What happens if you call VirtualAlloc to MEM_COMMIT a page you never MEM_RESERVE?

devblogs.microsoft.com/oldnewthing/20151008-00

October 8, 2015

Raymond Chen

A customer reported that while trying to solve a problem with their program, they noticed that they had been calling `VirtualAlloc` incorrectly for years. They were able to reduce it into a simple program:

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>

int _tmain(int argc, _TCHAR* argv[])
{
 LPVOID base = VirtualAlloc(NULL, 4096, MEM_COMMIT, PAGE_READWRITE);
 _tprintf(TEXT("Allocated at %p\n"), base);
 return 0;
}
```

First of all, thank you for reducing your program. That really focuses the investigation.

The customer noted that their code was passing the `MEM_COMMIT` flag without the `MEM_RESERVE` flag, a scenario that is specifically called out in the documentation:

> The function fails if you attempt to commit a page that has not been reserved. The resulting error code is **ERROR_INVALID_ADDRESS**.

But their call to `VirtualAlloc` was succeeding! The customer suspected that this was not actually the source of their problem, but they wanted to double-check that perhaps their incorrect use of `VirtualAlloc` was somehow indirectly contributing to it. Specifically, they were wondering if what they're doing is okay, or whether they should always use `MEM_RESERVE | MEM_COMMIT`.

What the customer found is a compatibility hack. A lot of application forget to set the `MEM_RESERVE` flag when they `MEM_COMMIT`, so the memory manager lets it slide if they also pass `lpAddress = NULL`, indicating that they are requesting a new allocation rather than modifying an existing one.

The problem is that MSDN fell into the trap of over-documenting. Instead of documenting the contract, MSDN documented the implementation. The contract is “A page being committed must also be reserved.” If you try to commit a page that is not also reserved, then the behavior is unspecified. It is therefore valid for the implementation to treat the violation as “Sorry, you lose,” or “Okay, I’ll let you do it, but just this time.”

It appears that some time after this issue was identified, the MSDN documentation was revised. But they didn’t revise it by documenting the contract. They revised it by documenting the implementation *more precisely*.

> Attempting to commit a specific address range by specifying **MEM_COMMIT** without **MEM_RESERVE** and a non-**NULL** *lpAddress* fails unless the entire range has already been reserved. The resulting error code is **ERROR_INVALID_ADDRESS**.

My recommendation to the customer was to switch to `MEM_RESERVE | MEM_COMMIT`, since that is the preferred behavior and therefore the one least likely to trigger compatibility behavior. But the fact that they were accidentally omitting the `MEM_RESERVE` was not related to their problem, and they should keep looking.

Raymond Chen

Follow