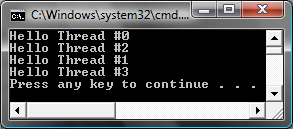
Alberto Martinez

CST407 – Threads

Lab 03

# Activity 1 – “Hello Threads”

This portion of the lab will execute 4 threads that will each print out text and an ID of which thread is printing. This will show that threads execute, and will reveal how threads execute. In this case, the behavior of execution will only be order of execution.



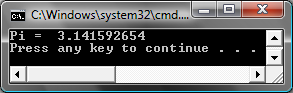
As you can see from the figure above, the threads seem to execute in a random order. The results here are correct and there is nothing wrong, but the order begs the question of what the scheduling for threads is on this particular system.

## Questions

1. The execution order of threads is unpredictable. True or False
   1. True and false. In any OS, there is some sort of threading system which could pin down the execution order of threads, timing, etc. However, it is doubtful that you could say any particular thread execution order and expect it to be correct for a given execution of the process.
2. What build options are required for any threaded software development?
   1. The Runtime libraries used must be the multi-threaded versions. This is significant so that the multi-thread constructs are made available to the programmer need not tip-toe through the specifics of run-time to insure thread integrity.

# Activity 2 – Computing Pi

This portion of the lab will thread the numerical integration to compute pi.



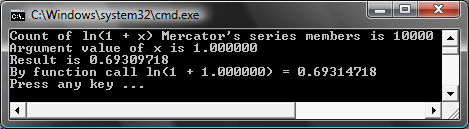
The initial execution of this application shows that pi is calculated correctly. Removing the application’s current model for thread synchronization, the calculation produced a false value. The original method for this calculation was to push calculated values into an array that would then be calculated once all threads were done. The modified method utilized critical sections so that each thread would add up the eventual calculated sum of pi.

## Questions

1. All threads should use the same CRITICAL\_SECTION. True or False
   1. False. A critical section creates an exclusive path of execution which would require other threads to wait on. This may not be optimal when thread count is high and there is no need for a critical section, when another solution could be implemented.
2. Threading errors in software can always be corrected by using only synchronization objects. True or False
   1. False. There are incorrect ways to use even synchronization objects. The importance is a programmers understanding of the problem, and proper implementation of a solution.
3. CRITICAL\_SECTION object should always be declared as global variables. True or False
   1. False. I am not completely sure about this, but the reading I did pointed to examples where there were nested critical sections. This does not prove that critical sections be all global, but it’s a strong case for locally declared critical sections.

# Activity 3 – Using Events

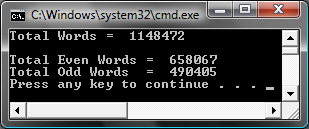
This portion of the lab uses windows events to signal thread completion. The first execution of this program uses threads only. I will modify the application to use events.



# Activity 4 – Using Semaphores

This portion of the lab will use semaphores to control access to a file that multiple threads will be getting data from. Using a semaphore, these threads will take turns pulling data from the file to prevent loss of data.

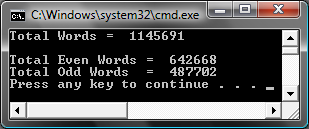
## Serial Program



This is what a successful run will yield.

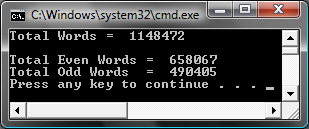
## Threaded Program

### Unmodified Run



The threads here are not blocked from pulling data from the file, and so we lose data.

### Modified Run



Success