

Re: Shared memory and IOCP

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2007-03/msg00127.htm>

- *From:* Anton Bassov <AntonBassov@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 12 Mar 2007 07:23:33 -0700
-

BTW: you wouldn't need a window either – PostThreadMessage works well

IIRC, WM_COPYDATA has to be processed synchronously, i.e. you have to send it only with SendMessage(). Once PostThreadMessage() is asynchronous, it does not seem to be the right option here. Actually, synchronous processing in itself is quite desirable in this situation – message sender can be sure that, by the time SendMessage() has returned, the recipient had already consumed/modified data in the buffer, so that there is no need to worry about synchronizing access to the buffer. However, if PostMessage()/PostThreadMessage() is used, the additional steps to synchronize an access to the buffer have to be taken.

Anton Bassov

"m" wrote:

BTW: you wouldn't need a window either – PostThreadMessage works well

"Anton Bassov" <AntonBassov@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message <news:B78D6046-F291-4325-85F7-15609F5791F5@xxxxxxxxxxxxxxxxxxxx>

Well, I think we speak about two completely separate domains. :-)

Not necessarily – it depends on how you want to do things.....

WM_COPYDATA is a convenient mechanism for passing simple messages between windows apps. I need a high throughput local communication layer for bulk transfers close to the peak performance of the memory subsystem.

Re: Shared memory and IOCP

From the very beginning it was obvious that you are looking for some possibility of asynch communication, and windows messages is really easy way to achieve what you want. After all, you don't really need to pass the actual data with this message, do you??? You can use it simply for notification – when process X receives this message, it has to respond to it by reading(or writing) memory of the process Y. Once WM_COPYDATA message allows you to pass some data along, you can pass the ID of process Y plus the address of the buffer (as it is known to the process Y) to the recipient process X with this message. At this point process X will read/write data from/to the buffer with ReadProcessMemory() / WriteProcessMemory(), and that's it....

Not to mention that my programs do not even have GUI to interpret the WM_ messages, they are big completion port-based, fully asynchronous multithreaded servers.

Actually, you don't really need full-fledged GUI here – the only thing you need is an invisible dummy window with a message loop....

Anton Bassov

"Piotr Wyderski" wrote:

Anton Bassov wrote:

What about WM_COPYDATA message?
Will it work for you?

Well, I think we speak about two completely separate domains. :-)
WM_COPYDATA is a convenient mechanism for passing simple messages between windows apps. I need a high throughput local communication layer for bulk transfers close to the peak performance of the memory subsystem. Not to mention that my programs do not

Re: Shared memory and IOCP

even have GUI to interpret the WM_ messages, they are big completion port-based, fully asynchronous multithreaded servers.

I have all the necessary components: shared memory provides enough bandwidth and the IOCP model provides high scalability.

I'm just looking for a way to match these interfaces using some nifty

trick, since the obvious solution, i.e.

PostRemoteQueuedCompletionStatus(), does not exist on XP and 2003 (don't know much about Vista).

Best regards
Piotr Wyderski