How to detect programmatically whether you are running on 64-bit Windows

devblogs.microsoft.com/oldnewthing/20050201-00

February 1, 2005

Raymond Chen

To detect programmatically whether your 32-bit program is running on 64-bit Windows, you can use the IsWow64Process function.

Do not do as some people do and hard-code the list of 64-bit processors. You'd think that after the hard-coded list of 64-bit processors changed the first time (when x64 was added to ia64), people would have learned their lesson.

But how do you detect programmatically from your 64-bit process whether you are running on 64-bit Windows? Easy.

```
BOOL Is64BitProcessRunningOn64BitWindows()
{
 return TRUE;
}
```

The fact that your 64-bit program is running at all means that you are running on 64-bit Windows! If it were a 32-bit machine, your program wouldn't be able to run.

It's like asking the question, "Is the power on?" If there were no power, your program wouldn't be able to ask the question.

Of course, if you want a single source code base that can be compiled both as a 32-bit program and as a 64-bit program, you have a tiny amount of work to do.

```
BOOL Is64BitWindows()
{
#if defined(_WIN64)
 return TRUE;  // 64-bit programs run only on Win64
#elif defined(_WIN32)
 // 32-bit programs run on both 32-bit and 64-bit Windows
 // so must sniff
 BOOL f64 = FALSE;
 return IsWow64Process(GetCurrentProcess(), &f64) && f64;
#else
 return FALSE; // Win64 does not support Win16
#endif
}
```

I threw in a branch for 16-bit programs if you're crazy enough to be still writing 16-bit Windows programs.

Raymond Chen

Follow