How are window manager handles determined in Windows NT?

devblogs.microsoft.com/oldnewthing/20070717-00

July 17, 2007

Raymond Chen

We left off our story last time by raising the problem of programs that send messages to windows that have already been destroyed and how window handle re-use exacerbates the problem. Although this is clearly a bug in the programs that use window handles after destroying the window, the problem is so widespread that the window manager folks in Windows NT decided to take a more proactive approach. (Reiterating in case you couldn't figure it out: This entry goes "behind the scenes". Behavior described here falls into the category of "implementation detail" and is subject to change at any time.) In Windows NT, the 32-bit `HWND` was broken into two parts. The bottom 16 bits acted like the window handle of Windows 95: It was an index into a table. But whereas Windows 95 left the upper 16 bits zero for compatibility with 16-bit programs, Windows NT used the top bits as a "uniquifier". For example, the first time entry 0x0124 in the table was used, the handle corresponding to that entry was 0x00010124. After that window is destroyed and a new one created in slot 0x0124, the handle for the new entry is 0x00020124. Each time the entry is re-used, the uniquifier increments by one. But what about those 16-bit programs? When a 16-bit program used a window handle, the window manager would just use that value as the index into the table. There was no uniquifier, so the window manager just ran with whatever was in that slot. After all, the uniquifier is wasn't essential to correct operation of the window manager; it was added merely to cover for buggy programs. Consequently, 16-bit programs were just as susceptible to the "handle re-use problem" as they were in Windows 3.1 and Windows 95. Only 32-bit programs got the extra protection. But since the window handle was an index, the full 16-bit range of window handles became available (minus special values like `NULL`), for a theoretical maximum of around 65,000 windows. Well, except that the actual theoretical maximum is is half that, around 32,700 windows. Why did we lose half of the window handles? Because there are programs out there that assume that window handles are always even numbers!

Next time, we'll look at the magical 10,000 per-process handle limit.

Raymond Chen

Follow