Is it okay to acquire an SRWLOCK recursively? (And why not?)

devblogs.microsoft.com/oldnewthing/20160506-00

May 6, 2016

Raymond Chen

A customer was using an `SRWLOCK` to protect access to an object. All functions that use the object acquire the `SRWLOCK` in shared mode, and the function that destroys the object acquires the `SRWLOCK` in exclusive mode.

> This seems correct to us, but we found that when running under the Application Verifier, we receive many errors complaining that our code is recursively acquiring the `SRWLOCK`. This happens, for example, when a function acquires the `SRWLOCK` in shared mode, and then calls another function which also acquires the `SRWLOCK` in shared mode. We completely understand why Application Verifier might warn of recursive *exclusive* lock acquisition, but why does it also complain about recursive *shared* lock acquisition?
>
> Before we dig in and try to fix this, can you confirm that is a real problem? Or is this an oversight in Application Verifier?

Application Verifier is correct to complain. As noted in the documentation, `SRWLOCK` objects cannot be acquire recursively. This applies both to shared and exclusive acquisition.

The technical reason is that the `SRWLOCK` was designed to be fast and require no dynamic memory allocation. In order to accomplish this, many potential features had to be sacrificed, among them, recursive acquisition and lock promotion from shared to exclusive.

If you want a synchronization object that supports recursive acquisition, you might want to try a `CRITICAL_SECTION`, or build your own data structure around `SRWLOCK` that also keeps track of each thread's recursive acquisition count.

The customer replied,

> Okay, so it's clear that we need to fix this. Our next question, then, is how urgent do we need to deploy this fix? Is this an actual broken scenario, or is it merely a theoretical possibility? In other words, do we need to issue a patch for it right now, or can we wait until our next major version?

Two of my colleagues shared their experiences:

> We encountered this issue in our own product. The conclusion of the investigation was that this is a critical error if recursive acquisition is indeed occurring.

> We hit a deadlock in production due to erroneous recursive acquisition. It is fiendishly difficult to debug. I would put it in the “immediate fix” category.

The customer thanks us for the information and began working on a fix.

Raymond Chen

Follow