What if two programs did this?

devblogs.microsoft.com/oldnewthing/20050607-00

June 7, 2005

Raymond Chen

The thought experiment "Imagine if this were possible" is helpful in thinking through whether Windows lets you do something or other. (A special case of this is "When people ask for security holes as features.") If the possibility leads to an obvious contradiction or the violation of generally-accepted rules of metaphysics, then you can be pretty sure that Windows doesn't support it. (Of course, the absence of such a contradiction doesn't prove that Windows **does** support it. But you can use it to rule out obvious bad ideas.)

The question "What if two programs did this?" is also helpful in evaluating a feature or a design request. Combining this with "Imagine if this were possible" leads to an impressive one-two punch. Here are a few examples:

"How do I create a window that is never covered by any other windows, not even other topmost windows?"
Imagine if this were possible and imagine if two programs did this. Program A creates a window that is "super-topmost" and so does Program B. Now the user drags the two windows so that they overlap. What happens? You've created yourself a logical impossibility. One of those two windows must be above the other, contradicting the imaginary "super-topmost" feature.

"How do I mark my process so that it always the first/last to receive the system shutdown notification? I want to do something before/after all other programs have shut down."
Imagine if this were possible and imagine if two programs did this. You now have two programs both of which want to be first/last. But you can't have two first or two last things. One of them must lose. (This of course generalizes to other things people might want to be first or last.)

"How do I make sure that my program is always the one that runs when the user double-clicks an .XYZ file?"
Imagine if this were possible and imagine if two programs did this. Now the user double-clicks an .XYZ file. Which program runs?

In this case, the solution is to leave the user in charge of their file associations; if they decide that they want your competitor's program to be used for .XYZ files, then that's their decision and you should respect it.

My colleague Zeke [link fixed 11am], who is responsible, among other things, for the way file associations work in Explorer, provides a few alternatives:

- Put your program in the "Open With" list and set .XYZ as one of your supported types. Users can right-click an .XYZ file and select "Open With…", then select your program from the list.
- Register a supplemental verb that says "Open with Your Program". Users can then right-click an .XYZ file and select "Open with Your Program".

Here is the entry point in MSDN to the documentation on file associations in Explorer.

For many of these "I want to be the X-est"-type questions, you can often come up with some sort of hack, where you run a timer that periodically checks whether you are still X, and if not, pushes you back into the X-position. And then you stop and think, "What if two programs did this?" and realize that it's a bad idea. At least I hope you do.

Even with this explanation, some people still don't get it. I'll ask them to consider, "What if two programs did this? They'll be fighting back and forth," and the answer I get back is, "Then I can have the second program check if the first program is already running." They don't understand that they didn't write the second program.

When two programs "duke it out" like this, you can't predict which one will win, but you can predict with 100% certainty who will lose: The user.

I remember well when one of my colleagues called me into his office to show me two very popular commercial programs that both wanted to guarantee that they were the program that ran when the user double-clicked an .XYZ document. Since there is no such guarantee, they faked it with the timer hack.

You installed the first program, it set itself as the .XYZ file handler, and everything seemed normal. You then installed the second program, it set itself as the new .XYZ file handler, and the first program noticed and said, "Uh-uh, **I'm** the program that runs .XYZ files", and changed things back. Then the second program said, "No way, **I'm** the program that runs .XYZ files" and set itself back.

This childish game of "Nuh-uh/Yuh-huh!" went on while the user sat there dumbfounded and helpless, watching the icon for their .XYZ files flicker back and forth between the two programs, both of whom egotistically believed they were doing the user a "favor" by insisting on being the program that runs .XYZ files.

Raymond Chen

Follow