King Abdulaziz University -Faculty of Engineering
Dept. of Electrical and Computer
Engineering

Lab 6

Name	ID
Nawaf Sami Alharbi	1936576

Example 1:

1) Execute the provided program and observe the output.

```
nawaf@lamp ~/lab6$ ./Ex1
Parent: My process# ---> 1401
Parent: My thread # ---> 140179540780864
Child: Hello World! It's me, process# ---> 1401
Child: Hello World! It's me, thread # ---> 140179540776704
Parent: No more child thread!
nawaf@lamp ~/lab6$
```

2) Are the process ID numbers of the parent and child threads the same or different? Why? No, they are different. Each thread possesses a unique ID, which is evident from the program's output.

Example 2:

3) Run the above program several times; observe its output every time. A sample output follows:

```
nawaf@lamp ~/lab6$ ./Ex2
Parent: Global data = 5
Child: Global data was 5.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
nawaf@lamp ~/lab6$
```

4) Does the program consistently produce the same output? Why? No, it does not consistently generate the same output. The outcome can differ because it relies on the system's ability to complete the thread's task before the main thread modifies the global data. While it may coincide in some instances, running the program multiple times can yield diverse results.

5) Do the threads possess separate copies of the global data? No, the threads share the same memory space and directly access the global data.

Example 3:

6) Run the above program several times and observe the outputs:

```
nawaf@lamp ~/lab6$ ./Ex3
I am the parent thread
I am thread #2, My ID #139800501442304
I am thread #9, My ID #139800442693376
I am thread #7, My ID #139800459478784
I am thread #5, My ID #139800476264192
I am thread #3, My ID #139800493049600
I am thread #1, My ID #139800509835008
I am thread #0, My ID #139800518227712
I am thread #8, My ID #139800451086080
I am thread #6, My ID #139800467871488
I am thread #4, My ID #139800484656896
I am the parent thread again
nawaf@lamp ~/lab6$ ./Ex3
I am the parent thread
I am thread #4, My ID #140615348688640
I am thread #2, My ID #140615365474048
I am thread #8, My ID #140615315117824
I am thread #7, My ID #140615323510528
I am thread #5, My ID #140615340295936
I am thread #3, My ID #140615357081344
I am thread #1, My ID #140615373866752
I am thread #6, My ID #140615331903232
I am thread #0, My ID #140615382259456
I am thread #9, My ID #140615306725120
I am the parent thread again
nawaf@lamp ~/lab6$ ./Ex3
I am the parent thread
I am thread #2, My ID #139626240567040
I am thread #0, My ID #139626257352448
I am thread #6, My ID #139626206996224
I am thread #9, My ID #139626181711616
I am thread #5, My ID #139626215388928
I am thread #4, My ID #139626223781632
I am thread #7, My ID #139626198497024
I am thread #8, My ID #139626190104320
I am thread #1, My ID #139626248959744
I am thread #3, My ID #139626232174336
I am the parent thread again
```

7) Do the output lines consistently appear in the same order? Why? No, the output lines may not consistently appear in the same order. The variation in thread creation time and priority results in different execution sequences and, consequently, diverse output orders.\

Example 4:

8) Run the above program and observe its output. Following is a sample output:

```
nawaf@lamp ~/lab6$ ./Ex4
First, we create two threads to see better what context they share...
Set this_is_global to: 1000
Thread: 139835648554752, pid: 1464, addresses: local: 0X628AEDC, global: 0X31B7D
07C
Thread: 139835648554752, incremented this_is_global to: 1001
Thread: 139835656947456, pid: 1464, addresses: local: 0X6A8BEDC, global: 0X31B7D
07C
Thread: 139835656947456, incremented this_is_global to: 1002
After threads, this_is_global = 1002

Now that the threads are done, let's call fork..
Before fork(), local_main = 17, this_is_global = 17
Parent: pid: 1464, lobal address: 0X96114E68, global address: 0X31B7D07C
Child : pid: 1467, local address: 0X96114E68, global address: 0X31B7D07C
Child : pid: 1467, set local_main to: 13; this_is_global to: 23
Parent: pid: 1464, local main = 17, this is global = 17
```

9) Did the value of "this_is_global" change after the threads finished? Why? Yes, the value of "this_is_global" changed because it was modified after joining all the threads. Any alterations made after invoking the join() function affect the global data.

10) Are the local addresses the same in each thread? What about the global addresses? The local addresses in each thread may differ as they possess their own stack memory. However, the global addresses remain the same since threads share the same memory space.

11) Did local_main and this_is_global change after the child process has finished? Why? Yes, after child has done its task, it will wait for the parent to finish. Hence, the parent can change them after child finish its job.

12) Are the local addresses the same in each process? What about global addresses? What happened?

Yes, processes don't share the same memory space. So, they have different memory spaces. Hence, addresses can be the same

Example 5:

13) Run the above program several times and observe the outputs, until you get different results.

```
End of Program. Grand Total = 42111272
nawaf@lamp ~/lab6$ ./Ex5
End of Program. Grand Total = 41567574
nawaf@lamp ~/lab6$ ./Ex5
End of Program. Grand Total = 43656646
nawaf@lamp ~/lab6$ ./Ex5
End of Program. Grand Total = 35246843
nawaf@lamp ~/lab6$ ./Ex5
End of Program. Grand Total = 49313479
nawaf@lamp ~/lab6$ ./Ex5
End of Program. Grand Total = 45244040
nawaf@lamp ~/lab6$
```

14) How many times is the line "tot_items = tot_items + *iptr;" executed? The line is executed 2,500,000 times. With 50 threads, each executing it 50,000 times, the total count amounts to 2,500,000

15) .What values does "*iptr" hold during these executions? "*iptr" points to the data value in the "tidrec" struct specific to each thread. Its value can differ based on the individual execution context of each thread..

16) What What is the expected value of the Grand Total? The Grand Total is anticipated to be 63,750,000, obtained by summing the numbers from 1 to 50 and multiplying the result by 50,000

17) .Why you are getting different results?

The variation in results stems from a race condition in the "tot_items" variable. When all threads concurrently attempt to modify it, the orderly incrementation of its value becomes disrupted. Consequently, the output differs from the expected true result.