What is a hard error, and what makes it harder than an easy error?

devblogs.microsoft.com/oldnewthing/20240116-00

January 16, 2024

Raymond Chen

Occasionally, you may see a reference in Windows to “hard errors”, such as in this article that shows how to configure an unattended system so that it auto-answers hard error dialogs. But why is it called a “hard error”? Are there “easy errors”?

In 16-bit Windows, if there was an I/O error reading a drive, you got an error message that looked like this:

System Error

Cannot read from drive B.

Abort
Retry
Cancel

These messages were special because they were generated from inside the I/O system, below the user interface level. Windows can’t put up a traditional `MessageBox` because that pumps messages and allows application code to run at a time when the rules of co-operative multitasking say that application code should not be running.

The code to display these special “hard system modal errors” was carefully written so as to rely only on parts of the user interface code that were re-entrant. In fact, the only user interface code it uses is processing mouse and keyboard input. All of the graphics are drawn by asking GDI to draw directly to the frame buffer, and all of the dialog behaviors are handwritten. No application code was allowed to run while this message was being shown to the user.

The opposite of the “hard error” message was the “soft error” message, which was your regular `MessageBox`.

These types of hard error messages disappeared from 32-bit Windows, but the name got repurposed to describe other types of critical error messages, usually low-level ones.