How are window manager handles determined in 16-bit Windows and Windows 95?

devblogs.microsoft.com/oldnewthing/20070716-00

July 16, 2007

Raymond Chen

(Prefatory remark: The following information is of the "behind the scenes" nature and does not constitute formal documentation. Actually, **nothing** I write counts as formal documentation, so I shouldn't have needed to write that, but people sometimes intentionally play stupid and interpret all forms of the future tense as if I were making some sort of "guarantee" on behalf of Microsoft Corporation. I assure you that I have no such authority! It's times like that that I'm tempted to just give up writing.) Let's start with 16-bit window handles. Those are simple: They are just pointers into the window manager's data segment, cast to the `HWND` data type. Since the window manager had a single 64KB data segment, all of these pointers were 16-bit values. In Windows 95, the window manager moved several categories of objects out of the default data segment into their own custom heaps. (And those were 32-bit heaps so they could be bigger than 64KB.) Window classes, menus, and windows each got their own "big" heap. There may have been other categories of objects that moved out of the default data segment, but those are the ones I remember. But since Windows 95 still had to support 16-bit programs, it needed a way to return 16-bit window handles back to those programs. To do this, the window manager allocated the memory in the 32-bit heap as "movable", which as we learned some time ago isn't actually movable. The purpose of allocating it as movable memory was to get that local memory handle, the `HLOCAL`. No, wait, but that doesn't actually solve the problem, because a local handle in a 32-bit heap is still a 32-bit value. How do we get a 16-bit value out of that? When the window manager created the 32-bit heap, it asked the 32-bit heap manager very nicely if it could give back 16-bit handles instead of 32-bit handles. The heap manager did this by pre-allocating a 64KB block of memory and allocating its handles out of that memory block, using the offset into the block as the handle. Since each entry in the handle table is four bytes (a 32-bit pointer), the 64KB handle table can hold up to 16384 entries. This is why the documentation for `CreateWindowEx` includes the following remark:

> **Windows 95/98/Me**: The system can support a maximum of 16,384 window handles.

Actually, it was a little bit less than that because some of the entries were lost to bookkeeping overhead. For example, the handle value of zero could not be used because that would be confused with `NULL`. Now, you may have asked, "Well, if all the window handles are multiples of four, why not divide by four and then you can get the full range of 65535 window handles?" Well, remember that Windows 3.1 could handle only around 700 windows. Increasing this to 16,384 was enormous progress already. I mean, it's more than 23 times as much as what you had before. A hundred windows was already considered excessive at the time, so the window manager already could accommodate 163 abusive, badly-written programs. There's really no reason to bump that up to 655 badly-written programs. That'd just be encouraging programs to behave badly. Both the 16-bit Windows technique and the Windows 95 technique did suffer from the problem of handle re-use. When a window is destroyed, its memory is freed (as well as its handle on Windows 95). When a new window is created, there's a good chance that the memory or handle will get re-used, and consequently the numerical value of the window handle once again becomes valid, but refers to a different window. It so happens that boatloads of programs (and "boatloads" is a technical term) contain bugs where they use window handles after the window has been destroyed. When a window handle is re-used, that program sends a message to the window it thinks is still there, but instead it sends the message to a completely unrelated window. This doesn't bode well for the program, and it usually doesn't bode well for the new window that received the message by mistake either. Next time, we'll look at how the Windows NT folks addressed this problem of window handle re-use. **Nitpicker's corner** "Boatloads" is not a technical term. That was a joke.

The initial version of this article accidentally omitted the word "not" from the opening sentence. Kudos to the people who were able to exercise their brain and figure this out from context instead of robotically taking everything at face value. There may be hope for the world yet.

Raymond Chen

Follow