Rules can exist not because there's a problem, but in order to prevent future problems

devblogs.microsoft.com/oldnewthing/20150826-00

August 26, 2015

Raymond Chen

I lost the link, but one commenter noted that the `ReadFile` function documentation says

> Applications must not read from, write to, reallocate, or free the input buffer that a read operation is using until the read operation completes.

The commenter noted, "What is the point of the rule that disallows reading from or writing to the input buffer while the I/O is in progress? If there is no situation today where this actually causes a problem, then why is the rule there?"

Not all rules exist to address current problems. They can also exist to prevent future problems.

In general, you don't want the application messing with an I/O buffer because the memory may have been given to the device, and now the device has to deal with bus contention. And there isn't really much interesting you can do with the buffer before the I/O completes. You can't assume that the I/O will complete the first byte of the buffer first, and the last byte of the buffer last. The I/O request may get split into multiple pieces, and the individual pieces may complete out of order.

So the rule against accessing the buffer while I/O is in progress is not a significant impediment in practice because you couldn't reliably obtain any information from the buffer until the I/O completed. And the rule leaves room for the future versions of the operating system to take advantage of the fact that the application will not read from or write to the buffer.

Tomorrow, I'll tell a story of a case where accessing the I/O buffer before the I/O completed really did cause problems in Windows 95.

Raymond Chen

Follow