You can ask the compiler to answer your calling convention questions

devblogs.microsoft.com/oldnewthing/20130220-00

February 20, 2013

Raymond Chen

If you want to figure out some quirks of a calling convention, you can always ask the compiler to do it for you, on the not unreasonable assumption that the compiler understands calling conventions.

"When a `__stdcall` function returns a large structure by value, there is a hidden first parameter that specifies the address the return value should be stored. But if the function is a C++ instance method, then there is also a hidden `this` parameter. Which goes first, the return value parameter or the `this` pointer?"

This is another case of You don't need to ask me a question the compiler can answer more accurately.

```
struct LargeStructure
{
 char x[256];
};


class Something
{
public:
 LargeStructure __stdcall TestMe();
};


void foo(Something *something)
{
 LargeStructure x = something->TestMe();
}
```

You could compile this into a program and then look in the debugger, or just ask the compiler to generate an assembly listing. I prefer the assembly listing, since it saves a few steps, and the compiler provides helpful symbolic names.

```
  00015 mov     eax, DWORD PTR _something$[ebp]

; LargeStructure x = something->TestMe();

  00018 lea     ecx, DWORD PTR _x$[ebp]
  0001e push    ecx
  0001f push    eax
  00020 call    ?TestMe@Something@@
                QAG?AULargeStructure@@XZ
                ; Something::TestMe
```

We see that the last thing pushed onto the stack (and therefore the top parameter on the stack at the point of the call) is the `something` parameter, which is the `this` for the function.

Conclusion: The `this` pointer goes ahead of the output structure pointer.

Raymond Chen

Follow