Why does the debugger show me the wrong function?

devblogs.microsoft.com/oldnewthing/20050322-00

March 22, 2005

Raymond Chen

Often you'll be minding your own business debugging some code, and you decide to step into one function and the debugger shows that you're in some other function. How did that happen?

```
class Class1
{
public:
 int *GetQ() { return q; }
private:
 int *p;
 int *q;
};
class Class2
{
public:
 virtual int GetValue() { return value; }
private:
 int value;
};
```

You then step through code that does something like this:

```
int Whatever(Class2 *p)
{
 return p->GetValue();
}
```

And when you step into the call to `p->GetValue()` you find yourself in `Class1::GetQ`. What happened?

What happened is that the Microsoft linker combined functions that are identical **at the code generation level**.

```
?GetQ@Class1@@QAEPAHXZ PROC NEAR    ; Class1::GetQ, COMDAT
  00000 8b 41 04         mov     eax, DWORD PTR [ecx+4]
  00003 c3               ret     0
?GetQ@Class1@@QAEPAHXZ ENDP         ; Class1::GetQ
?GetValue@Class2@@UAEHXZ PROC NEAR  ; Class2::GetValue, COMDAT
  00000 8b 41 04         mov     eax, DWORD PTR [ecx+4]
  00003 c3               ret     0
?GetValue@Class2@@UAEHXZ ENDP       ; Class2::GetValue
```

Observe that at the object code level, the two functions are identical. (Note that whether two functions are identical at the object code level is highly dependent on which version of what compiler you're using, and with which optimization flags. Identical code generation for different functions occurs with very high frequency when you use templates.) Therefore, the linker says, "Well, what's the point of having two identical functions? I'll just keep one copy and use it to stand for both `Class1::GetQ` and `Class2::GetValue`."

```
0:000> u Class1::GetQ
010010d6 8b4104          mov     eax,[ecx+0x4]
010010d9 c3              ret
0:000> u Class2::GetValue
010010d6 8b4104          mov     eax,[ecx+0x4]
010010d9 c3              ret
```

Notice that the two functions were merged: The addresses are identical. That one fragment of code merely goes by two names. Therefore, when the debugger sees that you've jumped to `0x010010d6`, it doesn't know which of the names it should use, so it just picks on.

That's why it looks like you jumped to the wrong function.

To disable what is called "identical COMDAT folding", you can pass the `/OPT:NOICF` flag to the linker.

Raymond Chen

Follow