CSC 415 Operating Systems Principles
Fall 2017, Anthony J Souza

HW #4. Processes, Threads & Race Conditions

Assignment Due Date:

Monday October 30th, 2017 @ midnight
Wednesday November1st, 2017 @ midnight for 75% credit

Assignment (50 points). For this assignment you will create a group of threads and then try to coax them into producing ***as many race conditions as possible***. First, your program should declare a single shared integer variable (i.e. a global variable) with initial value zero; then your program should spawn off 10 threads, each of which increments the shared variable by 10, waits for all the threads to complete and then prints out the final value of the global counter. If all threads execute sequentially, then (obviously) the final value of the counter should be 100. Each thread should initially fprintf out to stderr a 'hello I'm thread #x' message, where x is a unique thread id. Then copy the value of the global counter into a local thread variable, fprintf the value of the local thread variable (along with the thread ID number), increment the local thread variable by 10, fprintf the new value of the thread local variable (along with the thread ID number), and finally copy the new thread local value back to the global counter. Since these threads are unsynchronized, there is a possibility of race conditions occurring, depending on the actual interleaved execution order of the various threads. ***Insert nanosleep() commands into your thread code to induce as many race conditions as possible by forcing an (in)appropriate interleaved execution sequence.*** That is, the final value of the global counter printed out by main after the threads complete should be as far away from 100 as possible! As your program executes, it will produce a trace of output from each thread showing the actual interleaved order of execution – annotate this output clearly and completely to explain exactly how and where the race conditions are occurring during execution. ***Implement your solution using Pthreads on Linux***

What to submit:

1. Please copy-n-paste program output and annotate this output as describe above.
2. Please fill in readme.txt file given.
3. Push all completed code to your given repository by the deadline.

How to submit:

- To submit your work please do the following:
  - git add .
  - git commit -m "message"
  - git push