

Concurrency

Estimated time for completion: 1 hour

Overview:

Use Threadpool and Threading.

Goals:

* Enter goal(s) for this lab.

Lab Notes:

Enter any special setup-issues etc for the lab..

Implement Parallel Block with Threadpool

The project Concurrency1 calls a Process three times serially. The Process takes a long time to complete. Instead, we wish to call all three in parallel, thus reducing the total time spent waiting for them to complete.

Criteria:

* Enter criteria for the part here.

Notes:

* Enter any relevant notes for the part here.

Steps:

1. Declare a delegate that matches the signature of the DoIt method.
2. Create three delegates that point to the DoIt methods.
3. Execute all delegates using BeginInvoke.
4. Use WaitHandle.All to wait for all the delegates to complete.
5. Fetch the answers from each delegate using EndInvoke.
6. Run the program several times. You should see a substantial performance improvement.

Solutions:

Enter path(s) to solution file(s) local to the part here.

Coordinating threads with Events

In Concurrency2, we have a Main thread and a Generator thread. The Generator thread will update the field “number” with 0 to 9. The Main thread will read the variable “number” and print out the number it finds. Run the program a few times and you should see it fails to print out most numbers. The reason is that the Generator thread loops several times before the Main thread has a chance to read it. Modify the program to print out the numbers 0 to 9.

Steps:

1. Use ManualResetEvents to control access to the number variable. The task is to prevent the Generator from overwriting the number variable before the Main thread has a chance to read it.
2. You will only need to add calls to the ManualResetEvent in the right places. You won’t need to modify the program in any other way.

Solutions

Enter path(s) to solution file(s) here.