Hidden gotcha in the thread pool sample program on MSDN

devblogs.microsoft.com/oldnewthing/20151202-00

December 2, 2015

Raymond Chen

There's a hidden gotcha in the MSDN thread pool sample that one of our interns stumbled across.

"I am trying to create a simple thread pool rather than creating a new thread for each task I want to perform. I based this program on the MSDN thread pool sample, but I found that the work items never run in parallel. They always run sequentially. All calls succeed. Can anybody explain why this is happening? Thanks."

```
#include <windows.h>
#include <iostream>
#include <cstdlib>

VOID
CALLBACK
Callback(
    PTP_CALLBACK_INSTANCE Instance,
    PVOID                 /* Parameter */,
    PTP_WORK              /* Work */
    )
{
    std::cout << "Starting " << Instance << std::endl;
    Sleep(3000); // Pretend to do work
    std::cout << "Ending " << Instance << std::endl;
}

int
__cdecl
main(int, char**)
{
    TP_CALLBACK_ENVIRON CallBackEnviron;

    InitializeThreadpoolEnvironment(&CallBackEnviron);

    auto pool = CreateThreadpool(NULL);

    SetThreadpoolThreadMaximum(pool, 1);

    SetThreadpoolThreadMinimum(pool, 1);

    auto cleanupGroup = CreateThreadpoolCleanupGroup();

    SetThreadpoolCallbackPool(&CallBackEnviron, pool);


    SetThreadpoolCallbackCleanupGroup(&CallBackEnviron,
                                      cleanupGroup,
                                      NULL);


    auto work = CreateThreadpoolWork(Callback,
                                NULL,
                                &CallBackEnviron);

    for (int i = 0; i< 100; i++) {
        SubmitThreadpoolWork(work);
    }

    CloseThreadpoolCleanupGroupMembers(cleanupGroup,
                                       FALSE,
                                       NULL);
```

```
    CloseThreadpoolCleanupGroup(cleanupgroup);

    CloseThreadpool(pool);

    return 0;
}
```

The tasks all run sequentially because of these two lines:

```
    SetThreadpoolThreadMaximum(pool, 1);
    SetThreadpoolThreadMinimum(pool, 1);
```

If you set the minimum and maximum thread counts both to one, then that means that the thread pool consists of a single permanent thread. This really isn't much of a thread pool any more, although I guess it gives you the convenience of being able to add work to it relatively easily.

This hidden gotcha was called out in the sample where it says "The pool consists of one persistent thread." Mind you, it says so in a rather unobtrusive place, so I don't blame you for missing it.

If you want to allow tasks to run in parallel, remove the call to `SetThreadpoolThread-Maximum`, or at least set the maximum to more than one. While you're at it, remove the call to `SetThreadpoolThreadMinimum`, since there is nothing in this sample that requires that the threads be persistent. (If there is no work queued on the thread pool, we should let the thread pool destroy all its threads.)

And while I understand that this was just an intern playing around with a sample program, it should be called out that in general, you don't need to create your own thread pool. Just use the system one, and use a cleanup group if you want to be able to do bulk cancellation.

Raymond Chen

Follow