

Debugger 用法详解

目录

1. Debugger 用法详解	2
1.1 命令概览.....	2
1.1.1 断点设置命令	2
数据命令	2
控制调试运行命令.....	3
其它控制命令	3
1.2 命令详解.....	3
1.2.1 断点设置命令	3
数据命令	5
控制调试运行命令.....	5
其它控制命令	6

Debugger 用法详解

这里写些介绍的

1.1 命令概览

Debugger 以命令方式工作，以下根据功能归纳当前版本 Debugger 支持的所有标准命令。

控制调试目标进程执行：

1.1.1 断点设置命令

- BA：硬件断点
- BP：软件断点(int3)
- BL：列出所有断点
- BC：清除所有断点
- BD：禁止断点
- BE：重新启用断点
- BR：移除断点
- BM：添加一个内存断点
- BY：移除一个内存断点

数据命令

- D：查看内存数据

控制调试运行命令

- G: 让程序运行起来, 或者运行到指定地址处
- T: 单步步入
- P: 单步步过

其它控制命令

- S: 单步记录(动态跟踪)
- R: 查看修改寄存器的值
- L: 查看 PE 信息
- V: 显示调试器版本
- Q: 结束调试会话
- O: 运行脚本
- .KILL: 结束调试进程
- .RESTART: 重新加载调试进程
- .SHOW: 显示已加载 DLL 名称

1.2 命令详解

1.2.1 断点设置命令

- BA: 硬件断点

ba 断点地址 长度 类型

Debugger 用法详解

注: 断点地址必须在有效的内存分页之内, 长度可以为 1、2 或 4, 类型可以为 Execute、Write 或 Read,也可以用三个单词的开头首字母。

■ BP : 软件断点(int3)

bp 断点地址

注: 断点地址必须在有效的内存分页之内。如果 Dr 寄存器没有使用的话, 会优先使用硬件断点代替。

■ BL : 列出所有断点

bl 无参数

注: 列出当前所有的断点信息, 包括 Int3 断点, 硬件断点以及内存断点, 如果用户设了 Int3 断点, 但是用户却没有设硬件断点, Int3 断点将会被硬件断点代替。

■ BC : 清除所有断点

bc 无参数

注: 此命令将清除所有的 Int3 断点和硬件断点, 对内存断点没有影响。

■ BD : 禁止断点

bd 断点地址

注: 只影响 Int3 断点和硬件断点, 对内存断点没有影响。

■ BE : 重新启用断点

be 断点地址

注: 只影响 Int3 断点和硬件断点, 对内存断点没有影响。

■ BR: 移除断点

br 断点地址

Debugger 用法详解

注: 只移除 Int3 断点和硬件断点, 对内存断点没有影响。

■ BM: 添加一个内存断点

bm 断点地址 长度 属性

注: 断点地址必须在有效的内存分页当中, 长度可以为任意长度, 如果跨内后的内存分页无效, 则自动修改断点的长度, 属性可以为 Read、Write(单词首字母也可以), 如果内存分页没有那个属性, 则添加出错。

■ BY: 移除一个内存断点

by 断点地址

数据命令

■ D: 查看内存数据

d 不带参数

从当前 Eip 处开始显示 0x80 个字节的内存数据。

d 内存地址

从指定的内存地址处开始显示 0x80 个字节的内存数据。

D 内存地址 终止内存地址

从指定的内存地址开始显示, 直到终止内存地址处。

D 内存地址 长度

从指定的内存地址开始显示, 显示指定的长度的个数。

控制调试运行命令

■ G: 让程序运行起来, 或者运行到指定地址处

Debugger 用法详解

G 不带参数

直接让程序运行起来，如果遇到断点则中断下来。

G 目标地址

直接让程序运行到指定地址处，取消所有的 Int3 断点和硬件断点，内存断点不受影响，如果中间遇到内存断点，则中断下来。

■ T: 单步步入

T

单步步入，遇到 call 跟进。

■ P: 单步步过

P

单步步进，遇到 call 不跟进。

其它控制命令

■ S: 单步记录(动态跟踪)

S 记录起始范围 记录终止范围 [保存文件名]

■ R: 查看修改寄存器的值

R 不带参数

显示当前的寄存器的值。

R 寄存器名

修改指定寄存器的值。

■ L: 查看 PE 信息

Debugger 用法详解

L

显示被调试程序的 PE 基本信息。

- V: 显示调试器版本

V

显示调试器版本详细信息。

- Q: 结束调试会话

Q

结束被调试进程且并闭调试器进程

- O: 运行脚本

O 脚本路径名

自动运行脚本，脚本不支持注释。语法和手动控制的一样。

- .KILL: 结束被调试进程

.kill

结束被调试进程。

- .RESTART: 重新加载被调试进程

.restart

重新加载被调试进程，这个功能还在测试当中，如果有下内存断点，将会引发错误。

- .show: 显示已加载 DLL 名称

.show

显示当前已经加载的 DLL 的名称。

Debugger 用法详解

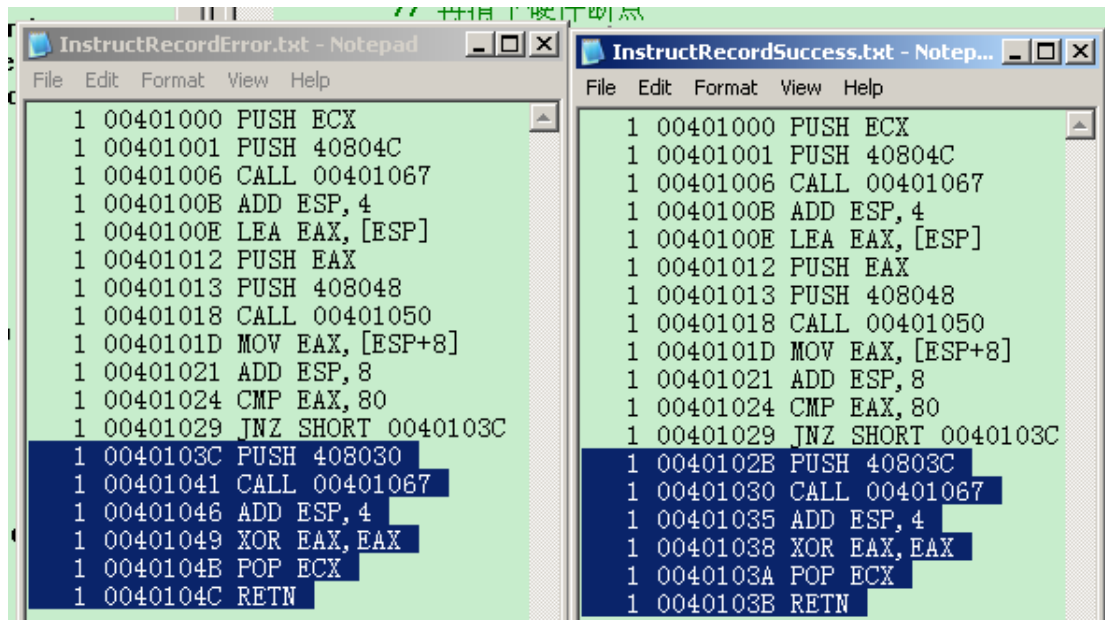
```
004013A3  6A FF          PUSH  -1
004013A5  68 D0404000    PUSH  4040D0
004013AA  68 D41E4000    PUSH  401ED4
004013AF  64:A1 00000000  MOV  EAX,FS:[0]
004013B5  50            PUSH  EAX
004013B6  64:8925 00000000 MOV  FS:[0],ESP
-u
004013BD  83EC 58        SUB  ESP,58
004013C0  A1 5C554000    MOV  EAX,[40555C]
004013C5  8915 5C554000  MOV  [40555C],EDX
004013CB  90            NOP
004013CC  33D2          XOR  EDX,EDX
004013CE  8AD4          MOV  DL,AH
004013D0  8915 28554000  MOV  [405528],EDX
004013D6  8BC8          MOV  ECX,EAX
-bm 40555c 4 r
添加内存断点成功!
-g
EAX=0012FFE0  EBX=7FFDE000  ECX=0012FFB0  EDX=7C90E514
ESI=00000000  EDI=00000000  ESP=0012FF58  EBP=0012FFC0
EIP=004013C0  iopl= 0      nv up ei pl nz ac po nc
cs=001B  ss=0023  ds=0023  es=0023  fs=003B  gs=0000
004013C0  A1 5C554000    MOV  EAX,[40555C]
内存中断!
```

记录功能演示:

```
int3中断!
-s 401000 40104d InstructRecordSuccess.txt
00401000: PUSH  ECX
00401001: PUSH  40804C
00401006: CALL  00401067
type passwd(number): 0040100B: ADD  ESP,4
0040100E: LEA  EAX,[ESP]
00401012: PUSH  EAX
00401013: PUSH  408048
00401018: CALL  00401050
128
0040101D: MOV  EAX,[ESP+8]
00401021: ADD  ESP,8
00401024: CMP  EAX,80
00401029: JNZ  SHORT 0040103C
0040102B: PUSH  40803C
00401030: CALL  00401067
Success!
00401035: ADD  ESP,4
00401038: XOR  EAX,EAX
```

通过对比，很快可以发现哪里不同了！大家也知道这个功能有啥用的了！

Debugger 用法详解



```
InstructRecordError.txt - Notepad
File Edit Format View Help
1 00401000 PUSH ECX
1 00401001 PUSH 40804C
1 00401006 CALL 00401067
1 0040100B ADD ESP, 4
1 0040100E LEA EAX, [ESP]
1 00401012 PUSH EAX
1 00401013 PUSH 408048
1 00401018 CALL 00401050
1 0040101D MOV EAX, [ESP+8]
1 00401021 ADD ESP, 8
1 00401024 CMP EAX, 80
1 00401029 JNZ SHORT 0040103C
1 0040103C PUSH 408030
1 00401041 CALL 00401067
1 00401046 ADD ESP, 4
1 00401049 XOR EAX, EAX
1 0040104B POP ECX
1 0040104C RETN

InstructRecordSuccess.txt - Notepad...
File Edit Format View Help
1 00401000 PUSH ECX
1 00401001 PUSH 40804C
1 00401006 CALL 00401067
1 0040100B ADD ESP, 4
1 0040100E LEA EAX, [ESP]
1 00401012 PUSH EAX
1 00401013 PUSH 408048
1 00401018 CALL 00401050
1 0040101D MOV EAX, [ESP+8]
1 00401021 ADD ESP, 8
1 00401024 CMP EAX, 80
1 00401029 JNZ SHORT 0040103C
1 0040102B PUSH 40803C
1 00401030 CALL 00401067
1 00401035 ADD ESP, 4
1 00401038 XOR EAX, EAX
1 0040103A POP ECX
1 0040103B RETN
```