The kooky STRRET structure

devblogs.microsoft.com/oldnewthing/20040823-00

August 23, 2004

Raymond Chen

If you've messed with the shell namespace, you've no doubt run across the kooky STRRET structure, which is used by IShellFolder::GetDisplayNameOf to return names of shell items. As you can see from its documentation, a STRRET is sometimes an ANSI string buffer, sometimes a pointer to a UNICODE string, sometimes (and this is the kookiest bit) an offset into a pidl. What is going on here?

The STRRET structure burst onto the scene during the Windows 95 era. Computers during this time were still comparatively slow and memory-constrained. (Windows 95's minimum hardware requirements were for 4MB of memory and a 386DX processor – which ran at a whopping 25MHz.) It was much faster to allocate memory off the stack (a simple "sub" instruction) than to allocate it from the heap (which might take **thousands** of instructions!), so the STRRET structure was designed so the common (for Windows 95) scenarios could be satisfied without needing a heap allocation.

The STRRET_OFFSET flag took this to an even greater extreme. Often, you kept the name inside the pidl, and copying it into the STRRET structure would take, gosh, 200 clocks (!). To avoid this wasteful memory copying, STRRET_OFFSET allowed you to return just an offset into the pidl, which the caller could then copy out of directly.

Woo-hoo, you saved a string copy.

Of course, as time passed and computers got faster and memory became more readily available, these micro-optimizations have turned into annoyances. Saving 200 clock cycles on a string copy operation is hardly worth it any more. On a 1GHz processor, a single soft page fault costs you over a million cycles; a hard page fault costs you tens of millions.

You can copy a lot of strings in twenty million cycles.

What's more, the scenarios that were common in Windows 95 aren't quite so common any more, so the original scenario that the optimization was tailored for hardly occurs any more. It's an optimization that has outlived its usefulness.

Fortunately, you don't have to think about the STRRET structure any more. There are several helper functions that take the STRRET structure and turn it into something much easier to manipulate.

The kookiness of the STRRET structure has now been encapsulated away. Thank goodness.

Raymond Chen

Follow