Myth: The /3GB switch lets me map one giant 3GB block of memory

devblogs.microsoft.com/oldnewthing/20040816-00

August 16, 2004

Raymond Chen

Just because the virtual address space is 3GB doesn't mean that you can map one giant 3GB block of memory. The standard holes in the virtual address space are still there: 64K at the bottom, and 64K near the 2GB boundary. Moreover, the system DLLs continue to load at their preferred virtual addresses which lie just below the 2GB boundary. The process heap and other typical process bookkeeping also take their bites out of your virtual address space. As a result, even though the user-mode virtual address space is nearly 3GB, it is not the case that all of the available space is contiguous. The holes near the 2GB boundary prevent you from getting even 2GB of contiguous address space. Some people may try to relocate the system DLLs to alternate addresses in order to create more room, but that won't work for multiple reasons. First, of course, is that it doesn't get rid of the 64K gap near the 2GB boundary. Second, the system allocates other items such as thread information blocks and the process environment variables before your program gets a chance to start running, so by the time your program gets around to allocating memory, the space it wanted may already have been claimed.

Third, the system really needs certain key DLLs to be loaded at the same address in all processes. For example, the syscall trap must reside at a fixed location so that the kernel-mode trap handler will recognize it as a valid syscall trap and not as an illegal instruction. The debugger requires this as well, so that it can use `CreateRemoteThread` to inject a breakpoint into the process when you tell it to break into the process you are debugging.

Raymond Chen

Follow