几种典型程序Button处理代码的定位

转载

Yuri800 于 2017-05-15 22:59:12 发布 1403 收藏
分类专栏： 调试 器之卷 文章标签： 调试


调试 同时被 2 个专栏收录
122 篇文章 3 订阅
订阅专栏


器之卷
68 篇文章 1 订阅
订阅专栏

转自看雪:http://bbs.pediy.com/thread-20078.htm

首先

1 od 下运行程序，F12 暂停；

2 View菜单中选击Windows项，在打开的窗口中可以从Title栏看到目标按钮，从而找到它的Handle(xxxxxxxx)；

对不同平台生成的程序，分别处理：

一、VB, Delphi, CBuilder 程序：

3 在CallWindowProcA入口下条件断点: [esp+8]==xxxxxxxx && [esp+0c]==202；

4 F9继续程序，点击目标按钮，程序中断；

5 Alt+F4，在代码段（.text）上下访问断点；

6 F9执行程序，程序中断,

1). VB程序中断在下面代码

PUSH DWORD PTR DS:[EAX+EBX] ; yyyyyyy

上面[EAX+EBX]的值（yyyyyy）就是我们要找的位置。

2). Delphi, CBuilder 的程序

程序直接断在我们要找的位置。

借moon一句，这姑妄称作CallWindowProcA条件断点加上code段内存断点法吧。

二、VC程序

又分MFC和Win32两种情况，二者相同之处：

3 在IsDialogMessageW入口下条件断点: [[esp+8]]==xxxxxxxx && [[esp+8]+4]==202

4 F9继续程序，点击目标按钮，程序中断；

5 Alt+F4，在代码段上下访问断点；

6 F9执行程序，程序中断, 注意这里虽然中断在code段，但却不是处理Button点击事件的代码处

这时：

对Win32程序，只需要按几下F7，当回到User32.dll领空后再重复一次第 5、6步就可以了；

而对于MFC程序，我们不得不多次重复这样的操作：单步回到MFC领空，再第 5、6步。好在已经看到大陆啦！

类比，这就叫IsDialogMessageW条件断点加上code段内存断点法了。

==========

备注一下:

我个人觉得看雪上这篇文章可取的地方是在od中断后在进程.text内存空间下访问断点，下面的列表dump自MFC Dlg程序响应WM_INITDLG消息时的堆栈。从输出来看只要在User32!InternalCallWinProc上下断点，当调试器中断，然后在.text段上下访问断点，再次运行就进会端在应用程序代码空间

```
0:000> kb
ChildEBP RetAddr  Args to Child
0023f160 7802f99f 0023fc2c 001301bc 0023f15c subclassing!CMyEdit1::PreSubclassWindow [c:\studio\subclassing
0023f18c 7802fb60 001301bc 0023fc2c 0028d086 mfc100d!CWnd::SubclassWindow+0x2f [f:\dd\vctools\vc7libs\ship\
0023f1a4 010834d4 000003e8 0023fb98 00000000 mfc100d!CWnd::SubclassDlgItem+0x70 [f:\dd\vctools\vc7libs\ship
0023f2a8 77f4b3b0 0023fb98 0023f2e0 7645c4b7 subclassing!CSubclassingDlg::OnInitDialog+0x44 [c:\studio\subc
0023f2b4 7645c4b7 000f0478 00000110 001301bc mfc100d!AfxDlgProc+0x40 [f:\dd\vctools\vc7libs\ship\atlmfc\src
0023f2e0 76475825 77f4b370 000f0478 00000110 USER32!InternalCallWinProc+0x23
0023f35c 764759c3 00000000 77f4b370 000f0478 USER32!UserCallDlgProcCheckWow+0xd6
0023f3a4 764671d6 00000000 00000110 001301bc USER32!DefDlgProcWorker+0xa8
0023f3c0 7645c4b7 000f0478 00000110 001301bc USER32!DefDlgProcA+0x22
0023f3ec 7645c5b7 764671b4 000f0478 00000110 USER32!InternalCallWinProc+0x23
0023f464 76451b01 00000000 764671b4 000f0478 USER32!UserCallWinProcCheckWow+0x14b
0023f494 76472bbe 764671b4 000f0478 00000110 USER32!CallWindowProcAorW+0x99
0023f4b4 78025e94 764671b4 000f0478 00000110 USER32!CallWindowProcA+0x1b
0023f4d8 7802450d 00000110 001301bc 00000000 mfc100d!CWnd::DefWindowProcA+0x34 [f:\dd\vctools\vc7libs\ship\
0023f4f4 77f4ca44 0023fb98 77c9e2d4 78109260 mfc100d!CWnd::Default+0x3d [f:\dd\vctools\vc7libs\ship\atlmfc\
0023f518 78028651 001301bc 00000000 ce5d84c6 mfc100d!CDialog::HandleInitDialog+0xe4 [f:\dd\vctools\vc7libs\
0023f688 78027dd2 00000110 001301bc 00000000 mfc100d!CWnd::OnWndMsg+0x841 [f:\dd\vctools\vc7libs\ship\atlmf
0023f6a8 78024383 00000110 001301bc 00000000 mfc100d!CWnd::WindowProc+0x32 [f:\dd\vctools\vc7libs\ship\atlm
0023f728 78024976 0023fb98 000f0478 00000110 mfc100d!AfxCallWndProc+0xf3 [f:\dd\vctools\vc7libs\ship\atlmfc
0023f748 77e10c7b 000f0478 00000110 001301bc mfc100d!AfxWndProc+0xa6 [f:\dd\vctools\vc7libs\ship\atlmfc\src
0023f784 7645c4b7 000f0478 00000110 001301bc mfc100d!AfxWndProcBase+0x5b [f:\dd\vctools\vc7libs\ship\atlmfc
0023f7b0 7645c5b7 77e10c20 000f0478 00000110 USER32!InternalCallWinProc+0x23
0023f828 76455264 00000000 77e10c20 000f0478 USER32!UserCallWinProcCheckWow+0x14b
0023f868 76474f3c 000ff018 010ffe90 001301bc USER32!SendMessageWorker+0x4d0
0023f924 7647532a 01070000 000f0478 000000bc USER32!InternalCreateDialog+0xb0d
0023f948 76467208 01070000 01095738 00000000 USER32!CreateDialogIndirectParamAorW+0x33
0023f968 77f4bf44 01070000 01095738 00000000 USER32!CreateDialogIndirectParamA+0x1b
0023fa30 77f4c6f8 01095738 00000000 01070000 mfc100d!CWnd::CreateDlgIndirect+0x264 [f:\dd\vctools\vc7libs\s
0023faa4 01082d8f cfb511a6 00000000 00000000 mfc100d!CDialog::DoModal+0x1a8 [f:\dd\vctools\vc7libs\ship\atl
0023fd34 7805a684 00000000 00000000 00000000 subclassing!CSubclassingApp::InitInstance+0x7f [c:\studio\subc
0023fd58 0108836a 01070000 00000000 00282b2b mfc100d!AfxWinMain+0x84 [f:\dd\vctools\vc7libs\ship\atlmfc\src
0023fd70 01084c30 01070000 00000000 00282b2b subclassing!WinMain+0x1a [f:\dd\vctools\vc7libs\ship\atlmfc\sr
0023fe18 010849bf 0023fe2c 76cfef8c 7ffdc000 subclassing!__tmainCRTStartup+0x260 [f:\dd\vctools\crt_bld\sel
0023fe20 76cfef8c 7ffdc000 0023fe6c 76fa367a subclassing!WinMainCRTStartup+0xf [f:\dd\vctools\crt_bld\self_
0023fe2c 76fa367a 7ffdc000 771b62f1 00000000 kernel32!BaseThreadInitThunk+0xe
0023fe6c 76fa364d 010816cc 7ffdc000 00000000 ntdll!__RtlUserThreadStart+0x70
0023fe84 00000000 010816cc 7ffdc000 00000000 ntdll!_RtlUserThreadStart+0x1b
```

看雪上有些文章称 User32!InternalCallWinProc函数为万能消息断点还是有一定道理的: