King Abdulaziz University

Faculty of Engineering

Department of Electrical and Computer Engineering

Semester: Fall – 2023

EE463 – Operating System

LAB: # 06

Name	ID
Abdulrahman Bakhsh	1936878

Example 1:

```
bakhsh@lamp ~$ gcc Lab6.c -o Lab6 -pthread
bakhsh@lamp ~$ ./Lab6
Parent: My process# ---> 18021
Parent: My thread # ---> 140263707113280
Child: Hello World! It's me, process# ---> 18021
Child: Hello World! It's me, thread # ---> 140263707109120
Parent: No more child thread!
```

-It is the same ID process numbers(PIDs), This is because they are running within the same process.

Example 2:

```
bakhsh@lamp ~$ gcc Lab6.c -o Lab6 -pthread
bakhsh@lamp ~$ ./Lab6
Parent: Global data = 5
Child: Global data was 5.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
bakhsh@lamp ~$
```

-No , there is a race condition between parent and children in accessing the global variable.

-No , it is shared among all threads.

Example 3:

```
bakhsh@lamp ~$ gcc Lab6.c -o Lab6 -pthread
bakhsh@lamp ~$ ./Lab6
I am the parent thread
I am thread #4, My ID #139852326618880
I am thread #9, My ID #139852284655360
I am thread #7, My ID #139852301440768
I am thread #5, My ID #139852318226176
I am thread #3, My ID #139852335011584
I am thread #1, My ID #139852351796992
I am thread #2, My ID #139852343404288
I am thread #0, My ID #139852360189696
I am thread #6, My ID #139852309833472
I am thread #8, My ID #139852293048064
I am the parent thread again
bakhsh@lamp ~$
```

-No , other than the start and end massage of the parent thread the other thread are running concurrently and they don't follow a pre-determined order, that is why we have different result in each run.

Example 4:

```
bakhsh@lamp ~$ gcc Lab6.c -o Lab6 -pthread
bakhsh@lamp ~$ ./Lab6
First, we create two threads to see better what context they share...
Set this_is_global to: 1000
Thread: 139695536015104, pid: 18074, addresses: local: 0X66CCFEDC, global: 0X832
3607C
Thread: 139695536015104, incremented this_is_global to: 1001
Thread: 139695544407808, pid: 18074, addresses: local: 0X674D0EDC, global: 0X832
3607C
Thread: 139695544407808, 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: 18074, global address: 0X8323607C
Child: pid: 18077, local address: 0X5840A1A8, global address: 0X8323607C
Child: pid: 18077, set local_main to: 13; this_is_global to: 23
Parent: pid: 18074, local_main = 17, this_is_global = 17
bakhsh@lamp ~$
```

-Yes, the global variable is incremented by the thread and after the end of the thread we see the +2 increment to the global variable.

-No, The address for each thread is different since each own a separate stack space, but they share the same global address since they don't create a copy of it nut refers to it in both thread.

-No, the fork make its own copy of data in parent and it process it within itself without affecting the original data and stay local within the child memory.

-Yes, the local addresses and global are the same in each process. But each process has its own stack, and the local variables reside in distinct memory locations. That is why they show different result for global and local data.

Example 5 :

```
bakhsh@lamp ~$ ./Lab6
End of Program. Grand Total = 43131815
bakhsh@lamp ~$ ./Lab6
End of Program. Grand Total = 41424827
bakhsh@lamp ~$ ./Lab6
End of Program. Grand Total = 37534032
bakhsh@lamp ~$ ./Lab6
End of Program. Grand Total = 38309201
bakhsh@lamp ~$ ./Lab6
End of Program. Grand Total = 37283999
```

-The line tot_items = tot_items + *iptr; is executed multiple times by each thread. In this case, each thread iterates 50,000 times, so the line is executed 50,000 times per thread.

-The value of *iptr during these executions corresponds to the value of m+1 for each thread. Since m ranges from 0 to 49, *iptr will have values from 1 to 50.

-The expected Grand Total can be calculated by summing the values from 1 to 50,000 for each thread. It can be calculated using the formula for the sum of an arithmetic series: (first term + last term) * number of terms / 2. In this case, the first term is 1, the last term is 50,000, and the number of terms is 50. Therefore, the expected Grand Total should be (1 + 50,000) * 50 / 2 = 1,250,000.

-It is due to a race condition. Multiple threads are concurrently accessing and modifying the tot_items global variable