Once you return from the WM_ENDSESSION message, your process can be terminated at any time

devblogs.microsoft.com/oldnewthing/20130627-00

June 27, 2013

Raymond Chen

A customer had a program which performed some final I/O operations as it exited. Various C++ objects deleted files or flushed buffers as part of their destructors. The customer found that if their program was left running when the user shut down Windows, then the files never got deleted, and the buffers were never flushed. On the other hand, if they inserted an artificial delay into the shutdown procedure, so that it waited ten seconds after the program exited before continuing with shutdown, then the files did indeed get cleaned up and the buffers were indeed flushed. The customer confirmed that the program did receive the `WM_ENDSESSION` message, but it appeared as if all disk I/O issued within five seconds of shutdown never gets committed to disk. This would appear to be a serious bug in Windows.

Because, of course, when you find a problem with your program, your first reaction should be to assume that you found a bug in Windows so blatant it should be affecting every program on the planet, and yet somehow this horrific data loss bug eluded not only the entirety of the Windows QA team, but also every software developer for the past twenty years who had a program that saved data at shutdown.

Or the problem could be in your code.

The documentation for the `WM_ENDSESSION` message says,

> ***wParam***
> If the session is being ended, this parameter is **TRUE**; the session can end any time after all applications have returned from processing this message.

What is much more likely to be happening is that when the application receives the `WM_ENDSESSION` message, it posts a message to itself to initiate controlled shutdown. After the program returns from the `WM_ENDSESSION` message, the message pump picks up the shutdown message and it is at this point that the program starts cleaning up objects, including running destructors and flushing buffers, and then finally calling `ExitProcess`.

In other words, the problem is not that the final I/O never got committed to disk. The problem is that the final I/O *was never issued by the program*. Once your program returns from the `WM_ENDSESSION` message, Windows has the right to terminate it without further warning. If your system shuts down quickly, that termination may occur before your destructors manage to run at all.

You cannot rely on any code in your program running once you have responded to the `WM_ENDSESSION` message. That message is your "final warning". If you need to do cleanup operations before termination, you need to do them *before* returning from the `WM_ENDSESSION` message. Because once you return from that message, your process is living on borrowed time.

Raymond Chen

Follow