Looking at world through __stdcall-colored glasses

devblogs.microsoft.com/oldnewthing/20210901-00

September 1, 2021

Raymond Chen

Windows core components are compiled with the `/Gz` flag, which sets `__stdcall` as the default calling convention. On x86-32, the `__stdcall` calling convention is slightly more efficient than `__cdecl` because the stack cleanup is done at function return, rather than at the call site.[1]

This configuration for Windows core components does mean that component authors wear `__stdcall`-colored glasses when they write their header files.

```
void DoSomething(int a, int b, int c);
```

This declares the function `DoSomething` with no explicit calling convention, which means that the compiler will use whatever calling convention was set as the default. For Windows core component authors, this is `__stdcall`, but for everybody else, the default is `__cdecl`.

And since the function is implemented in a Windows core component, the actual calling convention in the implementation is `__stdcall`.

The consequence of this situation is that the header file works great for the team that wrote the code, and their unit tests work great, and their test apps work great, everything works great. But anybody outside Windows who tries to call the function will probably be calling it with `__cdecl`, and then exciting things will happen.

“Works on my machine!”

To be fair, this is something that is easily overlooked, and the whole concept of calling conventions may not be something many developers are familiar with in the first place. It’s not like there’s a computer science course on ABIs and calling conventions.[2]

What this means for you is that if you see a Windows header file that declares a function or function pointer without an explicit calling convention, you first guess should be that the calling convention is `__stdcall`.

```
// bad header file

typedef void (*WIDGETFILTERPROC)(int a, int b);
void FilterWidgets(int c, WIDGETFILTERPROC filter);
```

Your first guess should be that the header file was intended to be written like this:

```
typedef void (CALLBACK *WIDGETFILTERPROC)(int a, int b);
void WINAPI FilterWidgets(int c, WIDGETFILTERPROC filter);
```

Next time, we'll look into the wages of this sin.

[1] On other architectures, `__stdcall` is identical to `__cdecl`.

[2] Even if such a course existed, it probably was just a semester, and quickly forgotten. For example, I often see multithreading errors in code from younger developers: They probably did study multithreading at some point in their degree program, but it was likely just one week out of twelve-week course, and it wasn't reinforced by subsequent work, so it ended up forgotten.

Raymond Chen

Follow