Why did Windows 95 use blue screen error messages instead of hard error messages?

devblogs.microsoft.com/oldnewthing/20240903-00

September 3, 2024

Some time ago, I talked about the so-called hard error and what makes it harder than an easy error. The idea is that the “hard” error is used for low-level I/O errors that occur at unpredictable times, and which must be handled without allowing application code to run (because of the unpredictability).

If the hard error message could do all of those things, why didn’t Windows 95 use them instead of the ugly blue screen messages?

TL;DR: In Windows 95, the hard error message is at the wrong layer.

In Windows in real mode or standard mode, you had Windows running on top of MS-DOS as the file I/O layer.

Windows GUI
Windows I/O layer
MS-DOS

When an I/O error occurred, it was reported to the Windows I/O layer, which then asked the Windows GUI to freeze all application so it could display the hard error dialog.

Recall that Windows in enhanced mode and Windows 95 were really three operating systems running together. There was a virtual machine manager which did the things traditionally associated with an operating system kernel: Memory management, paging, I/O, time slicing, resource virtualization, all that stuff. Inside the virtual machine manager, you had one virtual machine for running Windows programs, and you had additional virtual machines, one for each MS-DOS session.

Windows GUI	MS-DOS session	MS-DOS session
Windows I/O layer		
Virtual machine manager		

The types of errors that resulted in blue screen messages were detected by the virtual machine manager, and it happened asynchronously with respect to whatever is happening in the Windows or MS-DOS virtual machines. The virtual machine manager can't report all of its problems to the Windows I/O layer: For one thing, the problem may not even have originated from Windows. It could be a problem with an operation performed by an MS-DOS program.

For another thing, the problem may have been detected by the virtual machine manager at a time when various internal locks are held that may prevent the Windows virtual machine from doing anything. For example, the problem may have been discovered while the display driver owns an internal mutex, which means that a display context switch from MS-DOS to the Windows virtual machine is not possible. The virtual machine manager needs to report the problem and get an answer from the user, but it's not allowed to change ownership of the video card from the MS-DOS virtual machine to the Windows virtual machine.

Wait, if the display driver is locked, how can it display a blue screen message?

The blue screen message is special: One of the design requirements for virtual display drivers is that they must be able to switch from any mode into text mode at the drop of a hat (known as "message mode"), display a text message, and then switch back to the original mode, preserving all state. This requires setting aside a small amount of memory (enough to hold a 25 × 80 text mode screen) to save the video memory that is about to be used by the text mode message, plus a small amount of memory to keep track of the video card state that needs to be restored when exiting blue screen message mode.

This is a much more tractable problem than requiring the display driver to be able to do a full context switch from one graphics mode to another. For one thing, the amount of memory required to store an entire 1024 × 768 graphics screen at 32 bits per pixel is three megabytes. Reserving this much non-paged memory for emergency use is somewhat impractical seeing as it would consume all of the system memory in Windows 3.1 enhanced mode (minimum hardware requirements 3MB RAM), and nearly all of it in Windows 95 (minimum hardware requirements 4MB RAM).

Even worse, the problem may be detected at a time when a deadlock with the Windows virtual machine is possible. For example, the problem might be a resource conflict, say, because the Windows virtual machine wants to access the same communications port that an MS-DOS program is already using. The virtual machine manager needs to ask the user to

resolve this conflict by saying who should get access to the port. If the virtual machine manager asked the Windows virtual machine to display a hard error message, it would create a deadlock, because the Windows virtual machine is waiting for access to the communications port.

At the end of the day, the problem is one of layering. If the problem is detected by the virtual machine manager, that problem is effectively asynchronous with respect to what's happening inside the virtual machines that it is hosting. It is being generated at too low a layer in the architectural stack.

Bonus reading: How did Windows 3.1's virtual machine manager get the information to show in the text-mode Alt+Tab switcher? In the case where the request for input from the user is not blocking multitasking, the virtual machine manager could use an asynchronous pattern to communicate with some code running in the Windows virtual machine.