Exported functions that are really forwarders

devblogs.microsoft.com/oldnewthing/20060719-24

July 19, 2006

Raymond Chen

Last time, we saw how the way Win32 exports functions is pretty much the same as the way 16-bit Windows exports functions, but with a change in emphasis from ordinal-based exports to name-based exports. This change in emphasis is not expressed anywhere in the file format; both 16-bit and 32-bit DLLs can export either by name or by ordinal (or by both), but the designers of Win32 were biased in spirit in favor of name-only exports.

But there is a new type of exported function in Win32, known as a forwarder. A forwarder looks just like a regular exported function, except that the entry in the ordinal export table says, "Oh, I'm not really a function in this DLL. I'm really a function in that DLL over there." For example, if you do a `link /dump /exports kernel32.dll`, you'll see a line like this:

```
151   EnterCriticalSection (forwarded to NTDLL.RtlEnterCriticalSection)
```

This means that if a program links to `KERNEL32.EnterCriticalSection`, the loader silently redirects it to `NTDLL.RtlEnterCriticalSection`. Forwarders are a handy way to accommodate functionality moving from one DLL to another. The old DLL can continue to export the function but forward it to the new DLL.

The forwarding trick is actually better than just having a stub function in the old DLL that calls the function in the new DLL, because the stub function creates a dependency between the old DLL and the new one. (After all, the old DLL needs to be linked to the new DLL in order to call it!) With a forwarder, however, the new DLL is not loaded unless somebody actually asks for the forwarded function from the old DLL. As a result, you don't pay for the new DLL until somebody actually wants it.

Okay, we saw that with forwarders, Win32 has diverged from 16-bit Windows, but when it comes to imports, it's a whole new ball game. We'll pick up the story next time.

Raymond Chen

Follow