On the strange status of wchar_t in classic MIDL

devblogs.microsoft.com/oldnewthing/20240828-00

August 28, 2024

Raymond Chen

In Windows IDL files, there is a type called `wchar_t`. What does it represent?

If you're asking about the ABI representation of a `wchar_t`, then that's easy: It represents a 16-bit integer value that holds a UTF-16 code unit.

But if you're asking about what data type is used to hold that 16-bit integer value, well, things get a little more complicated.

With one exception, when you use the `wchar_t` type in a Windows IDL file, it is just passed through to the header file, and it is up to the code that includes the header file to decide what `wchar_t` means.

- If the including code is written in C, then for the Microsoft Visual C++ toolchain, the meaning of `wchar_t` comes from `wchar.h`, where it is defined as `unsigned short`.
- If the including code is written in C++, then for the Microsoft Visual C++ toolchain, the meaning of `wchar_t` depends on how you set the `/Zc:wchar_t` compiler flag.
  - If you specify `/Zc:wchar_t`, or don't specify anything at all, then `wchar_t` represents a unique data type whose binary representation is a 16-bit integer. We learned about this special data type some time ago.
  - If you specify `/Zc:wchar_t-`, then `wchar_t` is not predefined, and the Windows header files define it to mean `unsigned short`.

But at least from the IDL compiler's point of view, all this nonsense is irrelevant. The IDL compiler just emits `wchar_t` into the header file and leaves it to the C or C++ compiler to figure out what it means.

There is however, one frustrating wrinkle to this seemingly simple plan.

The Windows IDL compiler predates the standardization of `wchar_t`, and for historical reasons, it lets you provide your own definition of `wchar_t`.

```
typedef unsigned short wchar_t;
```

If you choose to define it manually, then the IDL compiler dutifully copies that definition into the generated header file.

And now you're at war with the C++ compiler, because your definition of `wchar_t` as `unsigned short` will conflict with the standard built-in definition of `wchar_t` as a unique type. In order to avoid this problem, you'll have to use the `/Zc:wchar_t-` option to tell the C++ compiler not to treat `wchar_t` as a unique type and allow it to be given a custom definition.

This whole nonsense about letting you provide a custom definition for `wchar_t` exists only for backward compatibility, and you shouldn't do it if you know what's good for you. To avoid making things *too* crazy, the only allowable custom definition for `wchar_t` is as `unsigned short`. Any other definition is rejected.