The OVERLAPPED associated with asynchronous I/O is passed by address, and you can take advantage of that

devblogs.microsoft.com/oldnewthing/20101217-00

December 17, 2010

Raymond Chen

When you issue asynchronous I/O, the completion function or the I/O completion port receives, among other things, a pointer to the `OVERLAPPED` structure that the I/O was originally issued against. And that is your key to golden riches.

If you need to associate information with the I/O operation, there's no obvious place to put it, so some people end up doing things like maintaining a master table which records all outstanding overlapped I/O as well as the additional information associated with that I/O. When each I/O completes, they look up the I/O in the master table to locate that additional information.

But it's easier than that.

Since the `OVERLAPPED` structure is passed by address, you can store your additional information *alongside* the `OVERLAPPED` structure:

```
// in C
struct OVERLAPPEDEX {
 OVERLAPPED o;
 CClient *AssociatedClient;
 CLIENTSTATE ClientState;
};
// or in C++
struct OVERLAPPEDEX : OVERLAPPED {
 CClient *AssociatedClient;
 CLIENTSTATE ClientState;
};
```

When the I/O completes, you can use the `CONTAINING_RECORD` macro or just `static_cast` the `LPOVERLAPPED` to `OVERLAPPEDEX*` and bingo, there's your extra information right there. Of course, you have to know that the I/O that completed is one that was issued against an `OVERLAPPEDEX` structure instead of a plain `OVERLAPPED` structure, but there are ways of keeping track of that. If you're using a completion function, then only use an `OVERLAPPEDEX` -aware completion function when the `OVERLAPPED` structure is part of an `OVERLAPPEDEX` structure. If you're using an I/O completion port, then you can use the completion key or the `OVERLAPPED.hEvent` to distinguish `OVERLAPPEDEX` asynchronous I/O from boring `OVERLAPPED` I/O.

Raymond Chen

Follow