Decoding the parameters of a thrown C++ exception (0xE06D7363), revisited

devblogs.microsoft.com/oldnewthing/20160915-00

September 15, 2016

Raymond Chen

When I explained how to decode the parameters of a thrown C++ exception, I noted that the mysterious second parameter at index 1 "is a pointer to the object being thrown (sort of)."

I've since learned more about that mysterious second parameter at index 1: It really is a pointer to the object being thrown, no header or anything. I was faked out because in the crash dumps I looked at, the object being thrown was a pointer to a stack object, so we had two things on the stack (the pointer being thrown, and the object being pointed to), and therefore I could dump my way from one to the other.

Basically, I got lucky.

So that's the story of the second parameter. It's a pointer to the object being thrown, but if the object being thrown is itself a pointer, then you'll have to perform another indirection to find the underlying object.

Here's an example from a `throw E_FAIL`:

```
0:000> .exr 02e0f704
ExceptionAddress: 75dd4878 (KERNELBASE!RaiseException+0x00000048)
   ExceptionCode: e06d7363 (C++ EH exception)
  ExceptionFlags: 00000001
NumberParameters: 3
   Parameter[0]: 19930520
   Parameter[1]: 02e0fc6c
   Parameter[2]: 00d1263c
0:000> dd 02e0fc6c l1
02e0fc6c  80004005 // the thing being thrown is value
```

And here's `throw "oops"`:

```
0:000> .exr 02a6f3e4
ExceptionAddress: 75dd4878 (KERNELBASE!RaiseException+0x00000048)
   ExceptionCode: e06d7363 (C++ EH exception)
  ExceptionFlags: 00000001
NumberParameters: 3
   Parameter[0]: 19930520
   Parameter[1]: 02a6f944
   Parameter[2]: 008d22b0
0:000> dd 02a6f944 l1
02a6f944  008d10e0
// since the thing being thrown is a pointer, we dump what it points to.
0:000> da 008d10e0
008d10e0  "oops" // what it points to
```

Bonus chatter: The layout of the mysterious third parameter (index 2) is explained in the `ehdata.h` header file that comes with Visual Studio.

Raymond Chen

Follow