For the Nitpickers: Enhanced-mode Windows 3.0 didn't exactly run a copy of standard-mode Windows inside the virtual machine

devblogs.microsoft.com/oldnewthing/20130208-00

February 8, 2013

Raymond Chen

Generally speaking, Enhanced-mode Windows 3.0 ran a copy of standard-mode Windows inside the virtual machine. This statement isn't exactly true, but it's true enough.

Commenter Nitpicker objected, "Why are you threatening us with the Nitpicker's Corner for asking about this issue instead of explaining it once and linking it everywhere?"

Okay, first of all, as far as I can tell, you're the first person to ask about the issue. So you can't say "Everybody who asks about the issue is threatened with the Nitpicker's Corner" because up until you made your comment, *nobody ever asked*. Okay, well, technically you can say it, because every statement quantified over the empty set is true. But it is equally true that, at the time you made your comment, that "Everybody who asks about the issue is awarded a new car." So it is not a meaningfully true statement.

I haven't bothered explaining the issue because the issue has never been central to the main point of whatever article happens to bring it up. The statement is true enough for the purpose of discussion, and the various little corners in which the statement breaks down have no bearing on the original topic. Nitpickers would point out that you can't combine velocities by simple addition because of the laws of Special Relativity. Even when the situation under discussion takes place at non-relativistic speeds.

As for the suggestion, "Explain it once and link it everywhere," you're assuming that I can even explain it once, that doing so is less work than just saying "not exactly true, but true enough," and that I would enjoy explaining it in the first place.

If you don't like it, you can ask for your money back.

Okay, I went back and dug through the old Windows 3.0 source code to answer this question. It took me about four hours to study it all, try to understand what the code was doing, and then distill the conclusions into this article. Writing up the results took another two hours. That's six hours I could've spent doing something enjoyable.

The 16-bit Windows kernel was actually three kernels. One if you were using an 8086 processor, another if you were using an 80286 processor, and a third if you were using an 80386 processor. The 8086 kernel was a completely separate beast, but the 80286 and 80386 kernels shared a lot of code in common. The major difference between the 80286 and 80386 kernels was in how they managed memory, because the descriptor tables on the 80386 were a different format from the descriptor tables on the 80286. The 80386 memory manager could also take advantage of the new 32-bit registers.

But the difference between the 80286 and 80386 kernels were not based on whether you were running Standard or Enhanced mode. If you're running on an 80386 processor, then you get the 80386 kernel, regardless of whether you're using Standard or Enhanced mode Windows. And since Enhanced mode Windows required an 80386 processor, the behavioral changes between Standard and Enhanced mode were restricted to the 80386 kernel.

The 80386 kernel was designed to run as a DPMI client. It asked the DPMI host to take it into protected mode, then used the DPMI interface to do things like allocate selectors and allocate memory. If you ran Windows in Standard mode, then the DPMI host was a custom-built DOS extender that was created just for Standard mode Windows. If you ran Windows in Enhanced mode, then the DPMI host was the 32-bit virtual machine manager. Abstracting to the DPMI interface allowed a single 80386 kernel to run in both Standard and Enhanced modes.

And in fact if you ran Enhanced mode Windows with paging disabled, then the code running in the 80386 kernel was pretty much the same code that ran if you had run the 80386 kernel under Standard mode Windows.

One obvious place where the behavior changed was in the code to manage MS-DOS applications, because Enhanced mode Windows could multi-task MS-DOS applications, and Standard mode Windows could not.

Another place where the behavior changed was in in the code to allocate more selectors: The attempt to retry after extending the local descriptor table was skipped if you were running under the Standard mode DOS extender, because the Standard mode DOS extender didn't support extending the local descriptor table.

And another difference is that the Windows idle loop in Enhanced mode would issue a special call to release its time slice to any multi-tasking MS-DOS applications. (If you were running in Standard mode, there were no multi-tasking MS-DOS applications, so there was nobody to release your time slice to.)

Another thing special that the 80386 kernel did was register with the virtual machine manager so that it could display an appropriate message when you pressed `Ctrl`+`Alt`+`Del`. For example, you saw this message if you hit `Ctrl`+`Alt`+`Del` while there was a hung Windows application:

Contoso Deluxe Music Composer
This Windows application has stopped responding to the system.
* Press ESC to cancel and return to Windows.
* Press ENTER to close this application that is not responding.
  You will lose any unsaved information in this application.
* Press CTRL+ALT+DEL again to restart your computer. You will
  lose any unsaved information in all applications.

But all these differences are minor in the grand scheme of things. The window manager behaved the same in Standard mode and Enhanced mode. GDI behaved the same in Standard mode and Enhanced mode. Printer drivers behaved the same in Standard mode and Enhanced mode. Only the low-level kernel bits had to change behavior between Standard mode and Enhanced mode, and as you can see, even those behavior changes were relatively minor.

That's why I said it was "true enough" that what was running inside the virtual machine was a copy of Standard-mode Windows.

Raymond Chen

Follow