How can I detect that a thread pool work item is taking too long?

devblogs.microsoft.com/oldnewthing/20220110-00

January 10, 2022

Raymond Chen

A customer wants to detect that their thread pool work items are not completing quickly enough and trigger a crash if they appear to be stuck. They looked at `WaitForThreadpool-WaitCallbacks`, but that function doesn't support a timeout. It just waits indefinitely. Is there a way to wait with a timeout?

Even though there is no way to wait with a timeout, it turns out that in this particular case, we don't need one. Since all we want to do is crash, it doesn't matter who does the crashing!

- Create a watchdog timeout that triggers after a timeout.
- Call `WaitForThreadpoolWaitCallbacks`.
- Cancel the watchdog timer.

If the watchdog timer fires, then the `WaitForThreadpoolWaitCallbacks` got stuck, and the watchdog timer handler can log the failure or trigger the crash.

Now, if you want a way to abandon the wait and keep running, then you can roll that yourself: You can associate an event (possibly a lightweight one you can use with `WaitOn-Address`), and you can wait for the event with a timeout. Meanwhile, when the work item is finished, the last thing it does is signal the event.

Bonus chatter: There is a `SetEventWhenCallbackReturns` function which tells the thread pool to set an event when the callback returns. However, an RAII class will do the job nicely in this case, and the RAII class will let you use a lighter-weight synchronization object. The `SetEventWhenCallbackReturns` function's primary purpose is to allow you to know when it's safe to unload the code running the callback, because the event is set after control has left the callback.

Raymond Chen

Follow