LAB #08

Threads 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

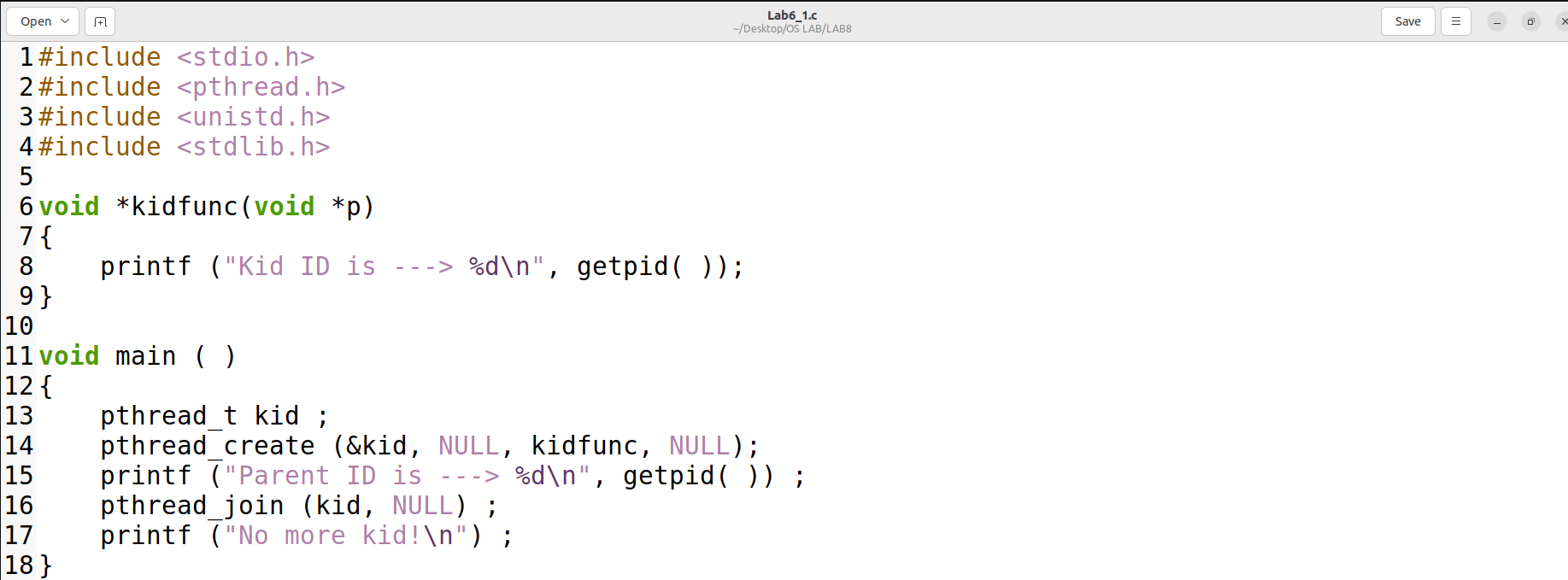
(20 May 2023)

Department of Computer systems engineering

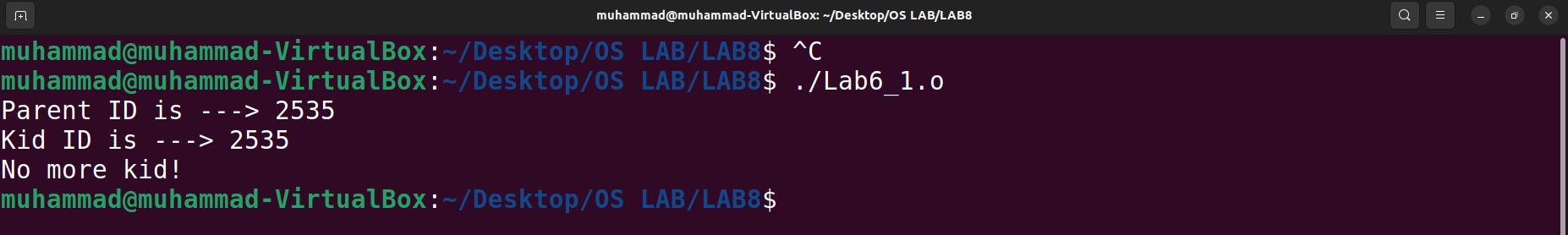
University of Engineering and Technology, Peshawar

**Example Lab6\_1**

**CODE:**

****

**Output:**

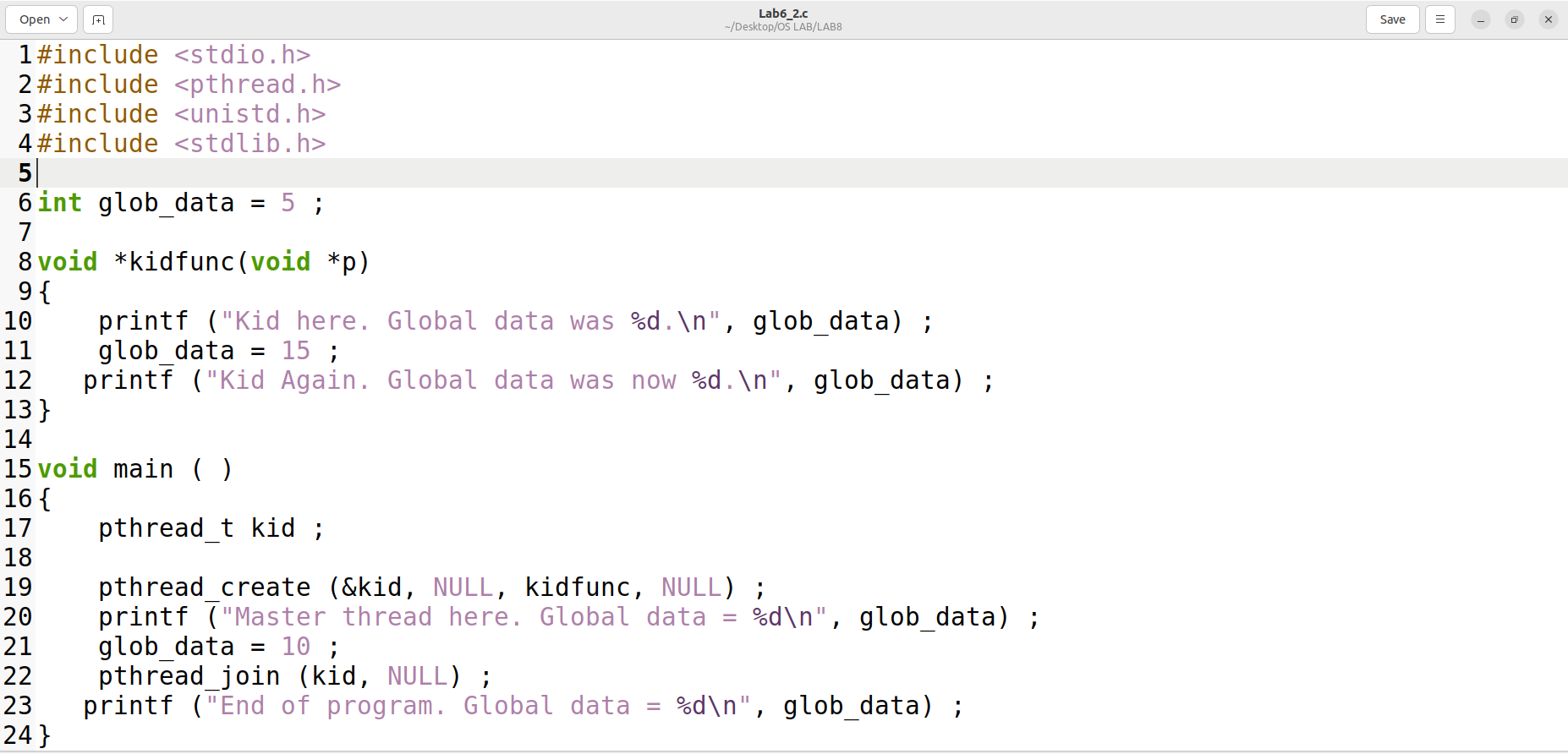
****

**Question: Are the process id numbers of parent and child thread the same or different? Give reason(s) for your answer.**

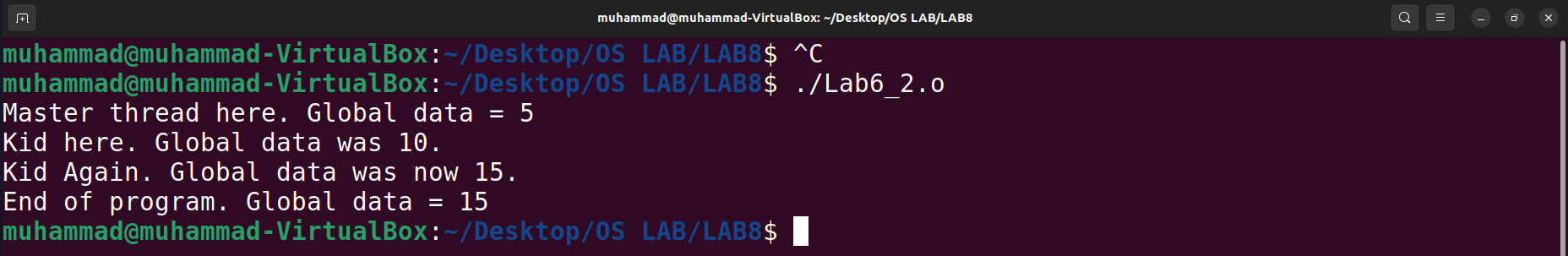
**Answer:** The process id numbers of parent and child thread are the same. This is because threads share the same address space and resources of the process that created them. They are different from processes, which have their own address space and resources and get different process id numbers when created by fork().

**Example Lab6\_2**

**CODE:**



**Output:**



Question: Do the threads have separate copies of glob\_data? Why? Or why not?

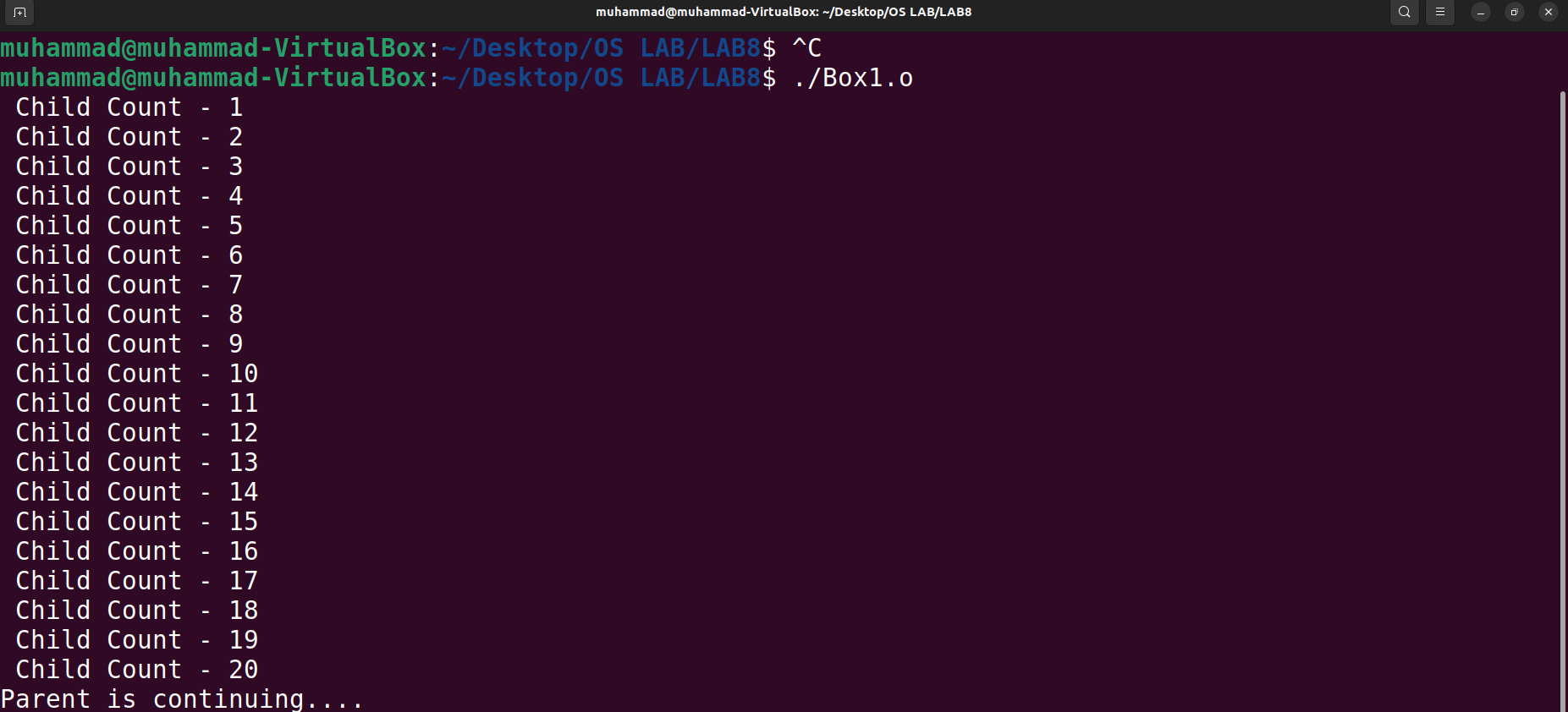
**Answer:** No, the threads do not have separate copies of glob\_data. This is because threads share the same global variables and heap memory of the process that created them. They only have their own stack memory and registers. Therefore, any changes made by one thread to a global variable will be visible to other threads as well.

**Problem#1:**

**CODE:**

|  |
| --- |
| **/\* Box #1:  Simple Child Thread \*/**  **#include <pthread.h>**  **#include <stdio.h>**  **void \*ChildThread(void \*argument)**  **{**  **int i;**  **for ( i = 1; i <= 20; ++i ){**  **printf(" Child Count - %d\n", i);**  **}     pthread\_exit(NULL);**  **}  int main(void)**  **{**  **pthread\_t   hThread;        int   ret;**  **ret=pthread\_create(&hThread, NULL, (void \*)ChildThread, NULL); /\* Create**  **Thread \*/**  **if (ret < 0)**  **printf("Thread Creation Failed\n");   return 1;**  **pthread\_join (hThread, NULL);  /\* Parent waits for  \*/**  **printf("Parent is continuing....\n");**  **return 0;**  **}** |

**Output:**

****

**Compile and execute the Box #1 program and show the output and explain why the output is so?**

**Answer:** The **main** function creates a new thread using the **pthread\_create** function and passes the **ChildThread** function as an argument. This function is then executed by the newly created thread and prints out the numbers from 1 to 20.

After creating the new thread, the **main** function waits for the child thread to finish using the **pthread\_join** function. Once the child thread has finished executing, the main function continues and prints out “Parent is continuing…”.

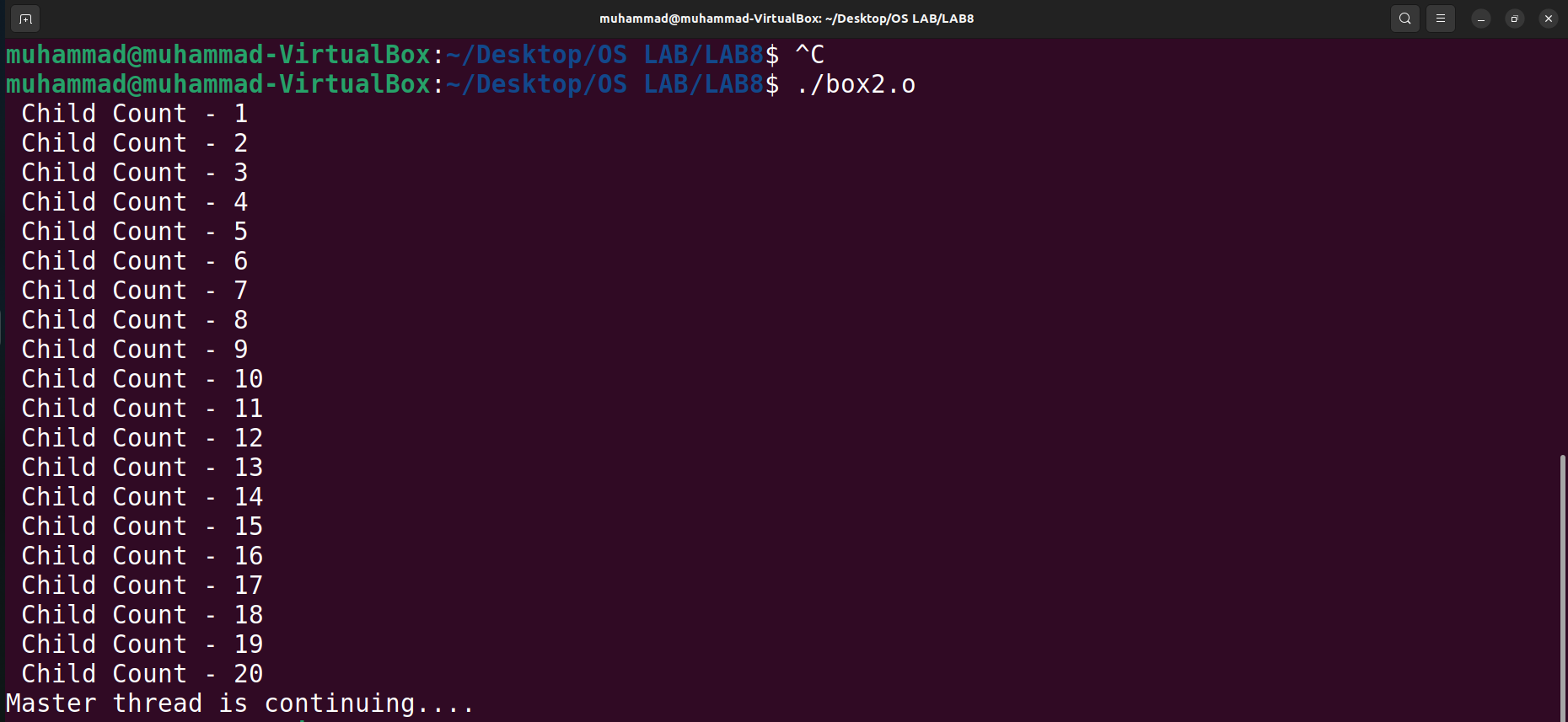
**Problem#2:**

**CODE:**

**Write a program Box # 2 by removing pthread\_exit function from child thread function and check the output? Is it the same as output when pthread\_exit is used? If so Why? Explain?**

****

**Output:**

****

**Answer:** The **pthread\_exit** function has been removed from the **ChildThread** function. When you compile and run this program, you should see the same output as in the previous version of the program.

The reason for this is that when a thread finishes executing its start routine (in this case, the **ChildThread** function), it implicitly calls **pthread\_exit** and terminates. This means that even if you don’t explicitly call **pthread\_exit** in your thread function, it will still be called automatically when the thread finishes executing.

So in this case, removing the **pthread\_exit** function from the **ChildThread** function does not change the behavior of the program.