

Re: WaitForSingleObject() will not deadlock

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-07/msg00153.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Mon, 02 Jul 2007 01:44:06 -0400
-

But the issue is how can you lock and traverse a potentially circular list with "node has been visited" detection without a recursive mutex?

I remember the pain we had to go through in the 1968-1972 era, when we were still inventing concepts like this, to create concurrent list-walkers in the presence of circular lists, and it wasn't easy. We even used a question like this on a PhD qualifier exam, it was considered so difficult. But recursive mutexes made the problem trivial.

Note the reason Alertable Wait State is required for I/O callbacks is so that the callback doesn't look like an interrupt.

Yes, worrying about the lock count could be an issue, but using exceptions to maintain invariants does avoid this problem.

The cost of supporting recursion led to the Kernel concept of a "fast mutex" which is used when recursion is not required or expected.
joe

On Sun, 01 Jul 2007 19:54:12 -0500, "Doug Harrison [MVP]" <dsh@xxxxxxxxx> wrote:

On Sun, 1 Jul 2007 17:01:07 -0700, "Alexander Grigoriev" <alegr@xxxxxxxxxxxxx> wrote:

Anybody tell me, why any other variants besides a recursive one were invented?

If I may turn the question around, here are some arguments against recursion:

1. Recursion makes it easy to hold a mutex across function calls and may even encourage it, which is at odds with the goal of holding a mutex for as short a time as possible.
2. People who routinely take advantage of recursive mutexes tend to develop lax locking protocols and may end up writing code that has subtler races than they'd get had they been limited to non-recursive mutexes.

Re: WaitForSingleObject() will not deadlock

3. The possibility of recursion makes it harder to implement algorithms that requiring releasing the mutex. For example, a thread that holds a mutex and needs to release it can't just unlock it, but must instead unlock it the correct number of times, which in Windows means keeping track of the lock count itself, since you can't query this property. (I wouldn't feel comfortable calling ReleaseMutex until it fails, and LeaveCriticalSection returns void.)

4. This is a relatively minor concern, but supporting recursion is more expensive than not supporting it.

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm

.