Caches are nice, but they confuse memory leak detection tools

devblogs.microsoft.com/oldnewthing/20091127-00

November 27, 2009

Raymond Chen

Knowledge Base article 139071 has the technically correct but easily misinterpreted title FIX: OLE Automation BSTR caching will cause memory leak sources in Windows 2000. The title is misleading because it makes you think that *Oh, this is a fix for a memory leak in OLE Automation*, but that's not what it is. The `BSTR` is the string type used by OLE Automation, and since strings are used a lot, OLE Automation maintains a cache of recently-freed strings which it can re-use when somebody allocates a new one. Caches are nice (though you need to make sure you have a good replacement policy), but they confuse memory leak detection tools, because the memory leak detection tool will not be able to match up the allocator with the deallocator. What the memory leak detection tool sees is not the creation and freeing of strings but rather the allocation and deallocation of memory. And if there is a string cache (say, of just one entry, for simplicity), what the memory leak detection tool sees is only a part of the real story.

- Program (line 1): Creates string 1.
- String manager: Allocates memory block A for string 1.
- Program (line 2): Frees string 1.
- String manager: Puts memory block A into cache.
- Program (line 3): Creates string 2.
- String manager: Re-uses memory block A for string 2.
- Program (line 4): Creates string 3.
- String manager: Allocates memory block B for string 3.
- Program (line 5): Frees string 3.
- String manager: Puts memory block B into cache.
- Program (line 6): Frees string 2.
- String manager: Deallocates memory block A since there is no room in the cache.

Your program sees only the lines marked *Program:*, and the memory leak detection tool sees only the underlined part. As a result, the memory leak detection tool sees a warped view of the program's string usage:

- Line 1 of your program allocates memory block A.

- Line 4 of your program allocates memory block B.
- Line 6 of your program deallocates memory block A.

Notice that the memory leak detection tool thinks that line 6 freed the memory allocated by line 1, even though the two lines of the program are unrelated. Line 6 is freeing string 2, and line 1 is creating string 1! Notice also that the memory leak detection tool will report a memory leak, because it sees that you allocated two memory blocks but deallocated only one of them. The memory leak detection tool will say, "Memory allocated at line 4 is never freed." And you stare at line 4 of your program and insist that the memory leak detection tool is on crack because there, you freed it right at the very next line! You chalk this up as "Stupid memory leak detection tool, it has all these useless false positives." Even worse: Suppose somebody deletes line 6 of your program, thereby introducing a genuine memory leak. Now the memory leak detection tool will report two leaks:

- Memory allocated at line 1 is never freed.
- Memory allocated at line 4 is never freed.

You already marked the second report as bogus during your last round of investigation. Now you look at the first report, and decide that it too is bogus; I mean look, we free the string right there at line 2! Result: A memory leak is introduced, the memory leak detection tool finds it, but you discard it as another bug in the memory leak detection tool. When you're doing memory leak detection, it helps to disable your caches. That way, the high-level object creation and destruction performed in your program maps more directly to the low-level memory allocation and deallocation functions tracked by the memory leak detection tool. In our example, if there were no cache, then every *Create string* would map directly to an *Allocate memory* call, and every *Free string* would map directly to a *Deallocate memory* call. What KB article 139071 is trying to say is *FIX: OLE Automation BSTR cache cannot be disabled in Windows 2000*. Windows XP already contains support for the `OANOCACHE` environment variable, which disables the `BSTR` cache so you can investigate those `BSTR` leaks more effectively. The hotfix adds support for `OANOCACHE` to Windows 2000. **Bonus chatter**: Why do we have `BSTR` anyway? Why not just use null-terminated strings everywhere?

The `BSTR` data type was introduced by Visual Basic. They couldn't use null-terminated strings because Basic permits nulls to be embedded in strings. Whereas Win32 is based on the K&R C&nbsp way of doing things, OLE automation is based on the Basic way of doing things.

Raymond Chen

Follow