Why does the CreateProcess function modify its input command line?

devblogs.microsoft.com/oldnewthing/20090601-00

June 1, 2009

Raymond Chen

One of the nasty gotchas of the `CreateProcess` function is that the `lpCommandLine` parameter must be a mutable pointer. If you pass a pointer to memory that cannot be written to (for example, a pointer to a page that is marked `PAGE_READONLY`), then you might crash. Commenter Ritchie wonders why this parameter is so weird.

Basically, somebody back in the 1980's wanted to avoid allocating memory. (Another way of interpreting this is that somebody tried to be a bit too clever.)

The `CreateProcess` temporarily modifies the string you pass as the `lpCommandLine` in its attempt to figure out where the program name ends and the command line arguments begin. Now, it could have made a copy of the string and made its temporary modifications to the copy, but hey, if you modify the input string directly, then you save yourself an expensive memory allocation operation. Back in the old days, people worried about avoiding memory allocations, so this class of micro-optimization is the sort of thing people worried about as a matter of course. Of course, nowadays, it seems rather antiquated.

Now, there may also be good technical reasons (as opposed to merely performance considerations) for avoiding allocating memory on the heap. When a program crashes, the just in time debugger is launched with the `CreateProcess` function, and you don't want to allocate memory on the heap if the reason the program crashed is that *the heap is corrupted*. Otherwise, you can get yourself into a recursive crash loop: While trying to launch the debugger, you crash, which means you try to launch the debugger to debug the new crash, which again crashes, and so on. The original authors of the `CreateProcess` function were careful to avoid allocating memory off the heap, so that in the case the function is being asked to launch the debugger, it won't get waylaid by a corrupted heap.

Whether these concerns are still valid today I am not sure, but it was those concerns that influenced the original design and therefore the interface.

Why is it that only the Unicode version is affected? Well, because the ANSI version of the function just converts its strings to Unicode and the calls the Unicode version of the function. Consequently, the ANSI version of the function happens to implement the workaround as a side effect of its original goal: The string passed to the Unicode version of the function is a temporary string!

Exercise: Why is it okay for the ANSI version of the `CreateProcess` to allocate a temporary string from the heap when the Unicode function cannot?

Raymond Chen

Follow