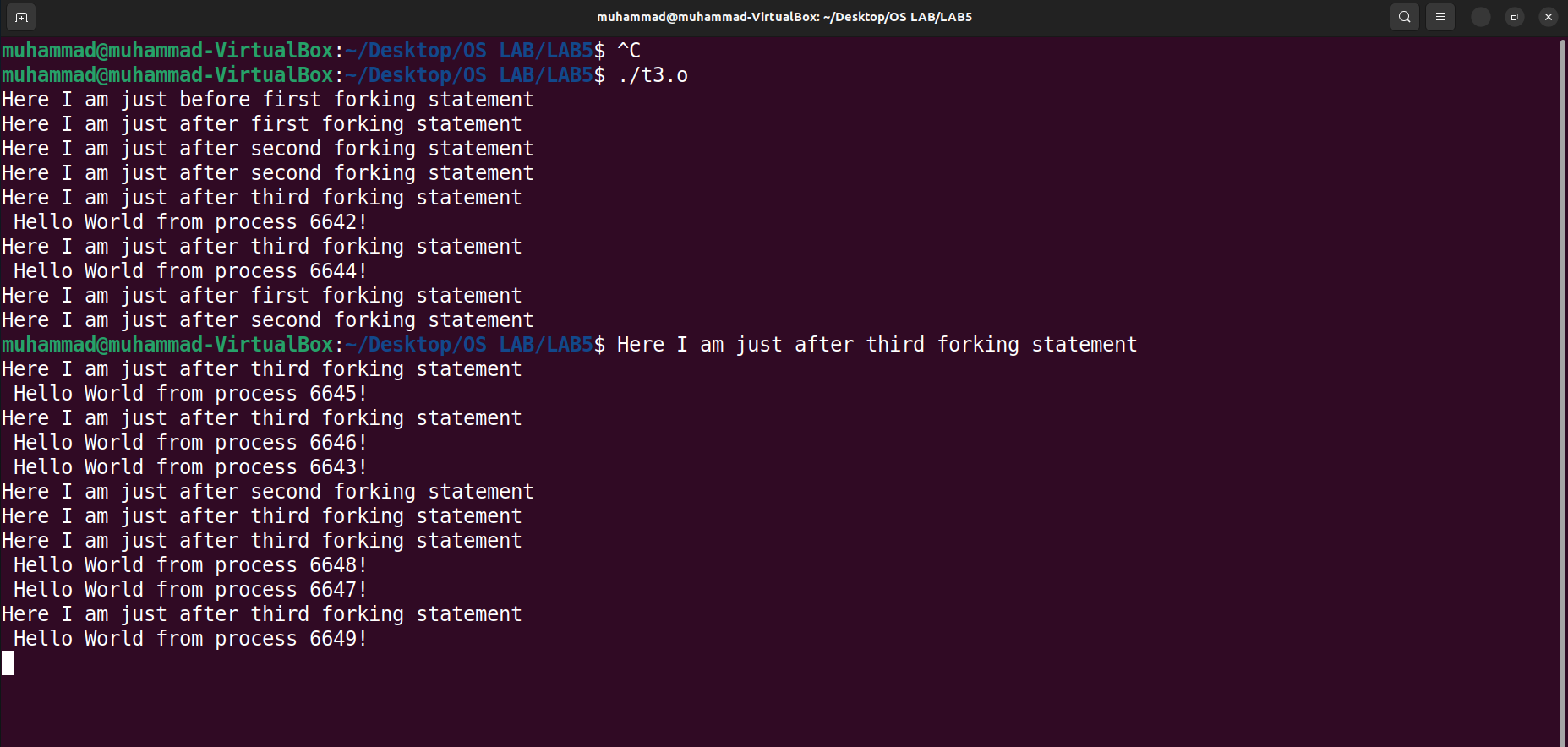
****

**Example 3**

**CODE:**

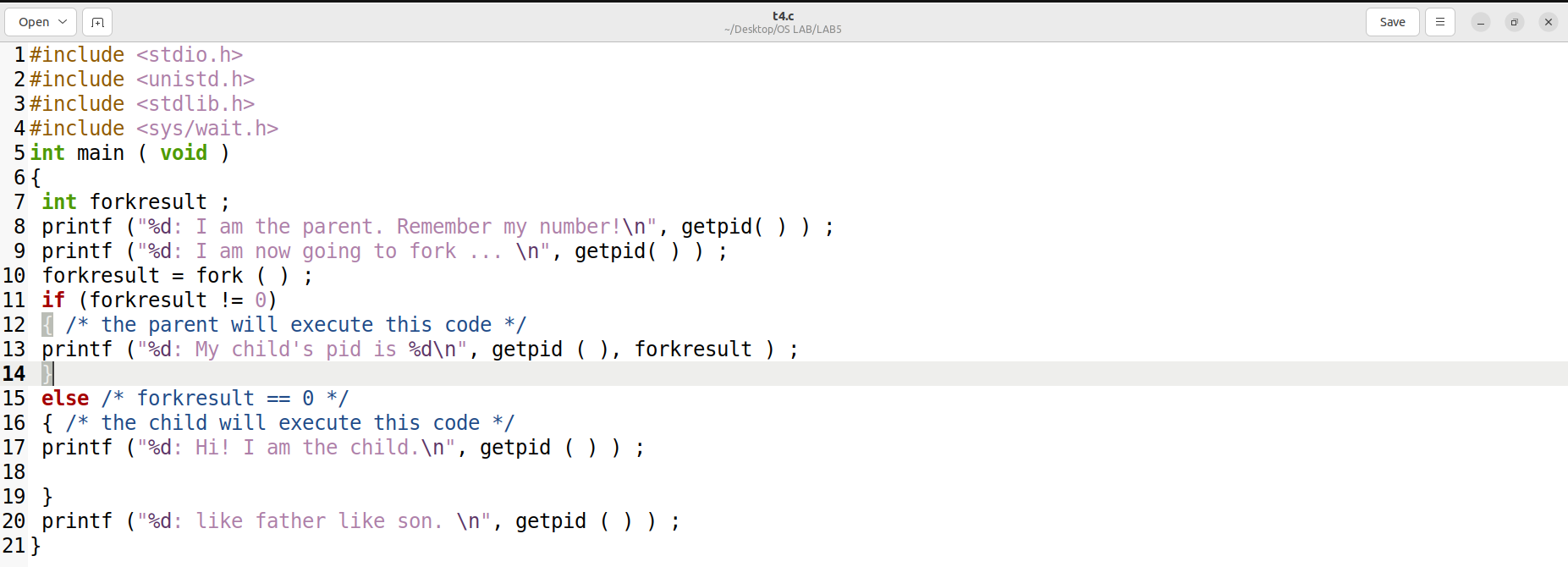
****

**Output:**

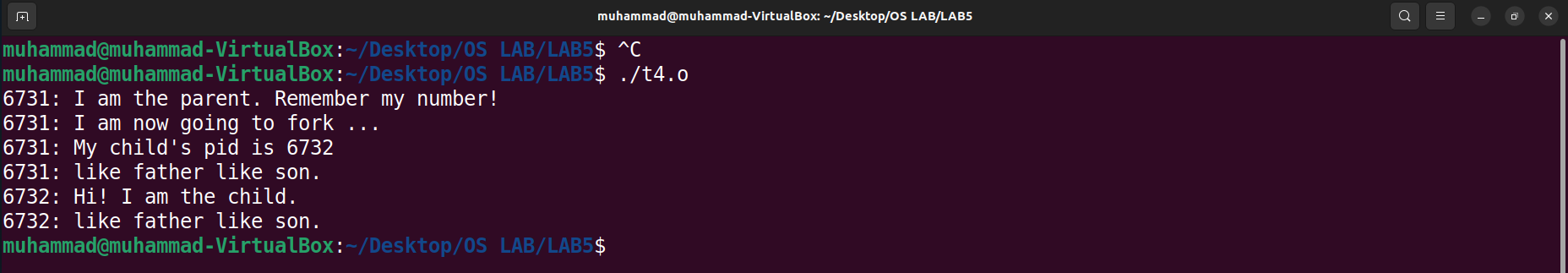
****

**Example 4**

**CODE:**

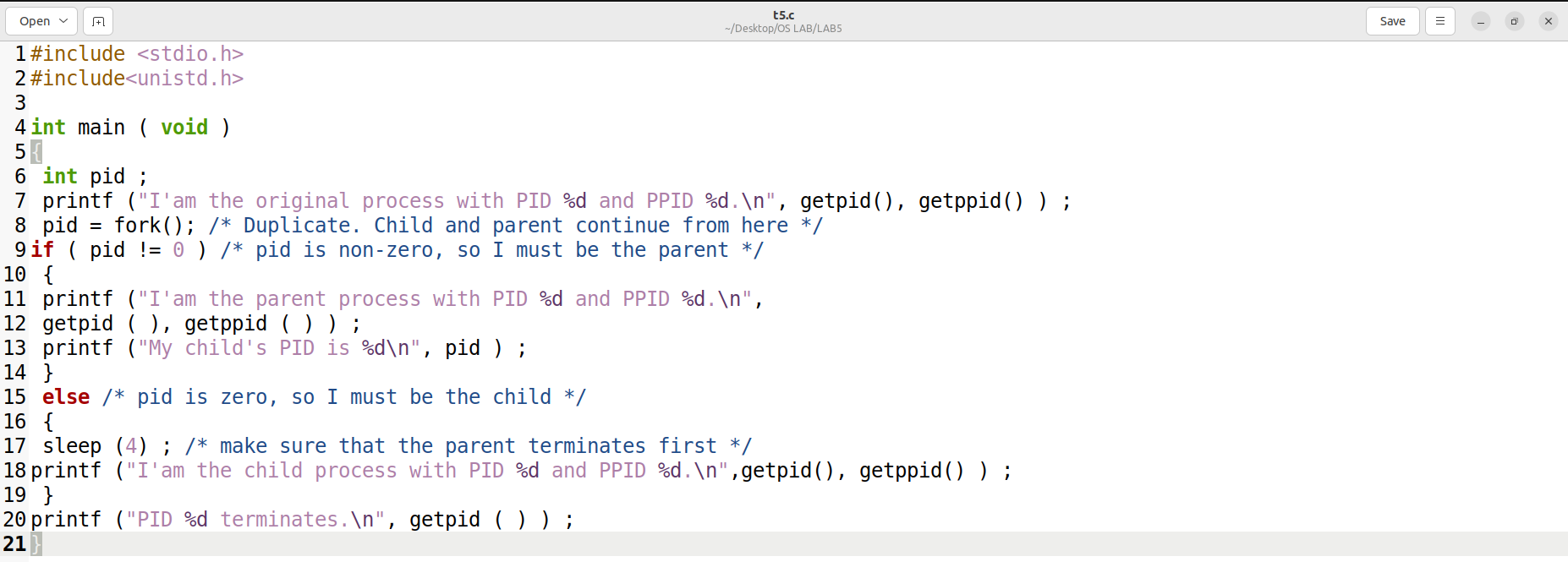
****

**Output:**

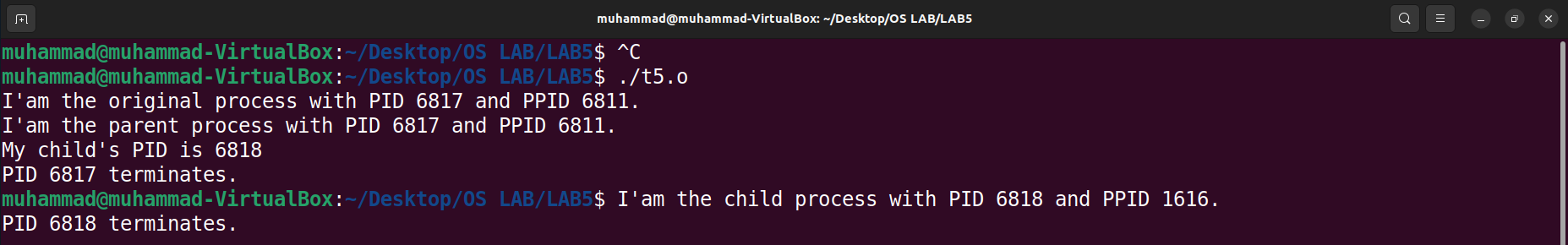
****

**Example 5**

**CODE:**

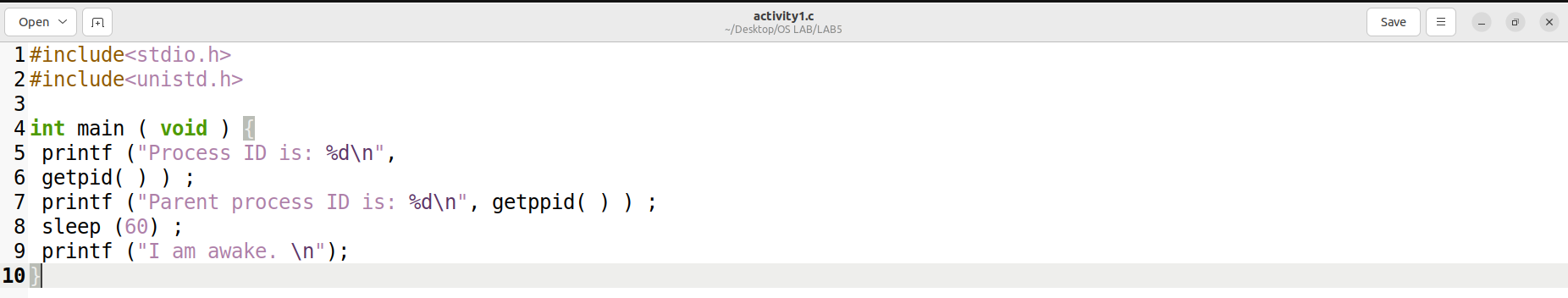
****

**Output:**

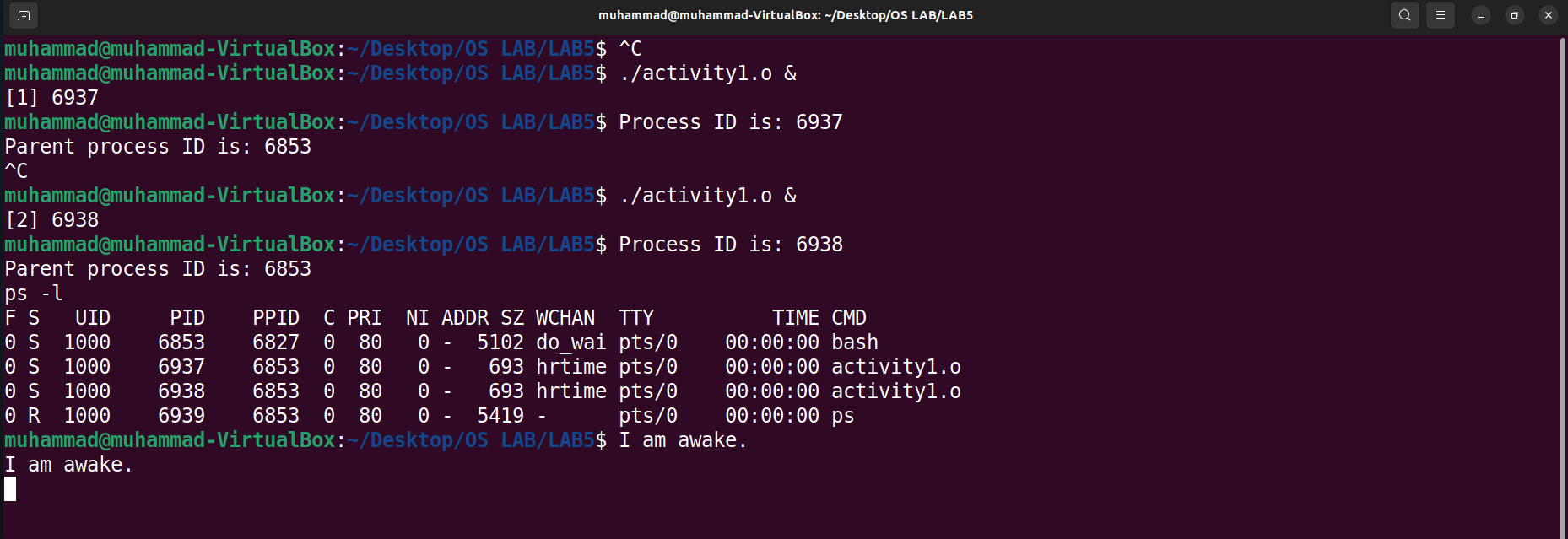
****

**Task #01**

**CODE:**

****

**Output:**

****

**Observation:**

When i run the program twice as a background process, two instances of the program run independently in the background, and I saw two processes with different process IDs in the process table.

Initially, both processes were in the "S" (Sleeping) state as they have invoked the "sleep" system call and are waiting for the specified amount of time (60 seconds) to elapse.

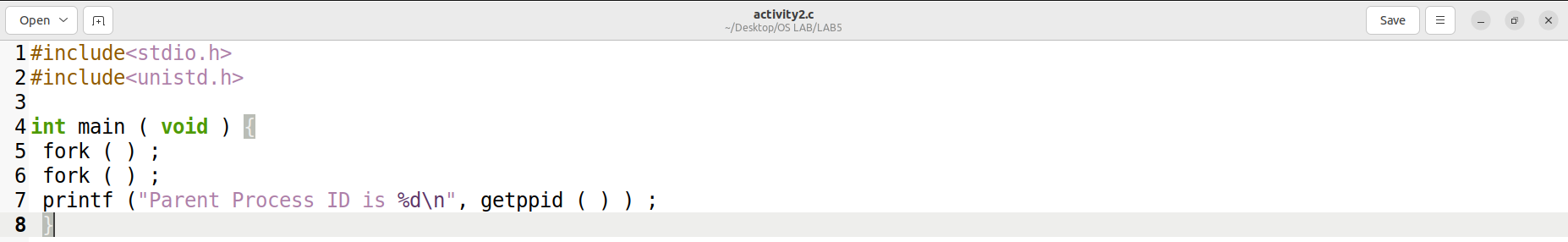
After 60 seconds, the processes waked up and print "I am awake" on the console. Then, they terminate, and their status change to "Z" (Zombie) until their parent process collects their exit status using the "wait" system call.

When i run the program again, we will see two new processes with different PIDs and PPIDs. The process state and behavior will be the same as before.

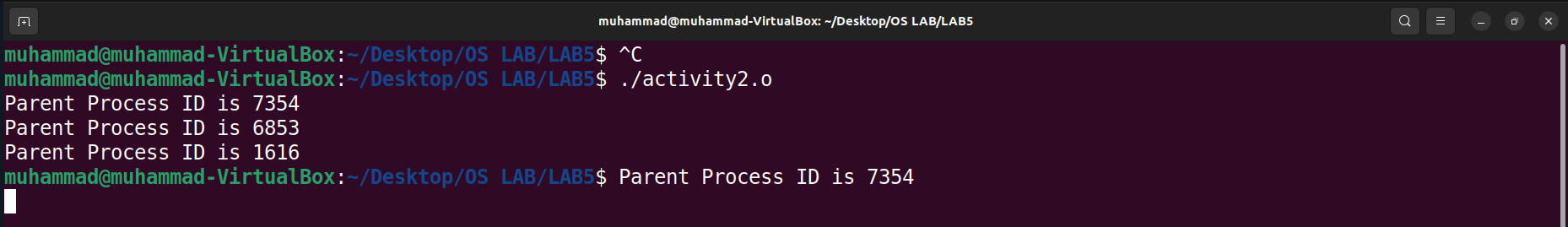
In summary, we can observe that each instance of the program runs independently as a separate process with its own PID and PPID. The process remains in the Sleeping state for 60 seconds and then terminates, leaving behind a Zombie process until the parent process collects its exit status.

**Task #02**

**CODE:**

****

**Output:**

****

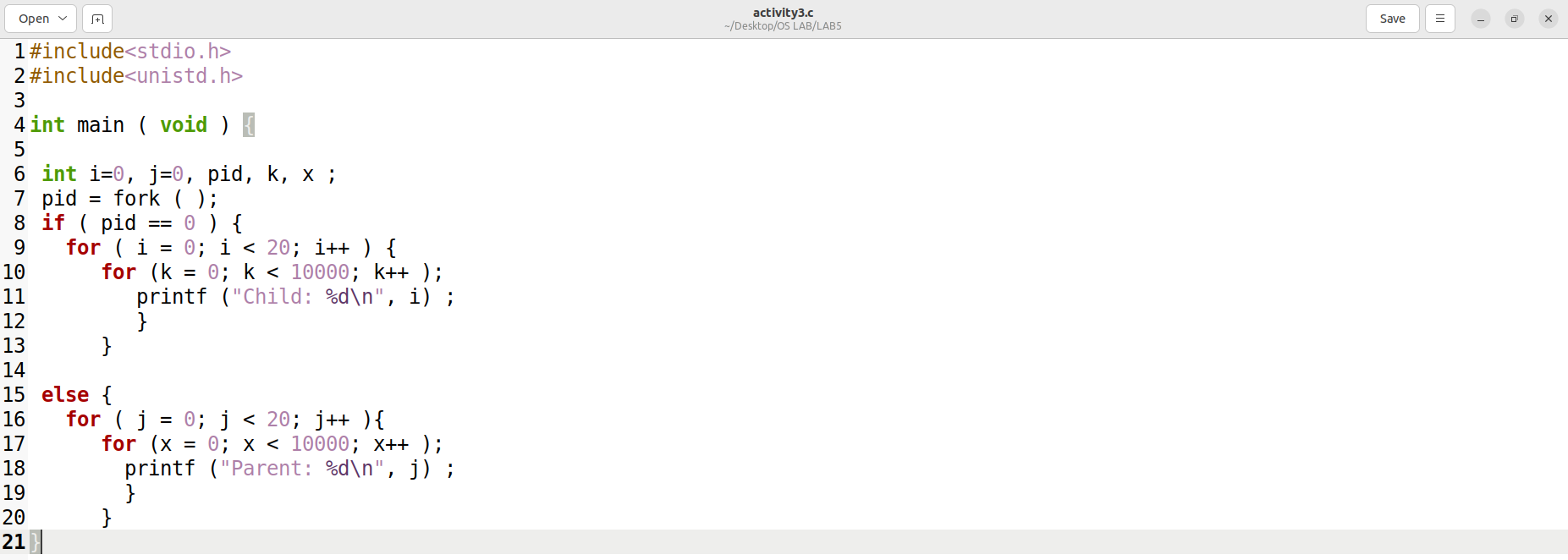
**Family Tree**

**Observation:**

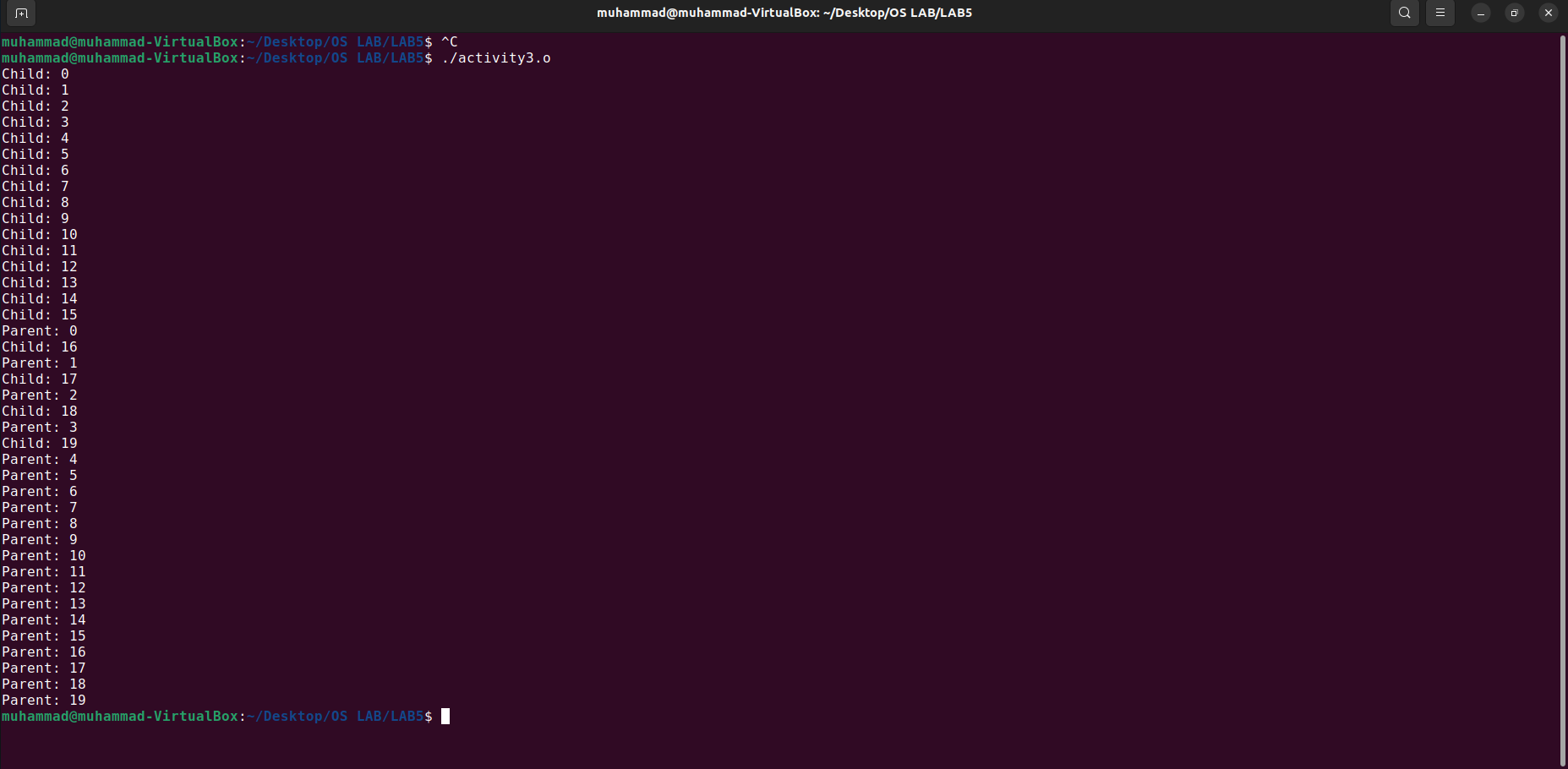
The program will create four processes, and each process will execute the printf() statement, outputting the ID of the original parent process. The order in which the statements are executed may not follow the order in which the fork() statements were executed.

**Task #03**

**CODE:**

****

**Output:**



**Observation:**

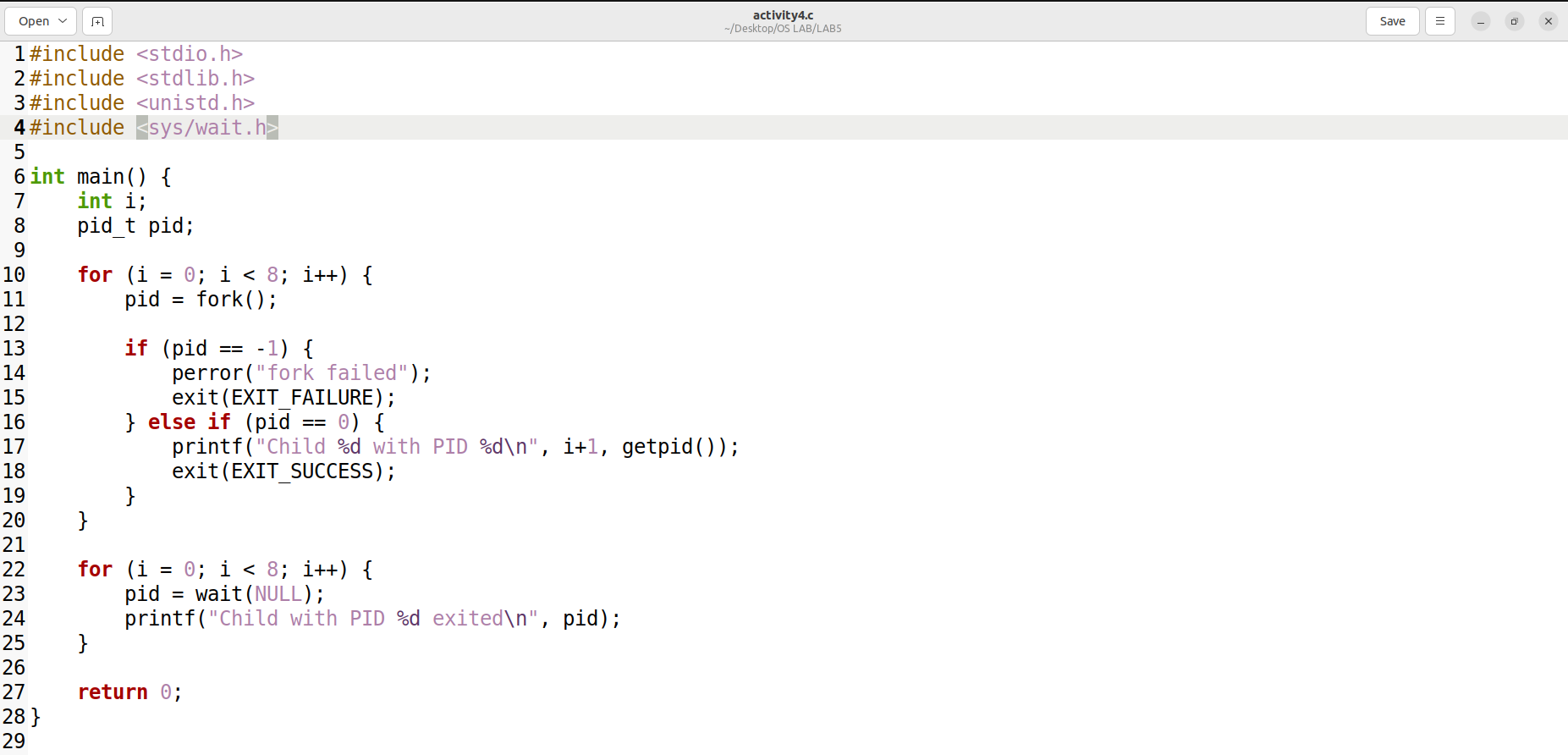
This program creates a parent and child process using the fork() system call. The child process executes the code inside the if block, and the parent process executes the code inside the else block.

Both the child and parent processes have a loop that iterates 20 times and performs a delay of 10000 iterations for each iteration of the loop using the for loop. Inside the loop, a message is printed to the console indicating whether the message was printed by the child or parent process and the current iteration number.

Since both the parent and child processes are running concurrently, the output from the two processes will be interleaved, and we can observe the time slicing used by UNIX.

**Task #04**

**CODE:**



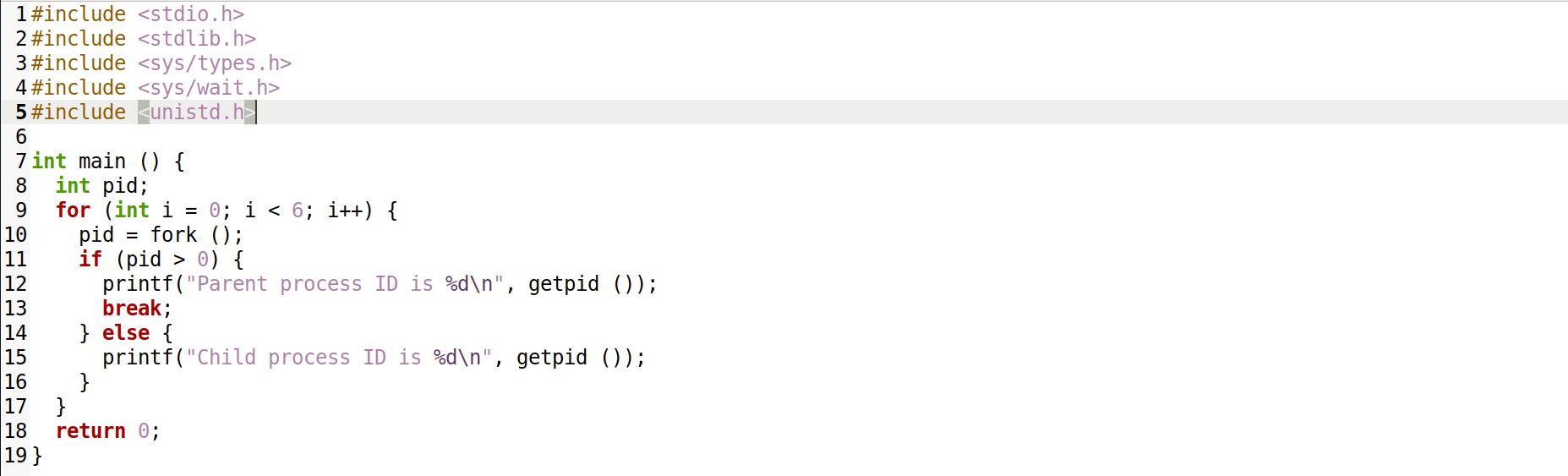
**Output:**



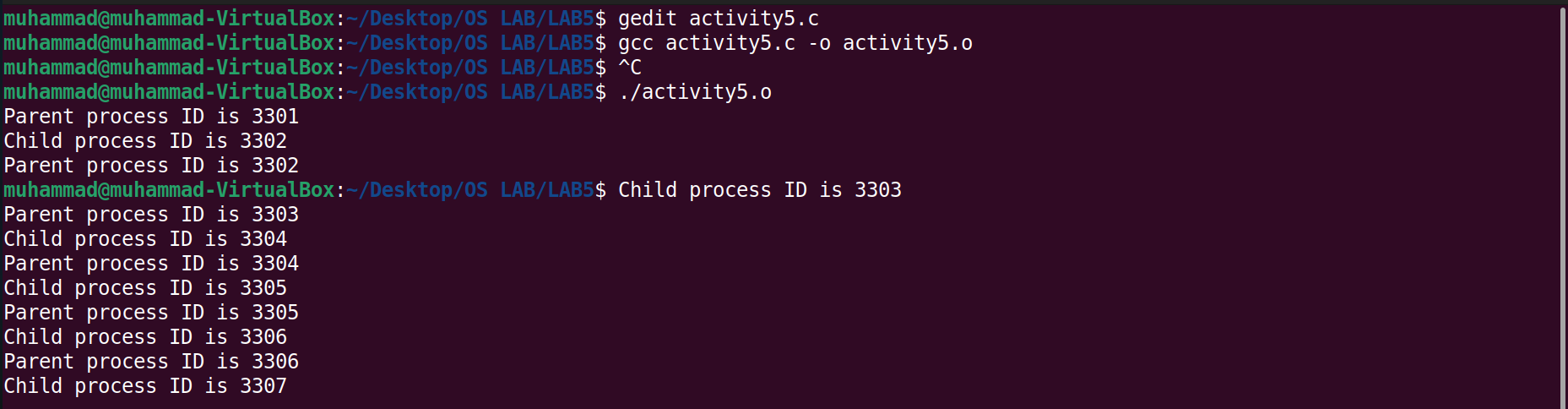
**Figure:**

**Task #05**

**CODE:**

****

**Output:**

****

**Figure:**

**Task #06**

**Figure:**