On the gradual improvements in how the system deals with the failure to initialize a critical section

devblogs.microsoft.com/oldnewthing/20171020-00

October 20, 2017

Raymond Chen

The documentation for the `InitializeCriticalSection` function says

> Return value
>
> This function does not return a value.
>
> **Windows Server 2003 and Windows XP**: In low memory situations, **InitializeCriticalSection** can raise a `STATUS_NO_MEMORY` exception. This exception was eliminated starting with Windows Vista.

In earlier versions of Windows, the `InitializeCriticalSection` function could fail in low memory conditions, in which case it raised a `STATUS_NO_MEMORY` exception.

Wait, let's go back even further.

In very old versions of Windows, the `InitializeCriticalSection` function could fail in low memory conditions, in which case it raised a `STATUS_NO_MEMORY` exception. However, the code wasn't particularly careful about exactly when it raised the exception, and it turns out that it didn't bother to completely unwind the partial-initialization-so-far before raising the exception. This means that if a program tried to recover from a failed `InitializeCriticalSection` by catching the `STATUS_NO_MEMORY` exception, it still experienced a memory leak.

Yes, it's rather ironic that if the kernel couldn't initialize the critical section due to low resources, it leaked memory, which made the low resource situation *even worse*.

There was a similar sad story with `EnterCriticalSection` and even `LeaveCriticalSection` : Under low resource conditions, those functions could fail and raise a `STATUS_NO_MEMORY` exception. Those are even worse because by the time you get the exception, it's probably too late to back out of whatever you were doing. I mean, maybe if

you're really clever, you can recover from a failed `EnterCriticalSection` by abandoning the operation (and undoing all the work done so far), but I can't think of any case where a program could do anything reasonable if `LeaveCriticalSection` fails.

And as Michael Grier noted, if `LeaveCriticalSection` raised an exception, not only wasn't there anything you could reasonably do about it, but it also left the critical section in a corrupted state!

The only thing you can do is to just crash the process before things get any worse.

I think it was in Windows XP that the kernel folks fixed the code so that it cleaned up the partially-initialized critical section before raising the `STATUS_NO_MEMORY` exception, so that a program could safely catch the exception and not leak memory. I believe they also fixed it so that the `EnterCriticalSection` and `LeaveCriticalSection` functions would not raise exceptions. If called properly, then they always succeeded. So at least those weird cases of "raising an exception and leaving the critical section fatally corrupted" went away.

And then in Windows Vista, the kernel folks decided to get rid of the problem once and for all and remove all the failure cases from all the critical section functions. The `InitializeCriticalSection` and `InitializeCriticalSectionAndSpinCount` functions always succeeded. The `EnterCriticalSection` and `LeaveCriticalSection` functions would not raise exceptions when used properly.

So for over a decade now, the `InitializeCriticalSection` and `InitializeCriticalSectionAndSpinCount` functions never fail. This means that (assuming they are called properly), `InitializeCriticalSection` never raises a `STATUS_NO_MEMORY` exception. The `InitializeCriticalSectionAndSpinCount` function a return value that says whether it succeeded, but it always succeeds and returns a nonzero value. The return value is now superfluous.

Bonus chatter: The documentation for the `InitializeCriticalSectionAndSpinCount` function says

> Return value
>
> This function always returns a nonzero value.
>
> **Windows Server 2003 and Windows XP**: If the function succeeds, the return value is nonzero. If the function fails, the return value is zero (0). To get extended error information, call **GetLastError**. This behavior was changed starting with Windows Vista.

I'm told that some people come away from the documentation still worried about the possibility that the `InitializeCriticalSectionAndSpinCount` might fail on Windows Vista and later. They see that on Windows Server 2003 and Windows XP, the function tells you whether or not it succeeded, but on Windows Vista it always reports success. "That

means that if the function fails, Windows Vista will lie to me and report success even though it failed!" No, that's not what it's saying. It's saying that starting in Windows Vista, the function *never fails*.

Raymond Chen

Follow