When looking to free up disk space, don't forget your symbol file caches

devblogs.microsoft.com/oldnewthing/20230918-35

September 18, 2023

Raymond Chen

If you're trying to free up disk space on a developer system, one place to look for unwanted files is your debugger symbol file caches.

The Windows debugging engine creates a cache of symbol files so it can avoid downloading them repeatedly from the configured symbol server. And that cache can get kind of big, since there's no automatic pruning.[1] This is the disk space version of A cache with a bad policy is another name for a memory leak.

Places to look are

- `$(DebuggerInstallDirectory)\sym`
- `%TEMP%\sym`
- Any directories that have ever been listed in `%_NT_SYMBOL_PATH%`[2]
- `C:\SymCache`
- `C:\symbols`

Visual Studio also caches symbols. You can configure where it saves them under *Tools*, *Options*, *Debugging*, *Symbols*, *Cache symbols in this directory*. It defaults to `%TEMP%\SymbolCache`.

My tool of choice for investigating disk space is WizTree. I have no stake in this program. Just a happy (paying) customer.

What I do is create a symbolic link from each of these potential cache directories into a single directory `C:\SymCache`, so that the different debuggers can share symbol caches.

Bonus chatter: Many years ago, I was working on a laptop and fired up the Disk Cleanup tool, and heard the laptop fan shriek to life as it was "calculating how much space you will be able to free". This seemed odd, so I connected a debugger to figure out why merely deciding what to do was a CPU-intensive operation.[3]

It turns out that the Disk Cleanup tool was in a tight loop checking whether the Cancel button had been clicked, while a background thread was busy doing the calculations. This behavior was part of the Disk Cleaner's original implementation in Windows 98, which was particularly ironic since Windows 98 did not support multiprocessing, so the code was saturating your only CPU just to see if it should keep waiting, starving out the background thread that's actually doing work.

For Windows 10 Version 1709, I fixed it so the code didn't poll the Cancel button. It just went into a normal message pump, and checked the state of the Cancel button after each message. Clicking the Cancel button would itself generate a message, so checking after each message was sufficient.

This dropped the CPU usage of the UI thread down to basically zero.

Previously, in CPU spin loops.

Bonus bonus chatter: A few weeks after I fixed the problem, the Windows performance team filed a bug saying that their telemetry identified the Disk Cleanup tool's Cancel dialog as consuming an unusually high amount of CPU. I had the satisfaction of telling them, "Already fixed."

[1] You can perform manual pruning with the AgeStore program.

[2] You can tell how old that page is, because it says "Because symbols can sometimes be tens of megabytes in size…" Nevermind that the page itself is over a megabyte. (21KB for the main page itself, and the rest is for the fluff around the page, like 263KB for the syntax highlighter JavaScript library and 121KB for the *Microsoft Build* advertisement that was present in early 2021.)

[3] Some time ago, we learned why cleaning Windows Update files is a CPU-intensive operation.