Windows 95 and Windows NT manage the TEB on x86 differently

devblogs.microsoft.com/oldnewthing/20150626-00

June 26, 2015

Raymond Chen

Today, a historical note of no consequence. Just a story.

The Win32 x86 ABI specifies that the FS register holds a selector which is based at the current thread's TEB. In other words, `fs:[n]` is the *n*th byte of the `TEB`.

It so happens that the two operating systems chose to manage the FS register differently.

Windows 95 gave each TEB in the system its own selector.

Windows NT allocated a single selector to represent the TEB, and each time the processor changed threads, the selector base was updated to match the TEB for the new thread. With this model, every thread has the same value for FS, but the selector's descriptor kept changing.

It's as if you had a car-rental service, and one of the features of the service is that the radio remembers your presets. The instructions for setting the radio are as follows:

- Enter the four-digit customer preferences ID printed on your receipt.
- Your radio is now set to your preferences.

There are two ways you could set up this system.

Windows 95 assigns each customer a unique customer preferences ID and prints it on the receipt. When the customer enters the ID, the radio looks up the ID in a database and applies the user's radio preferences.

Windows NT prints the same preferences ID on every receipt: 1234. Before the customer picks up the car from the rental service, the service agent sets the radio to the customer's preferences. When the customer enters the ID, the radio does nothing (aside from printing an error message if you enter anything other than 1234).

Even though the Windows NT way creates more work for the service agent, it does solve an important problem: It lets your service scale to more than 10,000 customers, for once you have 10,001 customers, you cannot assign each of them a unique four-digit ID any more.

Car	Windows
car	processor
customer	thread
radio preferences	TEB
customer ID	selector

By assigning a unique selector to each TEB, Windows 95 limited itself to at most 8192 threads. (In practice, the limit was much lower because selectors were used for other things, too.) This was not an issue in practice because Windows 95 would run into other problems long before you ran into the 8192-thread limit.

But Windows NT was designed to be scalable to very large workloads, and they couldn't artificially limit themselves to a maximum of 8192 threads.

The consequences of changing the meaning of the FS register at every thread switch means that some tricks that happened to work in Windows 95 didn't work in Windows NT. For example, in Windows 95, if you captured the value of the FS register in one thread, you could use it (in violation of the ABI) on another thread in the same process and still access the originating thread's TEB. If you tried this trick on Windows NT, you would just see your own TEB.

In the analogy, it's as if you copied the customer preferences ID from another customer's receipt and tried to use it in your car. In a Windows NT car, you would just get your own preferences again, because every receipt has the same customer preferences ID printed on it.

According to the Win32 ABI for x86, the only thing you can do with the FS register is dereference it to access your TEB. If you try to fiddle with its value or try to copy its value somewhere, you are off in unsupported territory, and the resulting behavior is undefined.

Raymond Chen

Follow