Simulating input via WM_CHAR messages may fake out the recipient but it won't fake out the input system

devblogs.microsoft.com/oldnewthing/20110728-00

July 28, 2011

Raymond Chen

We saw some time ago that you can't simulate keyboard input with `PostMessage`. You may get away with it, depending on how the application you're trying to fake out processes input, but since you're just faking data, the application may discover that it's all a ruse when they try to access information that you didn't fake out, say by calling `GetKeyState` and discovering that the key it was told was being pressed is in fact not being pressed after all. When you try to do this fake-out, you might or might not be able to fake out the application, but you're definitely not going to fake out the input system itself.

> I wrote a test program that simulates input via `WM_CHAR` messages to write characters into Notepad. I set the local screen saver timeout to one minute, but the screen saver launches after one minute even if my program has been running. On the other hand, when I type on the physical keyboard, the screen saver does not kick in, as expected. How does the screen saver know that the system is really idle and the input is just being generated by a program?

Actually, the question is backwards. How does the screen saver find out about your fake input? Pumping `WM_CHAR` messages directly into Notepad bypasses the input system. It's not that it "knew" that you pulled an end run around it; it is merely acting on what it knows, and your fake input is not part of what it knows. It's like prank-calling somebody and saying, "Hi, this is your credit card company. You don't need to pay your bill this month. Don't worry about it." Sure, that person may be fooled into not paying their bill, but a month later, they're going to get a late payment notice from the credit card company. "How does the credit card company know that the bill wasn't forgiven and the phone call was fake?" If you want to generate input programmatically, use `SendInput`.

Note that the window manager still knows whether the input came from hardware or from `SendInput`, and the `SPI_SETBLOCKSENDINPUTRESETS` system parameter controls whether programmatically-generated input should reset the screen saver.

Raymond Chen

Follow