The sad history of Visual Studio's custom __if_exists keyword

devblogs.microsoft.com/oldnewthing/20190828-00

August 28, 2019

Raymond Chen

The Visual Studio C++ compiler has this weird nonstandard keyword called `__if_exists`. (And its twin `__if_not_exists`.) Why does this keyword exist, and how should I use it?

Let's get this out of the way: **Do not use __if_exists under any circumstances whatsoever.** This is pretty much a direct quote from Stephan T. Lavavej, maintainer of the Visual C++ implementation of the C++ standard library and overall "person whose opinion on these sorts of matters you should take under serious consideration." We'll come back to this topic later in the article.

Okay, so where were we? Right, "What's the deal with `__if_exists`?"

Set the time machine to 1996.

The world was a much more innocent place. People left their doors unlocked at night and dialed into the Internet over their 28.8 Kb modems to check their AOL account to see if they got mail, hang out in a chat room, check out Yahoo's top ten sites of the day, and maybe download some software.

Well, that last bit is still true today: Some people still download software over the Internet.[1]

This was also an era in which developers counted bytes, so minimizing the size of the download was a big deal. There was this programming language called C++ that hadn't yet been standardized, but everybody wanted to use it anyway. Compilers of this era didn't know what vectorization was, they didn't think to defer code generation to link time, and they didn't realize that *undefined behavior* gave them permission to violate the laws of physics.

This was the world that the Active Template Library (ATL) was born into.

The ATL team was obsessed with minimizing code size. They came up with all sorts of ideas for how the compiler could shrink binary sizes. Some of those features turned out to be really good ideas that stood the test of time, like `__declspec(novtable)`.

Another of those features was `__if_exists`. It was, in retrospect, not one of those really good ideas.

Remember, this was pre-standard C++. Templates existed in rudimentary form. There was no explicit or partial specialization, no traits types, no template meta-programming, and no SFINAE. Templates in this era were not much more than macros on steroids.

The compiler folks were cajoled into creating a prototype for the `__if_exists` feature, and the ATL team was very pleased with how it turned out. The compiler team wasn't too excited about shipping this prototype-quality feature, but the ATL team insisted that it was essential to making binaries smaller, so the compiler team reluctantly went along with it.

And that's how `__if_exists` ended up in the Visual C++ compiler.

Over time, many people got access to high-speed Internet, the C++ language was standardized, template specialization, template partial specialization, and SFINAE were invented, and as a result, template meta-programming became possible. If all of these language features were put into a time machine and sent back to 1996, there would be no need for `__if_exists`.

The functionality of the `__if_exists` feature was permanently frozen to what shipped in that first prototype, which was itself just good enough to satisfy the immediate needs of the ATL team. The MSDN documentation was updated to document the specific scenarios that are known to work well, leaving everything else as undefined.

And if you go look in the ATL code base, it's not even used very much. The only places you'll find it are in the code that was written in that initial release back in 1996. If those classes needed to be rewritten today for some reason, the replacement would certainly not use `__if_exists`.

If you have code that uses `__if_exists`, you should consider moving to standards-compliant alternatives. For example, if you're using `__if_exists` to compile a block of code only if a member function exists (which is what ATL uses it for), you can use SFINAE to do the detection and `if constexpr` to conditionalize the compilation.

Thanks to my colleague Jonathan Caves for sharing the history behind `__if_exists` (and probably undoing several years of therapy in the process).

Bonus chatter: Jonathan tells me that `__if_exists` is a gift that keeps on giving. Whenever the Visual C++ team goes in and rewrites their C++ parser to handle new language features, or simply to make it run better, they have to retrofit `__if_exists` support into it. And that's when they discover some of the truly insane things people have done with it, like

putting code inside an `__if_exists` block that do not even form a complete C++ construct. The gift has even spread to other compilers: clang 3.0 added support for `__if_exists`. So now multiple compiler vendors have therapy bills to pay.

[1] Okay, and some people still access the Internet over dial-up.

Raymond Chen

Follow