Myth: Without /3GB a single program can't allocate more than 2GB of virtual memory

devblogs.microsoft.com/oldnewthing/20040810-00

August 10, 2004

Raymond Chen

Virtual memory is not virtual address space (part 2).

This myth is being perpetuated even as I write this series of articles.

The user-mode virtual address space is normally 2GB, but that doesn't limit you to 2GB of virtual memory. You can allocate memory without it being mapped into your virtual address space. (Those who grew up with Expanded Memory or other forms of bank-switched memory are well-familiar with this technique.)

```
HANDLE h = CreateFileMapping(INVALID_HANDLE_VALUE, 0,
                             PAGE_READWRITE, 1, 0, NULL);
```

Provided you have enough physical memory and/or swap file space, that 4GB memory allocation will succeed.

Of course, you can't map it all into memory at once on a 32-bit machine, but you can do it in pieces. Let's read a byte from this memory.

```
BYTE ReadByte(HANDLE h, DWORD offset)
{
 SYSTEM_INFO si;
 GetSystemInfo(&si);
 DWORD chunkOffset = offset % si.dwAllocationGranularity;
 DWORD chunkStart = offset – chunkOffset;
 LPBYTE pb = (LPBYTE*)MapViewOfFile(h, FILE_MAP_READ, 0,
      chunkStart, chunkOffset + sizeof(BYTE));
 BYTE b = pb[chunkOffset];
 UnmapViewOfFile(pb);
 return b;
}
```

Of course, in a real program, you would have error checking and probably a caching layer in order to avoid spending all your time mapping and unmapping instead of actually doing work.

The point is that virtual address space is not virtual memory. As we have seen earlier, you can map the same memory to multiple addresses, so the one-to-one mapping between virtual memory and virtual address space has already been violated. Here we showed that just because you allocated memory doesn't mean that it has to occupy any space in your virtual address space at all.

[Updated: 10:37am, fix minor typos reported in comments.]

Raymond Chen

Follow