2021/01/28_调试器_第4课_调试器内存断点的编写

笔记本： 调试器
创建时间： 2021/1/28 星期四 10:04
作者： ileemi

显示寄存器值

r：显示被调试进程的各个寄存器数值（显示汇编，单步异常，断点异常时都可显示）

> 运行到指定位置（F4）：在指定位置设置一个一次性的断点，程序运行到断点处（"g" 命令后面添加一个地址参数）。

调试器的核心功能就是：断点、单步

> 执行到返回（Ctrl + F9）：汇编代码一条一条的执行（步过），到ret指令时，就暂停。
> 执行到用户代码（Alt + F9）：程序需要执行到主模块代码中，不能停到系统模块中（可一条一条指令执行，拿eip和模块基址做比较（模块的大小）），判断EIP是在哪个模块中。

设置内存断点

> 实现：修改目标内存的读取权限，C0000005H异常为CPU异常（权限由操作系统给，检测由CPU完成）。16位CPU没有内存断点，因为其内存没有权限问题。实现原理：对目标内存进行写入时，将其内存的写入权限修改位不可写入，当写入时，就会产生异常，此时就知道那条 "eip" 访问了目标内存，就可以对其下断点。

> 调试器添加 "bm" 命令，注意设置内存断点时，分析其功能（**读取、写入、执行、访问**），重复断点需要断点和单步配合使用。

附加：调试运行起来的程序（API：**DebugActiveProcess**），操作系统会让被附加程序产生一条int3指令。反调试检测程序是否被调试器进行调试，检测代码一般写在程序的入口处，调试器可以让程序运行起来，等到检测代码执行完毕后，在附加目标程序进行调试。

修改内存保护（VirtualProtectEx）属性前不确定原来的保护属性时可通过 "VirtualQueryEx" 查询，内存属性主要就是：可读、可写、可执行。

保存访问的内存，以及字节长度。内存访问异常错误码：C0000005H
判读异常信息可通过 "EXCEPTION_DEBUG_INFO" 结构体中的 "ExceptionRecord" 中的 "ExceptionInfomation" 数组的第二个元素得出异常地址，同时还需要判断产生异常的地址是否在访问内存的区间中，满足条件就恢复内存属性。

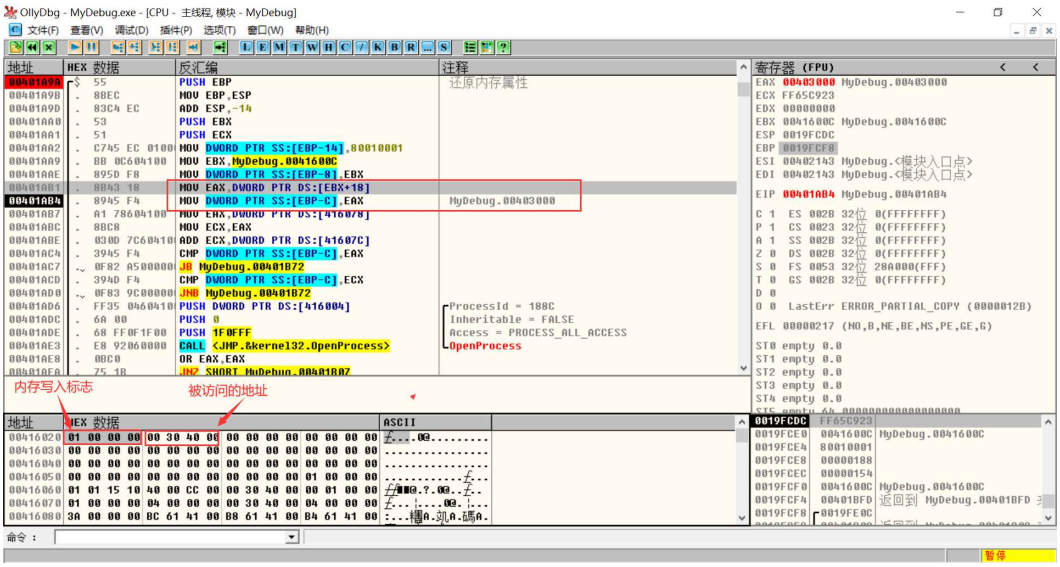
系统创建进程的时候，让其产生一个异常，并还原代码，注意 "断步配合" 时逻辑要搞好，用户通过调试器调试程序时，就需要一条一条汇编代码执行并显示，运行程序时就不需要显示汇编代码。断点的单步是为了修改以及还原程序的汇编代码。

> 如果要内存断点一直有效，也需要 "断步配合"，内存访问异常就设置一个内存断点，并执行单步，在还原内存保护属性之后再次修改内存保护属性使其产生内存访问异常。

```
CMyDebug::OnDebugEvent ExceptionCode:80000003 ExceptionAddress:00401008
00401008        MOV ESI, 00000000
eax=74BE343B ebx=7EFDE000 ecx:00000000 edx=00401008 eip:00401008
esi=00000000 edi=00000000 ebp=0018FF94 esp=0018FF8C eflags:00000344
cmd:g
CMyDebug::OnDebugEvent ExceptionCode:80000004 ExceptionAddress:0040100D
CMyDebug::OnDebugEvent ExceptionCode:C0000005 ExceptionAddress:0040100F
0040100F         INC DWORD PTR DS:[00403000]
cmd:g
CMyDebug::OnDebugEvent ExceptionCode:80000004 ExceptionAddress:00401015
CMyDebug::OnDebugEvent ExceptionCode:C0000005 ExceptionAddress:0040100F
0040100F         INC DWORD PTR DS:[00403000]
cmd:g
CMyDebug::OnDebugEvent ExceptionCode:80000004 ExceptionAddress:00401015
CMyDebug::OnDebugEvent ExceptionCode:C0000005 ExceptionAddress:0040100F
0040100F         INC DWORD PTR DS:[00403000]
cmd:g
CMyDebug::OnDebugEvent ExceptionCode:80000004 ExceptionAddress:00401015
CMyDebug::OnLoadDll lpBaseOfDll:10000000
CMyDebug::OnUnLoadDll lpBaseOfDll:10000000
CMyDebug::OnLoadDll lpBaseOfDll:10000000 C:\Windows\SysWOW64\SOGOUPY.IME
CMyDebug::OnLoadDll lpBaseOfDll:6F910000 C:\Windows\SysWOW64\MSIMG32.dll
CMyDebug::OnLoadDll lpBaseOfDll:76910000 C:\Windows\syswow64\WS2_32.dll
CMyDebug::OnLoadDll lpBaseOfDll:74BB0000 C:\Windows\syswow64\NSI.dll
CMyDebug::OnLoadDll lpBaseOfDll:6F8D0000 C:\Windows\SysWOW64\OLEACC.dll
CMyDebug::OnLoadDll lpBaseOfDll:6F610000 C:\Windows\SysWOW64\ntmarta.dll
CMyDebug::OnLoadDll lpBaseOfDll:76820000 C:\Windows\syswow64\WLDAP32.dll
```

Windows操作系统中内存的访问属性一定都是**可读、可执行**的，修改内存属性不会产生异常。如果希望样目标内存产生异常，就需要修改目标内存属性为无效的内存属性（**PAGE_NOACCESS**，就会产生内存访问异常：C0000005H）。区分内存异常产生的原因就可以通过 "EXCEPTION_DEBUG_INFO" 结构体中的 "ExceptionRecord" 中的 "ExceptionInfomation" 数组的第一个元素进行判断。

- 0x00000000：读取内存
- 0x00000001：写入内存
- 0x00000008：执行内存代码



> 可以把内存产生异常的原因当作断点的类型。修改内存属性的字节单位是 "页（0x1000）"，**所以修改内存属性会影响一个页甚至两个页内存的保护属性。影响到到内存可能会产生异常，所以就需要对其进行异常处理，不能将异常交还给程序去处理。**

内存断点跨页问题

还有可能程序中本来就由异常，自己也会对其进行处理，但是由于修改内存保护属性的影响，就会导致程序中的异常由调试器进行处理。修改内存保护属性后所产生的异常是没有办法判断异常的来源的，可行办法就是，调试器在设计时，在受影响的内存范围中产生的异常调试器全部接收（都下断点停下），之后问调试器使用者是否处理异常（类似 "ollydbg" 的 Shift + F9（将异常还给调试器使用者决定）、F9（异常由调试器处理））。