

Re: WaitForSingleObject() will not deadlock

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-07/msg00149.html>

- *From:* "Alexander Grigoriev" <alegr@xxxxxxxxxxxxx>
 - *Date:* Sun, 1 Jul 2007 21:37:36 -0700
-

My guess that recursive mutexes may be handy to work on some tree-like structures. File system drivers may be using them heavily.

"Doug Harrison [MVP]" <dsh@xxxxxxx> wrote in message news:66hg83t2evoskiqgf9pcb7clf6fr1rk8k@xxxxxxxxxx

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.
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.

Re: WaitForSingleObject() will not deadlock

--
Doug Harrison
Visual C++ MVP