Why bother with RegisterWaitForSingleObject when you have MsgWaitForMultipleObjects?

devblogs.microsoft.com/oldnewthing/20081117-00

November 17, 2008

Raymond Chen

Commenter kokorozashi wonders why you should bother with `RegisterWaitForSingleObject` when you have `MsgWaitForMultipleObjects` already. If you want to pump messages and wait for a kernel object, then you can change all calls to `PeekMessage`, `GetMessage`, and `WaitMessage` to replacement functions that use `MsgWaitForMultipleObjects`. Isn't that enough? Why waste an entire thread just to wait for that object?

If you're so clever that you can modify every call to `PeekMessage`, `GetMessage`, and `WaitMessage`, then more power to you. But in order to do this, you'll have to restrict the functions you call, because all sorts of functions contain their own message loops. Do you call `MessageBox`? Or `DialogBox`? Those functions contain a modal loop. (After all, they don't return until the user dismisses the dialog box; *somebody* has to be pumping messages because you're not.) Indeed, you can't even call `DefWindowProc` since that function goes into a modal loop if the user, say, grabs the caption bar and drags the window around: That drag loop happens inside `DefWindowProc`.

If your thread has any sort of visible UI, this sort of extreme control of all message loops is unreasonable. You have no practical choice but to have the wait happen on some other thread and either respond to the signalled object on that thread or post a notification back to the UI thread to do the work.

The advantage of `RegisterWaitForSingleObject` over creating your own thread for waiting is that the thread pool functions will combine multiple registered waits together on a single thread (by the power of `WaitForMultipleObjects`), so instead of costing a whole thread, it costs something closer to (but not exactly) 1/64 of a thread.

Raymond Chen

Follow