# 分析结果

最后把dump文件上传，可以看到最终的结果：

Instant Online Crash Analysis, brought to you by OSR Open Systems Resources, Inc.

Primary Analysis

Crash Dump Analysis provided by OSR Open Systems Resources, Inc. (http://www.osr.com)

Online Crash Dump Analysis Service

See http://www.osronline.com for more information

Windows 7 Kernel Version 7600 UP Free x86 compatible

Product: WinNt, suite: TerminalServer SingleUserTS

Built by: 7600.16385.x86fre.win7\_rtm.090713-1255

Machine Name:

Kernel base = 0x83c07000 PsLoadedModuleList = 0x83d46570

Debug session time: Thu Aug 2 01:05:11.006 2012 (UTC - 4:00)

System Uptime: 0 days 0:02:48.678

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Bugcheck Analysis \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IRQL\_NOT\_LESS\_OR\_EQUAL (a)

An attempt was made to access a pageable (or completely invalid) address at an

interrupt request level (IRQL) that is too high. This is usually

caused by drivers using improper addresses.

If a kernel debugger is available get the stack backtrace.

Arguments:

Arg1: 00000010, memory referenced

Arg2: 00000002, IRQL

Arg3: 00000000, bitfield :

bit 0 : value 0 = read operation, 1 = write operation

bit 3 : value 0 = not an execute operation, 1 = execute operation (only on chips which support this level of status)

Arg4: 83c19e6c, address which referenced memory

Debugging Details:

------------------

TRIAGER: Could not open triage file : e:\dump\_analysis\program\triage\modclass.ini, error 2

READ\_ADDRESS: GetPointerFromAddress: unable to read from 83d66700

Unable to read MiSystemVaType memory at 83d460c0

00000010

CURRENT\_IRQL: 2

FAULTING\_IP:

nt!IoDetachDevice+2f

83c19e6c 8b4710 mov eax,dword ptr [edi+10h]

CUSTOMER\_CRASH\_COUNT: 1

DEFAULT\_BUCKET\_ID: INTEL\_CPU\_MICROCODE\_ZERO

BUGCHECK\_STR: 0xA

PROCESS\_NAME: System

TRAP\_FRAME: 8d02ba50 -- (.trap 0xffffffff8d02ba50)

ErrCode = 00000000

eax=00000000 ebx=00000000 ecx=83d67380 edx=00000000 esi=00000000 edi=00000000

eip=83c19e6c esp=8d02bac4 ebp=8d02bad0 iopl=0 nv up ei pl zr na pe nc

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010246

nt!IoDetachDevice+0x2f:

83c19e6c 8b4710 mov eax,dword ptr [edi+10h] ds:0023:00000010=????????

Resetting default scope

LAST\_CONTROL\_TRANSFER: from 83c19e6c to 83c3fb5b

STACK\_TEXT:

8d02ba50 83c19e6c badb0d00 00000000 00000000 nt!KiTrap0E+0x2cf

8d02bad0 9d5b6331 00000000 8d02bcbc 83dc00b3 nt!IoDetachDevice+0x2f

WARNING: Stack unwind information not available. Following frames may be wrong.

8d02badc 83dc00b3 86ca0350 00000000 94b34cd0 HelloDrv+0x1331

8d02bcbc 83da8344 00000001 00000000 8d02bce4 nt!IopLoadDriver+0x90d

8d02bd00 83c62043 94b34cd0 00000000 856bb798 nt!IopLoadUnloadDriver+0x70

8d02bd50 83deed16 00000001 a925d6fa 00000000 nt!ExpWorkerThread+0x10d

8d02bd90 83c90159 83c61f36 00000001 00000000 nt!PspSystemThreadStartup+0x9e

00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x19

STACK\_COMMAND: kb

FOLLOWUP\_IP:

HelloDrv+1331

9d5b6331 ?? ???

SYMBOL\_STACK\_INDEX: 2

SYMBOL\_NAME: HelloDrv+1331

FOLLOWUP\_NAME: MachineOwner

MODULE\_NAME: HelloDrv

IMAGE\_NAME: HelloDrv.sys

DEBUG\_FLR\_IMAGE\_TIMESTAMP: 501a0673

FAILURE\_BUCKET\_ID: 0xA\_HelloDrv+1331

BUCKET\_ID: 0xA\_HelloDrv+1331

Followup: MachineOwner

---------

This free analysis is provided by OSR Open Systems Resources, Inc.

Want a deeper understanding of crash dump analysis? Check out our Windows Kernel Debugging and Crash Dump Analysis Seminar (opens in new tab/window)

Crash Code Links

Information About Address 0x10

Loaded Module List

Raw Stack Contents

Dump Header Information

Strings

# 蓝屏分析

CURRENT\_IRQL: 2

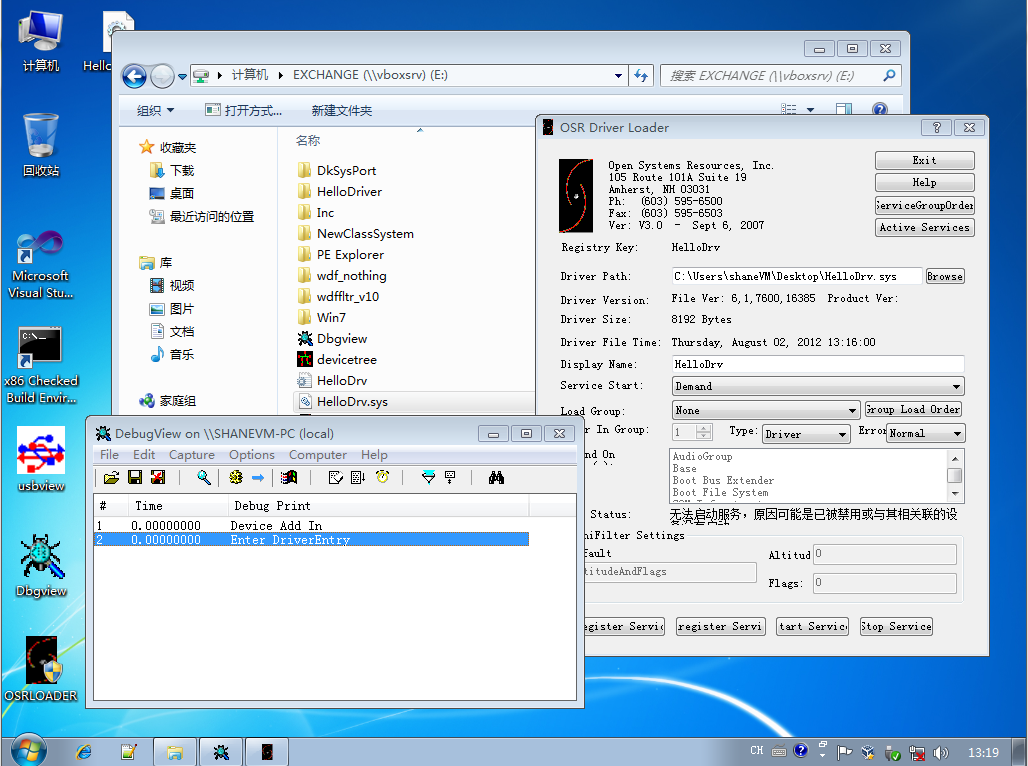
FAULTING\_IP:

nt!IoDetachDevice+2f

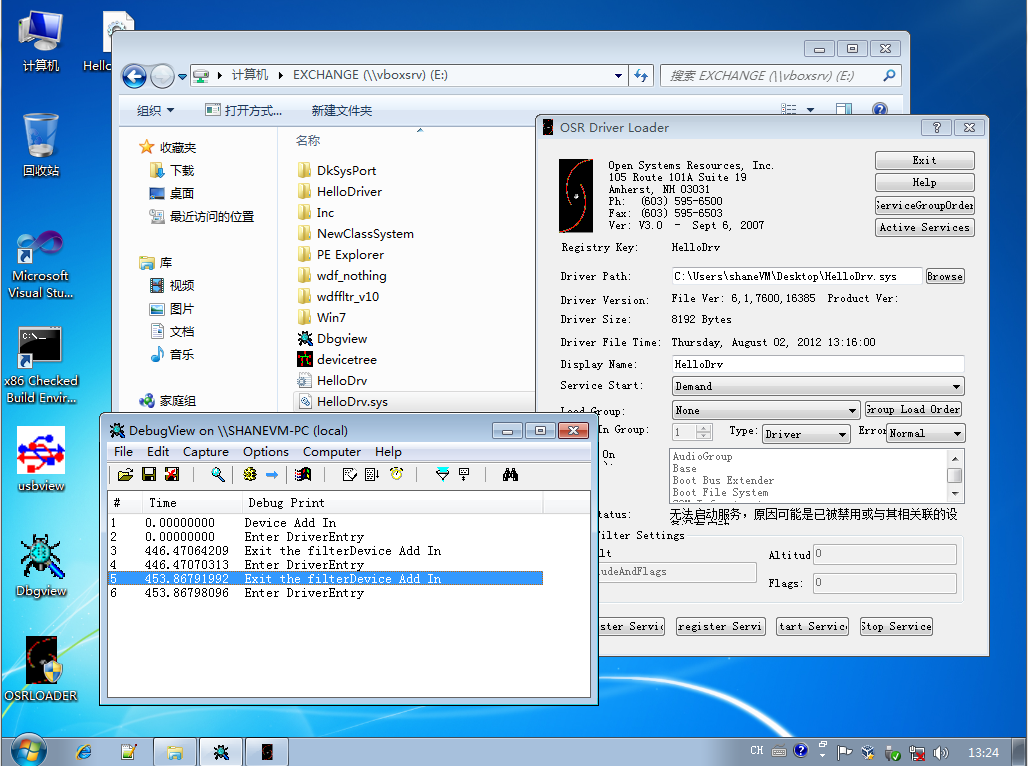
83c19e6c 8b4710 mov eax,dword ptr [edi+10h]

其中主要导致问题的为nt!IoDetachDevice+2f这个方法，之后发现的确这个方法没有仔细的考量。再注释掉之后重新编译测试。

# 更改调试

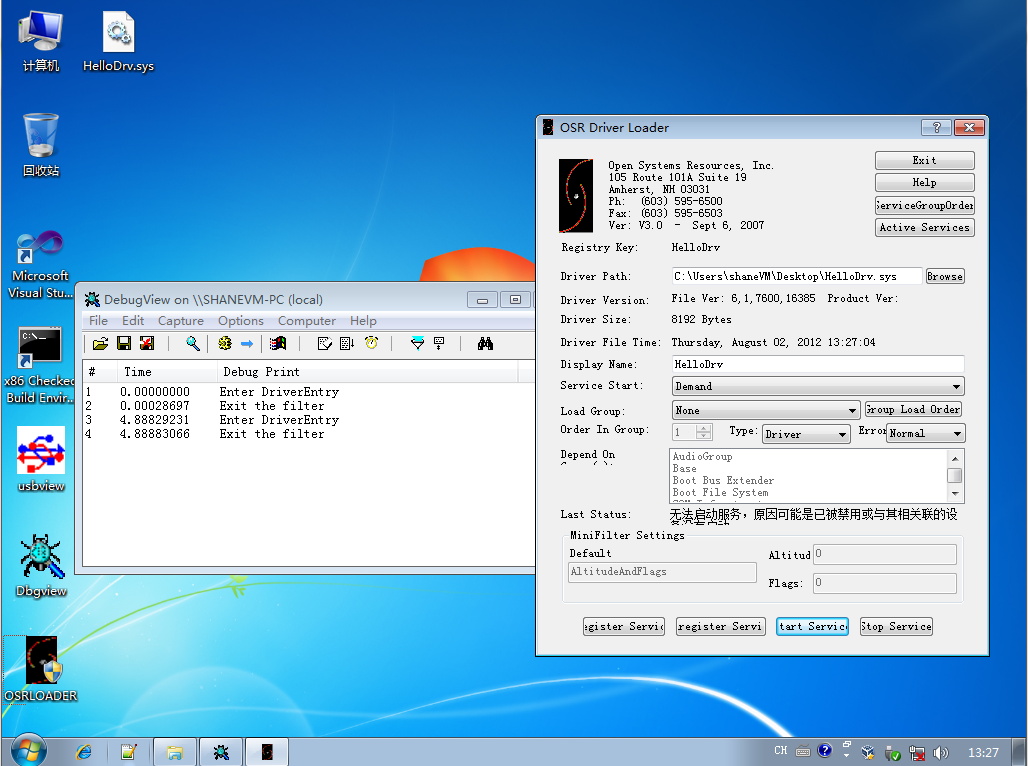


可以看到，最终成功的调用了AddDevice方法，但是问题是，目前还没有侦测到USB设备加入的时候。

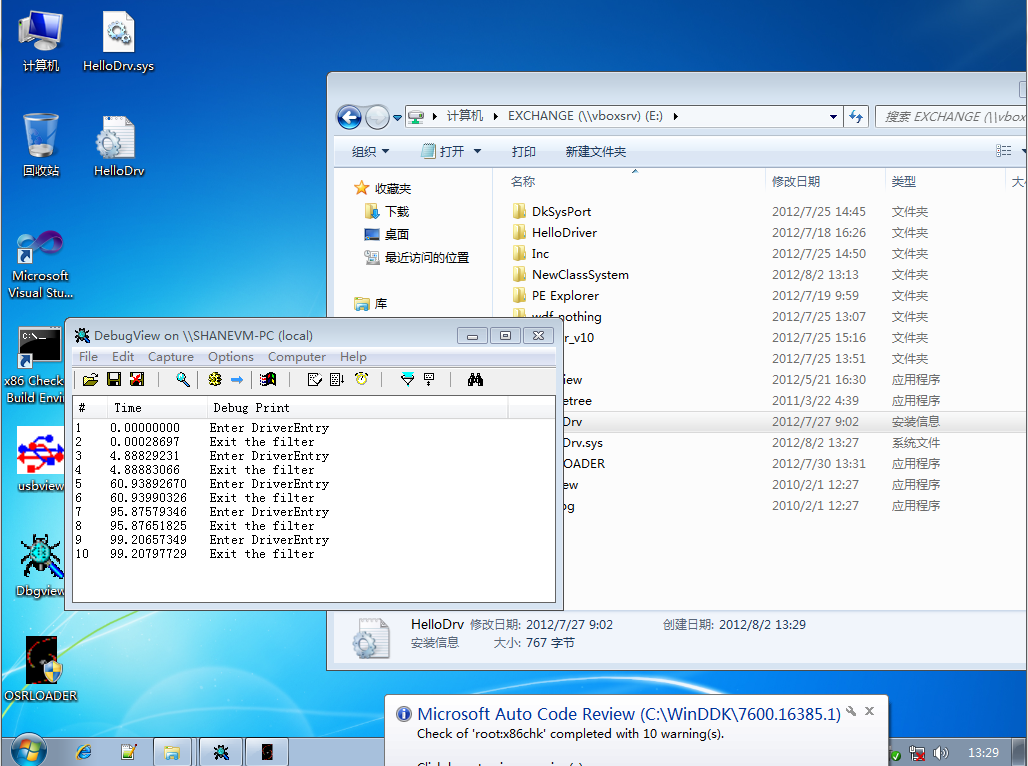


最后居然还调用了退出例程。

需要注意的是上面根本没有调用AddDevice例程，而是Debug输出的位置写错了。现在更改了一些信息之后可以看到



直接进入例程之后就直接退出了。



之后使用inf安装的方法，结果也一致。