Why didn't they use the Space Shuttle to rescue the Apollo 13 astronauts?

devblogs.microsoft.com/oldnewthing/20110119-00

January 19, 2011

Raymond Chen

Many decisions make sense only in the context of history. Much like the moviegoers who were puzzled why NASA didn't just use the Space Shuttle to rescue the Apollo 13 astronauts, computer users of today, when looking back on historical decisions, often make assumptions based on technology that didn't exist. Consider, for example, pointing out that the absence of a console subsystem in Windows 3.1 was no excuse for not porting the `ipconfig` program as a character-mode application. "Sure maybe you didn't have a console subsystem, but why not just use the DOS box?" The MS-DOS prompt is a virtual machine running a copy of MS-DOS. Since it's a virtual machine, as far as the MS-DOS prompt is concerned, it's just running all by its happy self on a dedicated computer running MS-DOS. In reality, of course, it's running inside a simulator being controlled by Windows, but the point of the simulation is so that old applications can continue to run even though they think they're running under MS-DOS. "There wasn't any security in place with Win 3.1, so any program run from a DOS box should have been able to affect anything on the system." Since the MS-DOS prompt ran in a virtual machine, everything it did was under the supervision of the virtual machine manager. If it tried to access memory it didn't have permission to access, an exception would be raised and handled by the virtual machine manager. If it tried to execute a privileged instruction, an exception would be raised, and the virtual machine manager would step in with a "Nope, I'm not going to let you do that" and terminate the virtual machine. In a sense, programs running in the MS-DOS prompt actually ran with *more* protection and isolation than Windows applications running on the desktop, because Windows created *a whole separate universe* for each MS-DOS prompt. One of the consequences of virtualization is that programs running in the MS-DOS prompt are plain old MS-DOS applications, not Windows applications. There is no Windows API in MS-DOS, so there is no Windows API in the MS-DOS prompt either. (It's like running Windows inside a virtual machine on your Linux box and wondering why your Windows program can't call `XCreateWindow`. It can't call `XCreateWindow` because that's a function on the host system, not in the virtual machine.) Okay, but let's suppose, just for the sake of argument, that somebody poked a hole in the virtual machine and provided a way for MS-DOS programs to call WinSock APIs. You still wouldn't want `ipconfig` to be an MS-DOS program. Recall that Windows 3.1 ran in one of two modes, either *standard mode* or *enhanced mode*. Standard mode is the version designed

for the 80286 processor. It didn't have virtual memory or support for virtual machines. When you ran an MS-DOS prompt, standard mode Windows would freeze all your Windows programs and effectively put itself into suspended animation. It then ran your MS-DOS program (full-screen since there was no Windows around to put it in a window), and when your MS-DOS program exited, Windows would rouse itself from its slumber and bring things back to the way they were before you opened that MS-DOS prompt. It would kind of suck if getting your computer's IP address meant stopping all your work, shutting down Windows (effectively), and switching the video adapter into character mode, just so it could print 16 characters to the screen. "Well, who cares about standard mode Windows any more? Let's say that it only works in enhanced mode. Enhanced mode can multi-task MS-DOS prompts and run them in a window." Recall that the minimum memory requirements for Windows 3.1 in enhanced mode was 1664KB of memory. Given that each MS-DOS box took up about 1MB of memory, you're saying that displaying 16 characters of information is going to consume over half of your computer's memory? "Okay, helpdesk wants to know my IP address so they can troubleshoot my computer. In order to do that, I have to run this program, but first I need to save all my work and exit all my programs in order to free up enough memory to run the program they want me to run." Better to just write a simple Windows application. **Bonus commentary**: 640k asks, "Why wasn't winipcfg called ipconfig?"

Right. "Let's have two completely different and incompatible programs with the same name." See how far you get with that.

Raymond Chen

Follow