A history of GlobalLock, part 2: Selectors

devblogs.microsoft.com/oldnewthing/20041105-00

November 5, 2004

Raymond Chen

With the advent of the 80286, Windows could take advantage of that processor's "protected mode". processor. There was still no virtual memory, but you did have memory protection. Global handles turned into "descriptors", more commonly known as "selectors".

Architectural note: The 80286 did have support for both a "local descriptor table" and a "global descriptor table", thereby making it possible to have each process run in something vaguely approximating a separate address space, but doing so would have broken Windows 1.0 compatibility, where all memory was global.

Addresses on the 80286 in protected mode consisted of a selector and an offset rather than a segment and an offset. This may seem like a trivial change, but it actually is important because a selector acts like a handle table *in hardware*.

When you created a selector, you specified a whole bunch of attributes, such as whether it was a code selector or a data selector, whether it was present or discarded, and where in memory it resided. (Still no virtual memory, so all memory is physical.)

GlobalAlloc() now returned a selector. If you wanted to, you could just use it directly as the selector part of an address. When you loaded a selector, the CPU checked whether the selector was present, discarded, or invalid.

- If present, then everything was fine.
- If discarded, a "not present" exception was raised. (Wow, we have exceptions now!) The memory manager trapped this exception and did whatever was necessary to make the selector present. This meant allocating the memory (possibly compacting and discarding to make room for it), and if it was a code selector, loading the code back off the disk and fixing it up.
- If invalid, an Unrecoverable Application Error was raised. This is the infamous "UAE".

Since memory accesses were now automatically routed through the descriptor table by the hardware, it meant that memory could be moved around with relative impunity. All existing pointers would remain valid since the selector remains the same; all that changes is the

internal bookkeeping in the descriptor table that specified which section of memory the descriptor referred to.

For compatibility with Windows 1.0, GlobalAlloc() continued to emulate all the moveability rules as before. It's just that the numeric value of the selector never really changed any more. (And please let's just agree to disagree on whether backwards compatibility is a good thing or not.)

Next time, transitioning to Win32.

Raymond Chen

Follow