Oh, that's probably why I'm in the Quake credits

devblogs.microsoft.com/oldnewthing/20151111-00

November 11, 2015

Raymond Chen

Back in 2012, I couldn't remember why I was in the Quake credits. But then the comment from Kristaps pinpointed the most likely reason: The `Sys_PageIn` function.

```
void Sys_PageIn (void *ptr, int size)
{
    byte    *x;
    int     j, m, n;

// touch all the memory to make sure it's there. The 16-page skip is to
// keep Win 95 from thinking we're trying to page ourselves in (we are
// doing that, of course, but there's no reason we shouldn't)
    x = (byte *)ptr;

    for (n=0 ; n<4 ; n++)
    {
        for (m=0 ; m<(size - 16 * 0x1000) ; m += 4)
        {
            sys_checksum += *(int *)&x[m];
            sys_checksum += *(int *)&x[m + 16 * 0x1000];
        }
    }
}
```

What this code does is access the memory block specified by the `ptr` and `size` parameters in an unusual pattern: It reads byte zero, then the byte at an offset of 16 pages, then byte one, then a byte at an offset of 16 pages plus one, and so on, alternating between a byte and its counterpart 16 pages ahead.

This specific access pattern in Windows 95 defeated the "sequential memory scan" detection algorithm.

Recall that computers in the Windows 95 era had 4MB of RAM. Suppose you were working in a document for a long time. Finally, you're done, and you close the window or minimize it. Boom, now your desktop is visible and the wallpaper bitmap needs to be paged in. If your screen is 1024 × 768 at 16 bits per pixel, that comes out to 1.5MB of memory. Paging in 1.5MB of memory means for the bitmap means kicking out 1.5MB of memory being used for

other stuff, and that's a lot of memory for a machine that has only 4MB to work with (especially since a lot of that 4MB belongs to stuff that isn't eligible for being paged out). The phenomenon we saw was that repainting your desktop would flush out most of your memory.

And then the next thing you do is probably launch a new application, which will cover the wallpaper, so the wallpaper memory isn't going to be needed any more. So we basically purged all the memory in your system in order to handle a huge block of memory that got accessed only once.

The trick that Windows 95 used was to watch your pattern of page faults, and if it saw that you were doing sequential memory access, it started marking the memory 16 pages behind the current access as *not recently accessed*. In the case of a straight sequential scan, this means that the entire buffer cycles through a 64KB window of memory, regardless of the buffer size. With this trick, a 4MB buffer ends up consuming only 64KB of memory, as opposed to using all the memory in your system.

The `Sys_PageIn` function specifically defeates the sequential-scan detector by intentionally going back 16 pages and accessing the page again. This causes it to be marked *recently used*, counteracting the *not recently used* that the sequential-scan detector had done. Result: The memory pages are all marked *recently used* and are no longer prime candidates for being paged out.

Raymond Chen

Follow