On 64-bit Windows, 32-bit programs run in an emulation layer, and if you don't like that, then don't use the emulator

devblogs.microsoft.com/oldnewthing/20081222-00

December 22, 2008

Raymond Chen

On 64-bit Windows, 32-bit programs run in an emulation layer. This emulation layer simulates the x86 architecture, virtualizing the CPU, the file system, the registry, the environment variables, the system information functions, all that stuff. If a 32-bit program tries to look at the system, it will see a 32-bit system. For example, if the program calls the `GetSystemInfo` function to see what processor is running, it will be told that it's running on a 32-bit processor, with a 32-bit address space, in a world with a 32-bit sky and 32-bit birds in the 32-bit trees.

And that's the point of the emulation: To keep the 32-bit program happy by simulating a 32-bit execution environment.

Commenter Koro

is writing an installer in the form of a 32-bit program that detects that it's running on a 64-bit system and wants to copy files (and presumably set registry entries and do other installery things) into the 64-bit directories, but the emulation layer redirects the operations into the 32-bit locations. The question is "What is the way of finding the x64 Program Files directory from a 32-bit application?"

The answer is "It is better to work with the system than against it." If you're a 32-bit program, then you're going to be fighting against the emulator each time you try to interact with the outside world. Instead, just recompile your installer as a 64-bit program. Have the 32-bit installer detect that it's running on a 64-bit system and launch the 64-bit installer instead. The 64-bit installer will not run in the 32-bit emulation layer, so when it tries to copy a file or update a registry key, it will see the real 64-bit file system and the real 64-bit registry.

Raymond Chen

Follow