Is it okay to call TryAcquireSRWLock from a thread that has already acquired the lock?

devblogs.microsoft.com/oldnewthing/20160819-00

August 19, 2016

Raymond Chen

A customer found the MSDN documentation ambiguous. Wouldn't be the first time.

In the description of Slim Reader/Writer Locks:

> An SRW lock is the size of a pointer. The advantage is that it is fast to update the lock state. The disadvantage is that very little state information can be stored, so SRW locks cannot be acquired recursively. In addition, a thread that owns an SRW lock in shared mode cannot upgrade its ownership of the lock to exclusive mode.

The ambiguity is over the word "cannot", used twice in the above paragraph. Does it mean "The system will not allow an SRW lock to be acquired recursively or to be upgraded from shared to exclusive"? Or does it mean "If you attempt to acquire an SRW lock recursively, or if you try to upgrade an SRW lock from shared to exclusive, then the result is undefined"?

The question was directed more specifically at the `Try` variants. If you "try" to acquire an SRW lock that the thread has already acquired, does the system detect this and cause the "try" to fail? Or is it a programming error?

It's a programming error. It is your responsibility as a programmer not to call `AcquireSRWLockShared` or `AcquireSRWLockExclusive` from a thread that has already acquired the lock. Failing to comply with this rule will result in undefined behavior.

All the rules that apply to `AcquireSRWLock*` also apply to `TryAcquireSRWLock*`. The only difference between the two is that if the lock cannot be acquired, the regular `AcquireSRWLock*` functions will block until the lock is acquired, whereas the `TryAcquireSRWLock*` functions will return immediately and say, "Sorry. I would have blocked, but you asked me not to block."

I'll see what I can do to make this clearer in the documentation.

Raymond Chen

Follow