What's the deal with OLDNAMES.LIB?

devblogs.microsoft.com/oldnewthing/20200730-00

July 30, 2020

Raymond Chen

Set the time machine to 1988. The C language has not yet been standardized. Everybody had their own libraries for doing stuff, and some of them even pretended to be compatible with each other. The Microsoft C compiler, for example, came with a bunch of functions like `unlink` and `stat` to provide sort-of compatibility with Unix-sort-of source code.

And then the C standard was ratified in 1989. Now things got interesting, because those functions were not part of the C standard. Standard-conforming C programs were welcome to define functions with those names, and the rules say that this is perfectly legal and are not references to the identically-named Unix-like functions.

Now, there was a lot of code written for the Microsoft C toolchain that used these sort-of-Unix-ish functions, and renaming those functions would cause those programs to break.

So should we rename the Unix-ish functions in the Microsoft C library, in order to conform to the C standard? Or should we leave the functions in the Microsoft C library under their pre-standard names, to keep existing code working?

Let's do both!

The Microsoft C library renamed these Unix-ish function to have a leading underscore. So `unlink` became `_unlink`, and so on. A program that didn't use the Unix-ish library functions could define its own function called `unlink`, and everything would work just fine. But if the program actually wanted to use the `unlink` function from the Unix-ish library, this magic library `OLDNAMES.LIB` would step in.

The `OLDNAMES.LIB` library doesn't contain any code of its own. Rather, it contains name redirections that say, "Hey, I have a symbol called `unlink`, in case you were looking for it. Wait, you want to know what this symbol represents? Um, it represents a symbol named `_unlink`. Good luck with that."

If the linker cannot find a symbol named `unlink`, it turns to `OLDNAMES.LIB` as a library of last resort, and that library resolves the symbol `unlink`, but replaces it with an unresolved symbol named `_unlink`. The linker then goes through the symbol resolution process again,

and this time it finds `_unlink` in the regular C library.

The net result is that your attempt to call the function `unlink` got redirected to the function `_unlink`, but only if you didn't already have a function named `unlink`.

You can suppress the `OLDNAMES.LIB` library in various ways, most notably by passing the `/NODEFAULTLIB` flag, which can be abbreviated to the somewhat enigmatic `/NOD`.

Raymond Chen

Follow