Remember that in a stack trace, the addresses are return addresses, not call addresses

devblogs.microsoft.com/oldnewthing/20170505-00

May 5, 2017

Raymond Chen

You may be faced with a stack trace like this:

```
00000000`001ebff0 00000000`ff6e2a94 ABC!CUIController::UpdateDisplay+0x156
[c:\src\abc\uicontroller.cpp @ 152]
00000000`001ec060 00000000`ff6e2f70 ABC!CUIController::displayEvent+0xea
[c:\src\abc\uicontroller.cpp @ 930]
00000000`001ec090 00000000`ff6e2eef ABC!CEventRouter::fire+0x34
[c:\src\abc\eventrouter.cpp @ 998]
00000000`001ec0d0 00000000`ff6e3469 ABC!CEngineState::storeAndFire+0x126
[c:\src\abc\enginestate.cpp @ 653]
00000000`001ec110 00000000`ff6e4149 ABC!CEngine::SetDisplayText+0x39
[c:\src\abc\engine.cpp @ 749]
```

But when you go to look at, say, line 930 of the file `uicontroller.cpp`, you don't see a call to `UpdateDisplay`:

```
// line 930
    DoSomethingElseEntirely(GetCurrentWidget(), false);
```

Why is the debugger saying that the call to `UpdateDisplay` came from line 930 when there's no call to `UpdateDisplay` anywhere in sight?

Recall that the stack trace extracts information from the stack. And the thing on the stack is the return address; in other words, it's the instruction that will execute after the called function returns. It's the instruction *after* the `call` instruction.

This means if that the call instruction was the last instruction for a line of code, then the return address will point to the first instruction of the *next* line of code.

Note also that the previous line of code might not be the one that comes before it in the source code.

```
    if (some_condition) {
        UpdateDisplay(); // line 924
    } else {
        OtherFunction(); // line 926
    }

    // line 930
    DoSomethingElseEntirely(GetCurrentWidget(), false);
```

If the optimizer concludes (perhaps as a result of profiling feedback) tnat `some_condition` is nearly always true, it may decide to move the entire `else` clause out-of-line:

```
    cmp [some_condition], 0
    jz  rare_case
    call UpdateDisplay
line_930:
    call GetCurrentWidget
    push 0
    push eax
    call DoSomethingElseEntirely
    ...

rare_case:
    call OtherFunction
    jmp  line_930
```

To see what line of code issued the call, you need to look at the address of the call.

```
0:00> u 0xff6e2a94-5 L1
ABC!CUIController::displayEvent+0xe5 [c:\src\abc\uicontroller.cpp @ 924]
00000000`ff6e2a8f call CUIController::UpdateDisplay
```

Aha, it was line 924.

Now, the actual `call` instruction might not have been a direct call. It could have been a memory indirect call, or a call through a register. I would just subtract 1 to get to the end of the previous instruction. It will disassemble as garbage, but that's okay.

```
0:00> u 0xff6e2a94-1 L1
ABC!CUIController::displayEvent+0xe9 [c:\src\abc\uicontroller.cpp @ 924]
00000000`ff6e2a9e add al, ch
```

Bonus chatter: This reminds me of a quirk of the 6502 processor: When it pushed the return address onto the stack, it actually pushed the return address *minus one*. This is an artifact of the way the 6502 is implemented, but it results in the nice feature that the stack trace gives you the line number of the call instruction.

Of course, this is all hypothetical, because 6502 debuggers didn't have fancy features like stack traces or line numbers.

Raymond Chen

Follow