Isn't the CompletionKey parameter to CreateIoCompletionPort superfluous?

devblogs.microsoft.com/oldnewthing/20130222-00

February 22, 2013

Raymond Chen

When you associate a file handle with an I/O completion port with the `CreateIoCompletionPort` function, you can pass an arbitrary pointer-sized integer called the `CompletionKey` which will be returned by the `GetQueuedCompletionStatus` function for every I/O that completes against that file handle. But isn't that parameter superfluous? If somebody wanted to associated additional data with a file handle, they could just extend the `OVERLAPPED` structure to contain that additional data.

Yes, they could, so in a purely information-theoretical sense, the parameter is superfluous. And heated seats in your car are superfluous, too. But they sure are nice!

From a purely information-theoretical point of view, a lot of things are superfluous. The `GWLP_USERDATA` window bytes are not necessary, because you could just put the information in the window extra bytes. And window extra bytes are superfluous, since you could have just put them in properties. And properties are superfluous, since you could just have a hash table which maps window handles to "all the other data I want to associate with this window handle."

But it's nice to have `GWLP_USERDATA`.

I find it interesting that people complain when Windows does not provide a convenience for some operation, and here's a case where it provides one, and then people complain that it's wasteful!

It can be nice to have some information associated with the file handle (to record some general information, like the overall operation this file is responsible for) and different information associated with the I/O request (to record some specific information, like which phase of the operation most recently completed). That way, you don't have to try to pack the two pieces of information together.

A more practical reason is that you may not be able to pass the extended `OVERLAPPED` through to the `ReadFile` . For example, you may be calling another function which will in turn issue the `ReadFile`, and that other function builds its own `OVERLAPPED` structure rather than letting you pass one in. In that case, you will thank your lucky stars that there's some redundant data elsewhere that will let you recover your state.

Raymond Chen

Follow