Why can programs empty the clipboard when they start up?

devblogs.microsoft.com/oldnewthing/20100510-00

May 10, 2010

Raymond Chen

Via the Suggestion Box, Johan Almén asks, "What was the rationale behind the decision to let Excel empty the clipboard when launched?"

Why can an application empty the clipboard? Because it's there.

After all, the point of the clipboard is to hold information temporarily. Programs are permitted to empty the clipboard, add data to the clipboard, or retrieve data from the clipboard. That's why it's there.

(I'm assuming that the naming of the program Excel was just an example of a program, and that the question wasn't "Why doesn't Windows have a specific check for the program `EXCEL.EXE` and block its clipboard access while still allowing clipboard access to everybody else.")

Okay, maybe the question wasn't so much "Why are programs allowed to empty the clipboard" as it was "Why are programs allowed to empty the clipboard when they launch?" Well, because that might have been the whole point of the program! Somebody might write a program called `emptyclip` whose sole purpose in life is to empty the clipboard. You run the program, it empties the clipboard, and then it exits. Short and sweet. If Windows didn't allow programs to empty the clipboard when they started up, then this program would not be able to get its work done.

You might not consider that particularly useful, but there are actually quite a few programs which empty the clipboard when they start up. For example, the `clip` program that comes with Windows takes its standard input and places it on the clipboard. Implied in that functional description is that it erases what used to be on the clipboard. Everything the program does is in its startup.

```
echo I'm on the clipboard! And I erased what use to be there.| clip
```

Many scripting languages provide access to the clipboard, if not natively, then through an extension. Since these are typically not GUI programs, as far as the window manager can tell, these programs as perpetually stuck in their startup code: They never go input idle because they never pump messages. Prohibiting programs from accessing the clipboard during startup means that console programs are effectively banned from modifying the clipboard at all.

Okay, maybe the question wasn't "Why are programs allowed to empty the clipboard when they launch?" so much as it was "Why are programs allowed to empty the clipboard outside of an explicit user action (like a click or a hotkey)?" Well, we still have the problem of programs whose design is to empty the clipboard without any user interaction, like all those console scripts. But you also remove many GUI usage patterns, such as pushing work to a background thread so that the program can remain responsive. And it would also prevent you from writing a program that modified the clipboard in response to a drop operation. I can imagine a program called `filecontentstoclip` which just sits there and waits for you to drag/drop a file onto its window. When you do that, it opens the file and places the file's contents onto the clipboard. Since the drag/drop operation is handled by the drag source, the drop target receives no input and (according to the rule of "no clipboard access without user input") is denied permission to erase the old clipboard contents.

In order for these sorts of interaction models to work, there would have to be some sort of `AllowClipboardAccess` function (akin to `AllowSetForegroundWindow`) so that one process can temporarily transfer clipboard access permission to another process. It could be done, but it would make writing applications more complicated, because you would have to anticipate what operations might result in another application wanting to access the clipboard and scattering calls to `AllowClipboardAccess` in various places in your program. If you miss a spot, you'll get some bug filed against your program that says, "When I click the *Preview* button, and I've set my custom previewer to program X and configure program X to say 'always copy image to clipboard when previewing', the feature doesn't work."

The clipboard was part of Windows 1.0, and back in those days, you didn't have a lot of memory available. You had to get a lot done with very little. Programmers were trusted to use their great power with great responsibility. And besides, as we saw with programs like `clip` (and hypothetical programs like `emptyclip` and `filecontentstoclip`), allowing programs to empty the clipboard at startup made it possible to write some interesting and useful tools. Windows historically didn't stop programmers from doing stupid things because that would also prevent them from doing clever things.

Raymond Chen

Follow