On the unanswerability of the maximum number of user interface objects a program can create

devblogs.microsoft.com/oldnewthing/20060901-11

September 1, 2006

Raymond Chen

The answer to the question “What is the maximum number of window classes a program can register?” is not a number. Most user interface objects come from a shared pool of memory known as the “desktop heap”. Although one could come up with a theoretical maximum number of window classes that can fit in the desktop heap, that number is not achievable in practice because the desktop heap is shared with all other user interface objects on the desktop. For example, menus and windows go into the desktop heap, as well as more obscure objects like active window enumerations, window positioning handles used by `DeferWindowPos`, and even how many threads have attached input queues (either implicitly done by having cross-thread parent/owner relationships or explicitly by calling the `AttachThreadInput` function). The more windows and menus you have, the less space available for other things like registered window classes. Typically, when somebody asks this question, the real problem is that they designed a system to the point where desktop heap exhaustion has become an issue, and they need to redesign the program so they aren’t so wasteful of desktop heap resources in general. (One customer, for example, was registering thousands of window classes in their program, which is excessive.) In the same way that somebody asking for the maximum number of threads a process can create is an indication that their program is in need of a redesign, a program that registers thousands of window classes needs to be given another look. After all, even just creating a thousand windows is excessive—any UI that shows the user a thousand windows is too confusing to be usable.

(Pre-emptive link: Q126962: On the desktop heap.)

Raymond Chen

Follow