The operating system doesn't know which language programs are written in – by the time the code hits the CPU, they all look the same

devblogs.microsoft.com/oldnewthing/20090824-00

August 24, 2009

Raymond Chen

Reader Will Rayer asks about "the degree to which 'plain vanilla' C Windows API code works under Vista with the native look and feel."

It works just as well as code written in any other language. The operating system doesn't know which language programs are written in. By the time the code reaches the CPU, they all look the same. It's just a bunch of instructions that occasionally call an API function. You can write it in C, C++, assembly, Delphi, Perl, whatever.

Of course, some languages are better-suited to calling Win32 than others. Win32 is a C-based API, in the sense that the way you call an exported function is expressed in a C header file, and `__stdcall` calling convention matches up reasonably well with the way C does things (once you convince your compiler to follow that convention). The way types are passed on the stack or in registers, how return values are represented, the fact that pointers are just the address of some blob of data, these all follow the C way of thinking. It stands to reason that the C language (and languages which follow in C's footsteps, like C++) have a pretty easy time of calling Win32 exported functions.

But that doesn't mean that those are the only languages. After all, at the end of the day, it's all machine code. If you can write assembly language that pushes the parameters in the right format in the right order, then you can use Win32 from assembly language. (There appears to be a whole subculture devoted to this endeavor.)

Now, it is indeed the case that COM programming is much more convenient in C++ because the COM object layout matches that of many C++ compilers. But that doesn't mean you can't use some other language to do it. As long as that language knows how to indirect through a vtable, you can use COM objects. Indeed, the COM header files go out of their way to make sure even you old-school C programmers can call COM objects. If you define the `COBJMACROS` symbol, then you get access to macros like this:

```
#define IPersistFile_GetClassID(This,pClassID)      \
    (This)->lpVtbl -> GetClassID(This,pClassID)
```

This snippet from the `objidl.h` header file is some syntactical sugar to help C programmers use COM. Under pure C, you would retrieve the `CLSID` from an `IPersistFile` interface pointer like this:

```
CLSID clsid;
IPersistFile* ppf = ...;
HRESULT hr = ppf->lpVtbl->GetClassID(ppf, &clsid);
```

The above macro at least removes the error potential of passing the wrong `this` pointer:

```
CLSID clsid;
IPersistFile* ppf = ...;
HRESULT hr = IPersistFile_GetClassID(ppf, &clsid);
```

If you want to write your programs in C, you still have a lot of company. Huge chunks of Windows are still written in the C language. Not that you can tell, because once the compiler is done doing its thing, the identity of the source language is long gone.

Raymond Chen

Follow