How can I detect programmatically whether the /3GB switch is enabled?

devblogs.microsoft.com/oldnewthing/20141023-00

October 23, 2014

Raymond Chen

A customer was doing some diagnostic work and wanted a way to detect whether the `/3GB` switch was enabled. (Remember that the `/3GB` switch is meaningful only for 32-bit versions of Windows.)

The way to detect the setting is to call `GetSystemInfo` and look at the `lpMaximum-ApplicationAddress`.

```
#include <windows.h>
#include <stdio.h>
int __cdecl main(int, char **)
{
 SYSTEM_INFO si;
 GetSystemInfo(&si);
 printf("%p", si.lpMaximumApplicationAddress);
 return 0;
}
```

Compile this *as a 32-bit program* and run it.

Configuration	LARGEADDRESS-AWARE?	Result	Meaning
32-bit Windows, standard configuration	Any	7FFEFFFF	2GB minus 64KB
32-bit Windows, /3GB	Any	BFFFFFFF	3GB
32-bit Windows, increaseuserva = 2995	Any	BB3EFFFF	2995 MB
64-bit Windows	No	7FFEFFFF	2GB minus 64KB

64-bit Windows	Yes	FFFEFFFF	4GB minus 64KB

On 32-bit systems, this reports the system-wide setting that specifies the maximum user-mode address space, regardless of how your application is marked. Note, however, that your application must be marked `LARGEADDRESSAWARE` in order to take advantage of the space above 2GB.

On the other hand, when you run a 32-bit application on 64-bit Windows, it runs the application in an emulation layer. Therefore, 64-bit Windows can give each application a different view of the system. In particular, depending on how your application is marked, Windows can emulate a 32-bit system with or without the `/3GB` switch enabled, based on what the application prefers.

Armed with this knowledge, perhaps you can help this customer. Remember, you sometimes need to go beyond simply answering the question and actually solve the customer's problem.

> We would like to know how to detect from our 32-bit application whether the host operating system is 64-bit or 32-bit.
>
> We need to know this because our program does some data processing, and we have to choose an appropriate algorithm. We have written one algorithm that is faster but uses 1½GB of address space, and we have also written a fallback algorithm that is slower but does not use anywhere near as much address space. When running on a native 32-bit system, there is typically not 1½GB of address space available, so we have to use the slow algorithm. But when running on a native 64-bit system (or a native 32-bit system with the `/3GB` switch enabled), our program can use the fast algorithm. Therefore, we would like to detect whether the native operating system is 64-bit so that we can decide whether to use the fast or slow algorithm.

Here's another customer question you can now answer:

> We have a 64-bit program, and since we know that Windows currently does not use the full 64-bit address space, we would like to steal the upper bits of the pointer to hold additional information: If there are at least 8 bits available, we can use a more efficient data format. Otherwise, we fall back to a less efficient format. How can we detect whether the upper 8 bits are being used for addressing?

Update: Clarified the table based on misunderstanding in comments.

Raymond Chen

Follow