King Abdul Aziz University
EE-463
Homework #6
4-6-2023

OPERATING SYSTEMS

HOMEWORK #6

Omar alahamdi_1936683

Ex1:

1) Run the above program, and observe its output:

```
omar_alahmadi@lamp ~/lab6$ ./example1
Parent: My process# ---> 828
Parent: My thread # ---> 140226413324096
Child: Hello World! It's me, process# ---> 828
Child: Hello World! It's me, thread # ---> 140226413319936
Parent: No more child thread!
omar_alahmadi@lamp ~/lab6$
```

2) Are the process ID numbers of parent and child threads the same or different? Why?

They are not, though. The program's output demonstrates that each thread has a distinct ID, which is something we already knew.

Ex2:

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

```
omar_alahmadi@lamp ~/lab6$ ./example2
Parent: Global data = 5
Child: Global data was 10.
Child: Global data is now 15.
Parent: Global data = 15
Parent: End of program.
omar_alahmadi@lamp ~/lab6$
```

4) Does the program give the same output every time? Why?

Yes, but there is no assurance that the results will be the same each time. Due to my system's ability to complete the thread's task before the main thread changed the glob_data, it just so occurred to have the identical output. Although it is not guaranteed to occur on every machine, I ran it numerous times, and each time, it did.

5) Do the threads have separate copies of glob_data?

That's incorrect; threads share memory.

Ex3:

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

```
omar_alahmadi@lamp ~/lab6$ ./example3
I am the parent thread
I am thread #2, My ID #140646009583360
I am thread #9, My ID #140645950834432
I am thread #7, My ID #140645967619840
I am thread #5, My ID #140645984405248
I am thread #3, My ID #140646001190656
I am thread #1, My ID #140646017976064
I am thread #0, My ID #140646026368768
I am thread #4, My ID #140645992797952
I am thread #6, My ID #140645976012544
I am thread #8, My ID #140645959227136
I am the parent thread again
```

```
omar_alahmadi@lamp ~/lab6$ ./example3
I am the parent thread
I am thread #4, My ID #140139351746304
I am thread #9, My ID #140139309782784
I am thread #2, My ID #140139368531712
I am thread #8, My ID #140139318175488
I am thread #6, My ID #140139334960896
I am thread #0, My ID #140139385317120
I am thread #3, My ID #140139360139008
I am thread #1, My ID #140139376924416
I am thread #5, My ID #140139343353600
I am thread #7, My ID #140139326568192
I am the parent thread again
omar_alahmadi@lamp ~/lab6$ ./example3
I am the parent thread
I am thread #0, My ID #140408470832896
I am thread #2, My ID #140408454047488
I am thread #1, My ID #140408462440192
I am thread #3, My ID #140408445548288
I am thread #4, My ID #140408437155584
I am thread #5, My ID #140408428762880
I am thread #8, My ID #140408403584768
I am thread #7, My ID #140408411977472
I am thread #9, My ID #140408395192064
I am thread #6, My ID #140408420370176
I am the parent thread again
```

7) Do the output lines come in the same order every time? Why?
No, the order won't be the same each time because the thread's creation time and priority vary.

Ex4:

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

```
omar_alahmadi@lamp ~/lab6$ ./example4
First, we create two threads to see better what context they share...
Set this_is_global to: 1000
Thread: 140111550199552, pid: 868, addresses: local: 0X432DAEDC, global: 0X7A537
07C
Thread: 140111550199552, incremented this_is_global to: 1001
Thread: 140111558592256, pid: 868, addresses: local: 0X43ADBEDC, global: 0X7A537
07C
Thread: 140111558592256, 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: 868, lobal address: 0X54C2FB38, global address: 0X7A53707C
Child : pid: 871, local address: 0X54C2FB38, global address: 0X7A53707C
Child : pid: 871, set local_main to: 13; this_is_global to: 23
Parent: pid: 868, local_main = 17, this_is_global = 17
```

9) Did this_is_global change after the threads have finished? Why?
Yes, as we modified it following each thread's invocation to the join() function.
10) Are the local addresses the same in each thread? What about the global addresses?

The addresses used locally are different. But they are, in fact, global.

11) Did local_main and this_is_global change after the child process has finished? Why?

Yes, the child will wait for the parent to finish after completing its assignment. Thus, once the youngster has completed its task, the parent may alter them.

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

The answer is because processes do not share the same memory. They therefore have unique memory areas. Addresses may therefore be the same.

Ex5:

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

```
omar_alahmadi@lamp ~/lab6$ ./example5
End of Program. Grand Total = 44028195
omar_alahmadi@lamp ~/lab6$
```

```
omar_alahmadi@lamp ~/lab6$ ./example5
End of Program. Grand Total = 43571042
omar_alahmadi@lamp ~/lab6$
```

```
End of Program. Grand Total = 40264492
omar_alahmadi@lamp ~/lab6$ ./example5
End of Program. Grand Total = 39061752
omar_alahmadi@lamp ~/lab6$ ./example5
End of Program. Grand Total = 38964904
omar_alahmadi@lamp ~/lab6$ ./example5
End of Program. Grand Total = 40017909
omar_alahmadi@lamp ~/lab6$
```

14) How many times the line tot_items = tot_items + *iptr; is executed?

Each of the 50 threads that we have runs it 50,000 times. Therefore, 50 times 50,000 is 2,500,000.

15) What values does *iptr have during these executions?

Points to the data's value in the tidrec struct for the thread.

16) What do you expect Grand Total to be?

1 through 50 added together, then multiplied by 50,000 yields 63,750,000.

17) Why you are getting different results?

The tot_items variable has a race condition. The value will not increase in an orderly manner as it should because all threads are attempting to update it simultaneously. Consequently, the result is different and false.