What’s so special about the number 64 when it comes to TLS slots?

devblogs.microsoft.com/oldnewthing/20160613-00

June 13, 2016

Raymond Chen

Last time, we ended with the question, “What’s so special about the number 64?” when discovering that a program crashed if it ever got a TLS slot index greater than or equal to 64.

Versions of Windows prior to Windows 2000 supported up to 64 TLS slots. This was codified in the constant

```
#define TLS_MINIMUM_AVAILABLE 64
```

but even back then, it was noted in the documentation that 64 was merely the minimum. According to my really ancient copy of the Win32 SDK documentation

> The constant `TLS_MINIMUM_AVAILABLE` defines the minimum number of TLS indices available in each process. This minimum is guaranteed to be at least 64 for all systems.

(I found a site that has something very close to the original documentation.)

Somehow, this got misinterpreted as “The constant `TLS_MINIMUM_AVAILABLE` defines the maximum number of TLS indices available in each process.”

One theory is that the code was originally written back in the days when the actual limit was indeed 64, and the code was written based on the implementation rather than the documentation. (Possibly because they either got a source code license or reverse-engineered the function and observed that `TlsAlloc` always returned a value less than 64.)

I’m not entirely convinced of this theory because the maximum number of TLS slots increased in Windows 2000, but the program in question was released in 2007.

On the other hand, this theory could still be valid the program was using a library that was originally written pre-2000. Even though the program itself was written after 2000, parts of it were written before 2000, back when the TLS limit was 64.

Another theory is that somebody “heard from somewhere” that TLS slots will never go higher than 63, and they simply believed it.

Maybe you can come up with your own theory. Share it in the comments.

Raymond Chen

Follow