Ready… cancel… wait for it! (part 2)

devblogs.microsoft.com/oldnewthing/20110203-00

February 3, 2011

Raymond Chen

A customer had a question about I/O cancellation. They have a pending `ReadFileEx` call with a completion procedure. They then cancel the I/O with `CancelIoEx` and wait for the completion by passing `TRUE` as the `bWait` parameter to `GetOverlappedResult`.

> Assuming both return success, can I assume that my completion procedure will not be called after GetOverlappedResult returns? It appears that GetOverlappedResult waits non-alertably for the I/O to complete, so I'm assuming it just eats the APC if there was one. But if an APC had been posted just before I called CancelIoEx, will it also cancel that APC?

`GetOverlappedResult` does not magically revoke completion callbacks. Why should it? Recall that completion is not the same as success. Completion means that the I/O subsystem has closed the books on the I/O operation. The underlying operation may have completed successfully or it may have failed (and cancellation is just one of the many possible reasons for failure). Either way, the completion procedure signed up to be notified when the I/O completes, and therefore it will be called to be informed of the completion due to cancellation. Besides, as the customer noted, there is a race condition if the `CancelIoEx` call is made just after the I/O completed, in which case it didn't get cancelled after all. This answers our question from last time, namely, how our fix for the cancellation code was incomplete. If the I/O had been issued with a completion routine (or equivalently, if it had been issued against an I/O completion port), then the code frees the `OVERLAPPED` structure before the completion routine runs. The kernel doesn't care that you did that (the kernel is finished with the `OVERLAPPED` structure), but your completion routine is probably not going to be happy that it was given a pointer to freed memory as its `lpOverlapped` parameter.

You have to delay freeing the `OVERLAPPED` structure until the completion routine executes. Typically, this is done by allocating the `OVERLAPPED` structure on the heap rather than the stack, and making it the completion routine's responsibility to free the memory as its final act.

Raymond Chen

Follow