配置好驱动测试环境后，就可以正式编写驱动了。市面上讲解驱动开发的书籍汗牛充栋，但讲得较为太复杂，让初学者不好理解。本文从一个简单的 hello,world 驱动（驱动模板）讲起，力求讲解得简单明了，让大家好理解。

本文主角：

1. DbgView。DbgView 是查看程序调试输出的工具，由美国高富帅 Mark Russinovich 编写（不得不说，此人长得帅，编程技术又牛，让多少男人羡慕妒忌，让多少女人一见倾心）。下载地址：http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx

2. KmdMgr。KmdMgr 是一个由俄国人编写的驱动加载工具。比起国内那些乱七八糟的驱动加载工具，它的特点是可以与驱动进行通信（虽然无法设置 I/O 缓冲区）。下载地址：https://www.assembla.com/code/L2h/subversion/nodes/LowLevel/KmdManager.exe?_format=raw&rev=1

3. WIN64AST。作者自行开发的 64 位 ARK 类工具。在本章中用来查看驱动是否加载成功。在后续章节还有其他的用途。下载地址：www.win64ast.com。

4. WIN64UDL。作者自行开发的驱动加载工具，能在正常模式下加载没有签名的驱动。因为这个工具，被人举报滥用签名，最终导致价值 15000 人民币的数字签名被吊销。下载地址：http://www.m5home.com/bbs/thread-7845-1-1.html

编写驱动：

以下是一个我写的 WIN64 驱动模板（代码中已经加了详细的注释，完整工程文件可以在论坛上下载）：

```
//【0】包含的头文件，可以加入系统或自己定义的头文件
#include <ntddk.h>
#include <windef.h>
#include <stdlib.h>

//【1】定义符号链接，一般来说修改为驱动的名字即可
#define DEVICE_NAME          L"\\Device\\KrnlHW64"
#define LINK_NAME            L"\\DosDevices\\KrnlHW64"
#define LINK_GLOBAL_NAME     L"\\DosDevices\\Global\\KrnlHW64"

//【2】定义驱动功能号和名字，提供接口给应用程序调用
#define IOCTL_IO_TEST        CTL_CODE(FILE_DEVICE_UNKNOWN, 0x800,
METHOD_BUFFERED, FILE_ANY_ACCESS)
#define IOCTL_SAY_HELLO      CTL_CODE(FILE_DEVICE_UNKNOWN, 0x801,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//【3】驱动卸载的处理例程
VOID DriverUnload(PDRIVER_OBJECT pDriverObj)
{
    UNICODE_STRING strLink;
    DbgPrint("[KrnlHW64]DriverUnload\n");
    RtlInitUnicodeString(&strLink, LINK_NAME);
    IoDeleteSymbolicLink(&strLink);
```

```
    IoDeleteDevice(pDriverObj->DeviceObject);
}

//【4】IRP_MJ_CREATE对应的处理例程，一般不用管它
NTSTATUS DispatchCreate(PDEVICE_OBJECT pDevObj, PIRP pIrp)
{
    DbgPrint("[KrnlHW64]DispatchCreate\n");
    pIrp->IoStatus.Status = STATUS_SUCCESS;
    pIrp->IoStatus.Information = 0;
    IoCompleteRequest(pIrp, IO_NO_INCREMENT);
    return STATUS_SUCCESS;
}

//【5】IRP_MJ_CLOSE对应的处理例程，一般不用管它
NTSTATUS DispatchClose(PDEVICE_OBJECT pDevObj, PIRP pIrp)
{
    DbgPrint("[KrnlHW64]DispatchClose\n");
    pIrp->IoStatus.Status = STATUS_SUCCESS;
    pIrp->IoStatus.Information = 0;
    IoCompleteRequest(pIrp, IO_NO_INCREMENT);
    return STATUS_SUCCESS;
}

//【6】IRP_MJ_DEVICE_CONTROL对应的处理例程，驱动最重要的函数之一，一般走正常途径调用驱动功能的程序，都会经过这个函数
NTSTATUS DispatchIoctl(PDEVICE_OBJECT pDevObj, PIRP pIrp)
{
    NTSTATUS status = STATUS_INVALID_DEVICE_REQUEST;
    PIO_STACK_LOCATION pIrpStack;
    ULONG uIoControlCode;
    PVOID pIoBuffer;
    ULONG uInSize;
    ULONG uOutSize;
    DbgPrint("[KrnlHW64]DispatchIoctl\n");
    pIrpStack = IoGetCurrentIrpStackLocation(pIrp);
    //控制码
    uIoControlCode = pIrpStack->Parameters.DeviceIoControl.IoControlCode;
    //输入输出缓冲区
    pIoBuffer = pIrp->AssociatedIrp.SystemBuffer;
    //输入区域大小
    uInSize = pIrpStack->Parameters.DeviceIoControl.InputBufferLength;
    //输出区域大小
    uOutSize = pIrpStack->Parameters.DeviceIoControl.OutputBufferLength;
    switch(uIoControlCode)
```

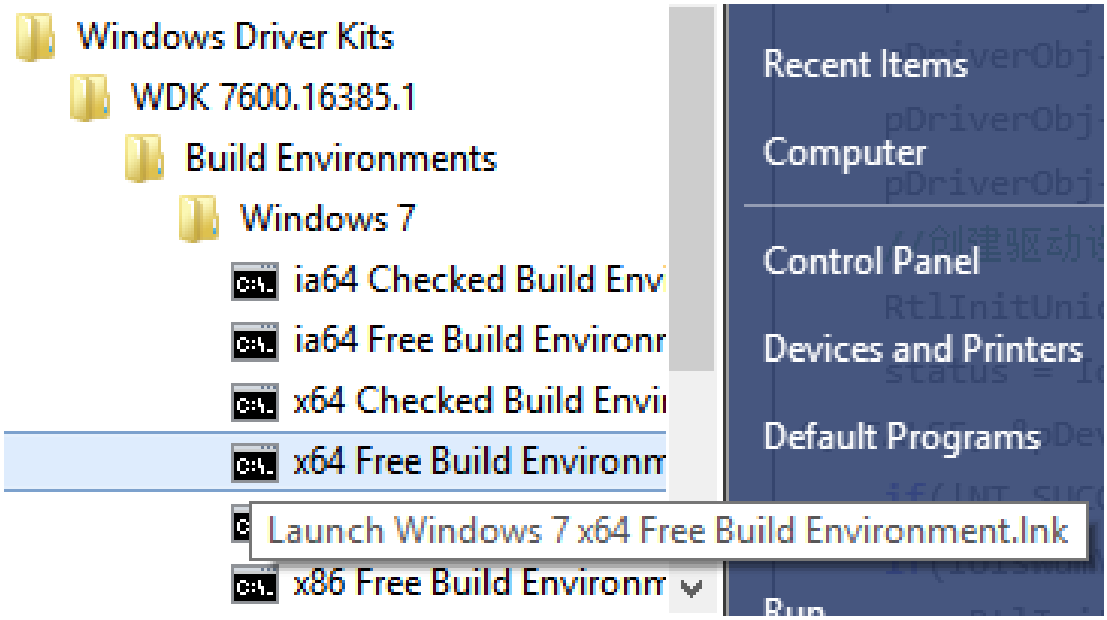
```
    {
        //在这里加入接口
        case IOCTL_IO_TEST:
        {
            DWORD dw=0;
            //获得输入的内容
            memcpy(&dw,pIoBuffer,sizeof(DWORD));
            //使用输入的内容
            dw++;
            //输出处理的结果
            memcpy(pIoBuffer,&dw,sizeof(DWORD));
            //处理成功，返回非STATUS_SUCCESS会让DeviveIoControl返回失败
            status = STATUS_SUCCESS;
            break;
        }
        case IOCTL_SAY_HELLO:
        {
            DbgPrint("[KrnlHW64]IOCTL_SAY_HELLO\n");
            status = STATUS_SUCCESS;
            break;
        }
    }
    if(status == STATUS_SUCCESS)
        pIrp->IoStatus.Information = uOutSize;
    else
        pIrp->IoStatus.Information = 0;
    pIrp->IoStatus.Status = status;
    IoCompleteRequest(pIrp, IO_NO_INCREMENT);
    return status;
}

//【7】驱动加载的处理例程，里面进行了驱动的初始化工作
NTSTATUS DriverEntry(PDRIVER_OBJECT pDriverObj, PUNICODE_STRING
pRegistryString)
{
    NTSTATUS status = STATUS_SUCCESS;
    UNICODE_STRING ustrLinkName;
    UNICODE_STRING ustrDevName;
    PDEVICE_OBJECT pDevObj;
    //初始化驱动例程
    pDriverObj->MajorFunction[IRP_MJ_CREATE] = DispatchCreate;
    pDriverObj->MajorFunction[IRP_MJ_CLOSE] = DispatchClose;
    pDriverObj->MajorFunction[IRP_MJ_DEVICE_CONTROL] = DispatchIoctl;
    pDriverObj->DriverUnload = DriverUnload;
```

```
    //创建驱动设备
    RtlInitUnicodeString(&ustrDevName, DEVICE_NAME);
    status = IoCreateDevice(pDriverObj, 0, &ustrDevName, FILE_DEVICE_UNKNOWN,
0, FALSE, &pDevObj);
    if(!NT_SUCCESS(status)) return status;
    if(IoIsWdmVersionAvailable(1, 0x10))
        RtlInitUnicodeString(&ustrLinkName, LINK_GLOBAL_NAME);
    else
        RtlInitUnicodeString(&ustrLinkName, LINK_NAME);
    //创建符号链接
    status = IoCreateSymbolicLink(&ustrLinkName, &ustrDevName);
    if(!NT_SUCCESS(status))
    {
        IoDeleteDevice(pDevObj);
        return status;
    }
    //走到这里驱动实际上已经初始化完成，下面添加的是功能初始化的代码
    DbgPrint("[KrnlHW64]DriverEntry\n");
    return STATUS_SUCCESS;
}
```

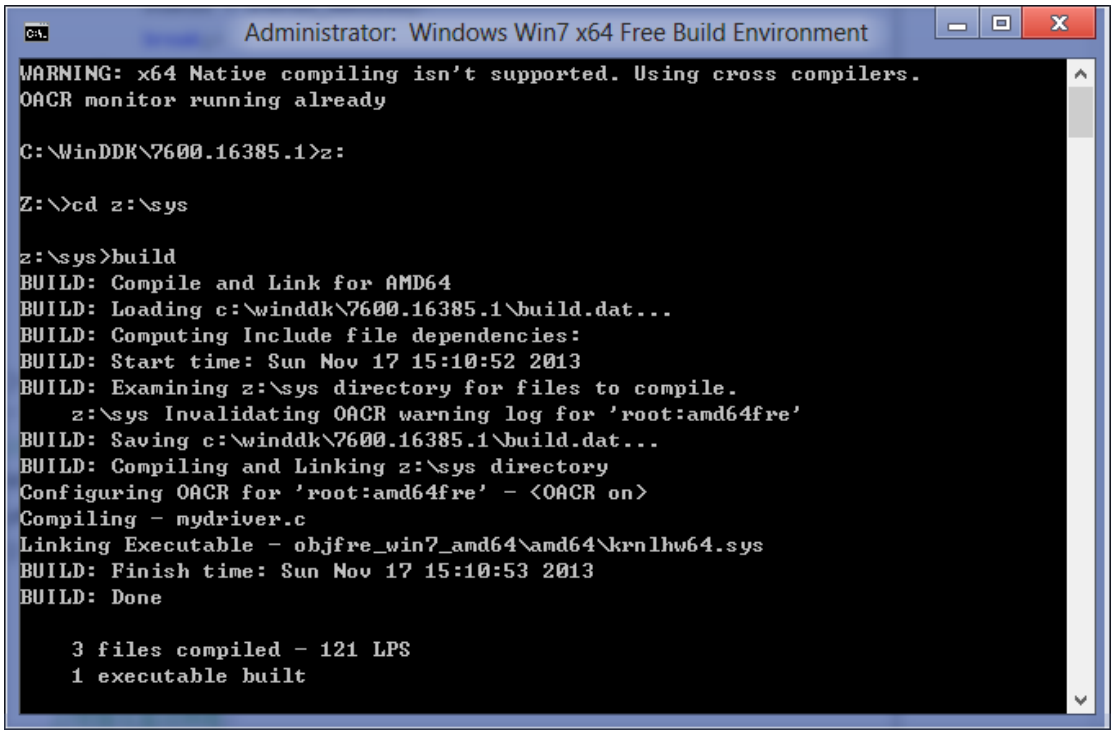
如果你懒得认真看完上面的代码，也没问题，我总结几句：1.DriverEntry 就是驱动的 main 函数，驱动加载后会从 DriverEntry 开始执行。2.驱动类似 DLL，可以提供接口给应用程序调用，不过以导出函数的方式，而是用一套专门的通信函数 DeviceIoControl。关于应用程序与驱动程序通信，后面会讲。

编译驱动：

1.打开『x64 Free Build Environment』：



2.切换到源码目录（假设源码目录是：z:\sys），并输入 BUILD 编译：



3.如果看到『1 executable built』字眼，则证明编译成功。

4.驱动的编译跟目录下的 source 文件有关系，比如本例中，它的内容如下（注意不要手贱把空行去掉了，否则可能会导致无法编译）：

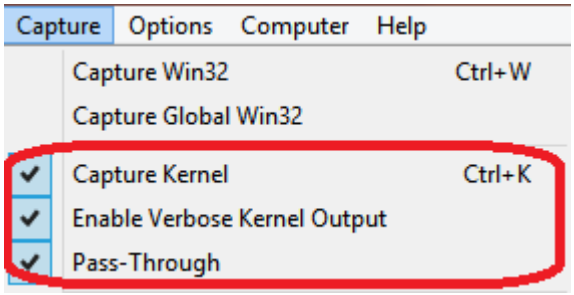
```
TARGETNAME=KrnlHW64     <-驱动的文件的名称，一般来说修改这个就行了
TARGETTYPE=DRIVER     <-编译的类型
TARGETPATH=obj

INCLUDES=.\

SOURCES = MyDriver.c     <-多个 C 文件时，把所有 C 文件的名称分成多行写
```

测试驱动前的准备：

1.以管理员权限运行 DBGVIEW。

2.把 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Debug Print Filter 的 Default 值修改为 ffffffff

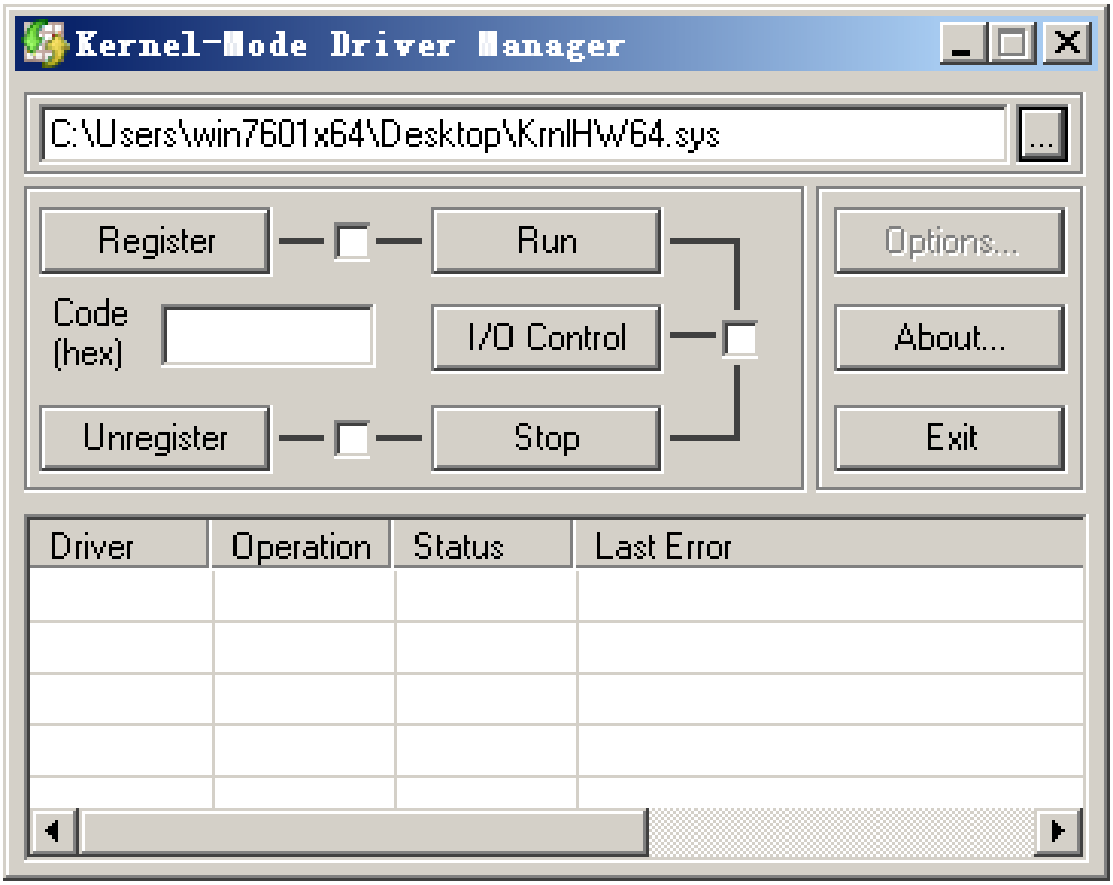
3.打开 DBGVIEW 并把以下选项全部勾上：



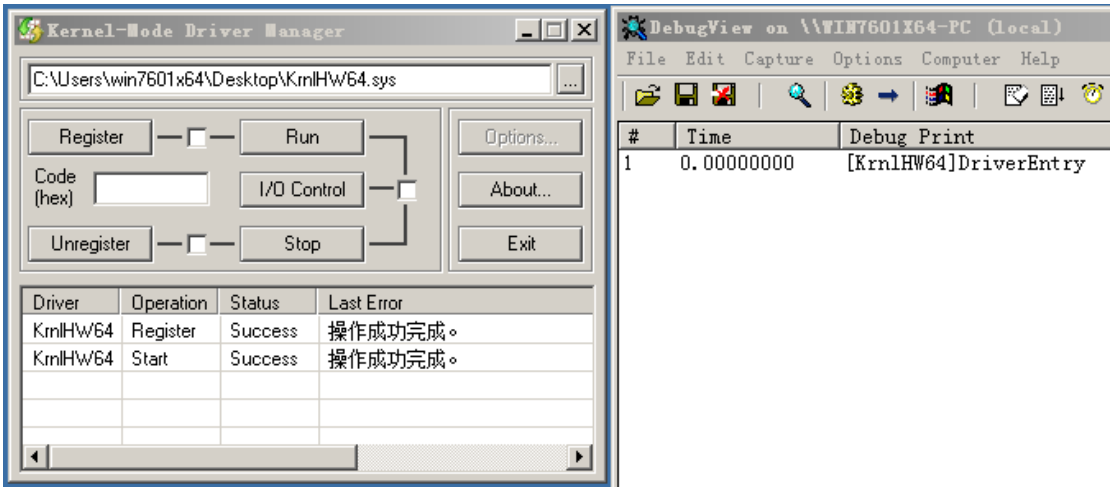
标准的驱动测试方法：

1.打开虚拟机，进入双机调试的环境（忘记了就参考上节课的内容）。

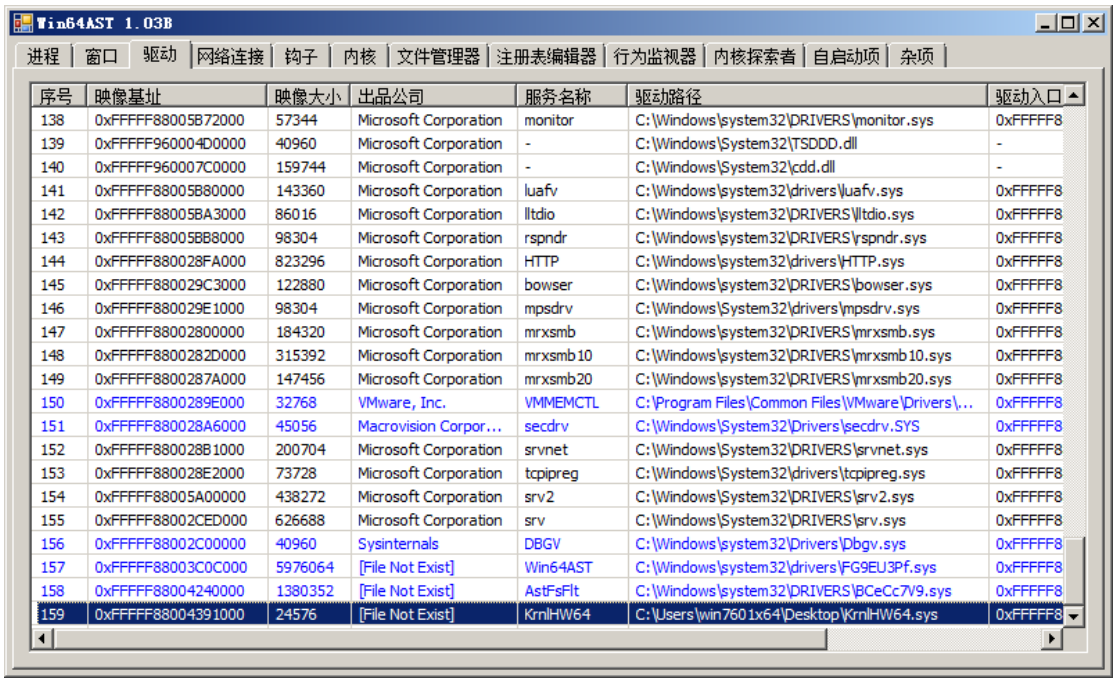
2.运行 KmdMgr.exe，把 SYS 拖动到文本框里。



3.点击“Register”和“Run”按钮，看看输出是否提示成功。如果成功会有类似的输出：

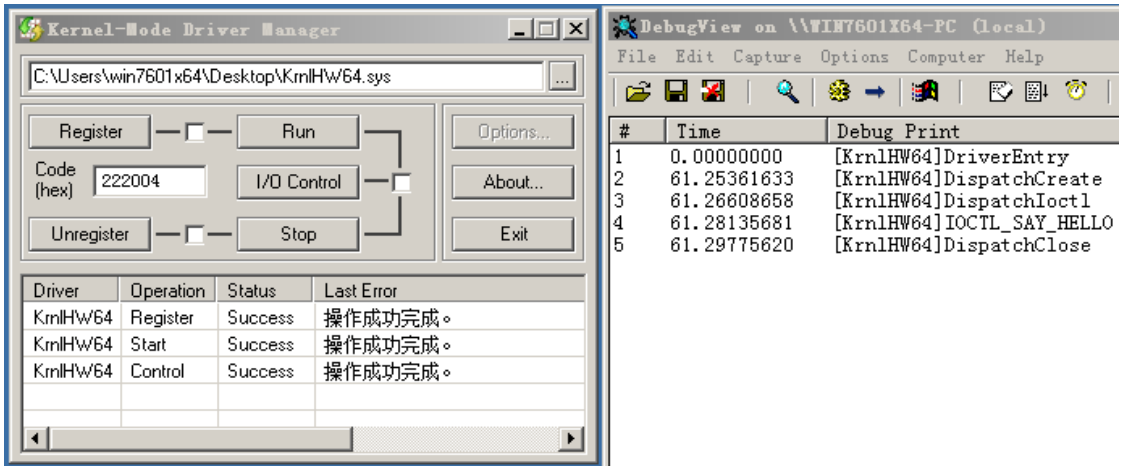


4.运行 WIN64AST，点击内核模块，查看驱动是否已经存在于内核里了：

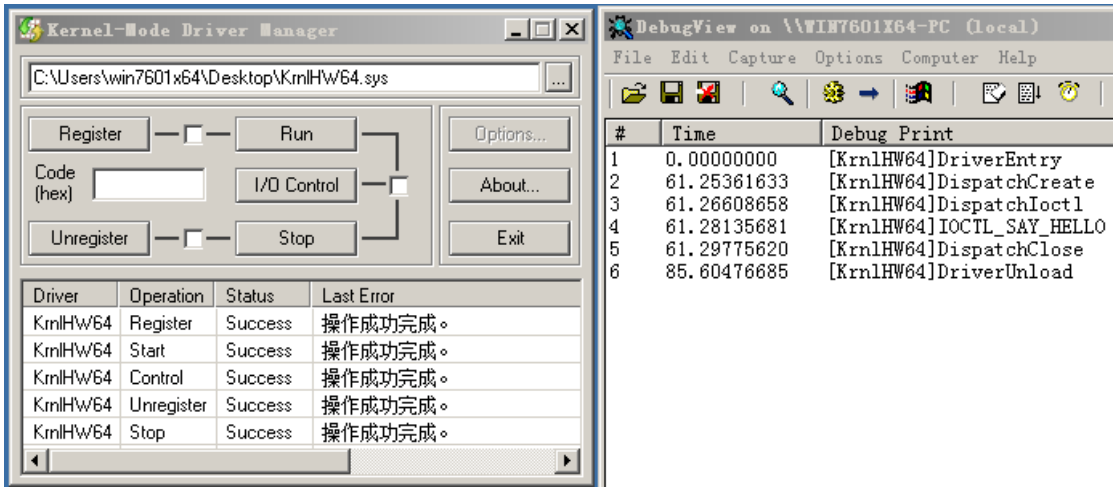


序号	映像基址	映像大小	出品公司	服务名称	驱动路径	驱动入口
138	0xFFFFF88005B72000	57344	Microsoft Corporation	monitor	C:\Windows\system32\DRIVERS\monitor.sys	0xFFFFF8
139	0xFFFFF960004D0000	40960	Microsoft Corporation	-	C:\Windows\System32\TSDDD.dll	-
140	0xFFFFF960007C0000	159744	Microsoft Corporation	-	C:\Windows\System32\cdd.dll	-
141	0xFFFFF88005B80000	143360	Microsoft Corporation	luafv	C:\Windows\system32\drivers\luafv.sys	0xFFFFF8
142	0xFFFFF88005BA3000	86016	Microsoft Corporation	lltdio	C:\Windows\system32\DRIVERS\lltdio.sys	0xFFFFF8
143	0xFFFFF88005BB8000	98304	Microsoft Corporation	rspndr	C:\Windows\system32\DRIVERS\rspndr.sys	0xFFFFF8
144	0xFFFFF880028FA000	823296	Microsoft Corporation	HTTP	C:\Windows\system32\drivers\HTTP.sys	0xFFFFF8
145	0xFFFFF880029C3000	122880	Microsoft Corporation	bowser	C:\Windows\system32\DRIVERS\bowser.sys	0xFFFFF8
146	0xFFFFF880029E1000	98304	Microsoft Corporation	mpsdrv	C:\Windows\System32\drivers\mpsdrv.sys	0xFFFFF8
147	0xFFFFF88002800000	184320	Microsoft Corporation	mrxsmb	C:\Windows\system32\DRIVERS\mrxsmb.sys	0xFFFFF8
148	0xFFFFF8800282D000	315392	Microsoft Corporation	mrxsmb10	C:\Windows\system32\DRIVERS\mrxsmb10.sys	0xFFFFF8
149	0xFFFFF8800287A000	147456	Microsoft Corporation	mrxsmb20	C:\Windows\system32\DRIVERS\mrxsmb20.sys	0xFFFFF8
150	0xFFFFF8800289E000	32768	VMware, Inc.	VMMEMCTL	C:\Program Files\Common Files\VMware\Drivers\...	0xFFFFF8
151	0xFFFFF880028A6000	45056	Macrovision Corpor...	secdrv	C:\Windows\System32\Drivers\secdrv.SYS	0xFFFFF8
152	0xFFFFF880028B1000	200704	Microsoft Corporation	srvnet	C:\Windows\System32\DRIVERS\srvnet.sys	0xFFFFF8
153	0xFFFFF880028E2000	73728	Microsoft Corporation	tcpipreg	C:\Windows\System32\drivers\tcpipreg.sys	0xFFFFF8
154	0xFFFFF88005A00000	438272	Microsoft Corporation	srv2	C:\Windows\System32\DRIVERS\srv2.sys	0xFFFFF8
155	0xFFFFF88002CED000	626688	Microsoft Corporation	srv	C:\Windows\System32\DRIVERS\srv.sys	0xFFFFF8
156	0xFFFFF88002C00000	40960	Sysinternals	DBGV	C:\Windows\system32\Drivers\Dbgv.sys	0xFFFFF8
157	0xFFFFF88003C0C000	5976064	[File Not Exist]	Win64AST	C:\Windows\system32\drivers\FG9EU3Pf.sys	0xFFFFF8
158	0xFFFFF88004240000	1380352	[File Not Exist]	AstFsFlt	C:\Windows\system32\DRIVERS\BCeCc7V9.sys	0xFFFFF8
159	0xFFFFF88004391000	24576	[File Not Exist]	KrnlHW64	C:\Users\win7601x64\Desktop\KrnlHW64.sys	0xFFFFF8

5. 在 CODE 处输入 222004（为什么是 222004 而不是 801？这个后面会讲到，这里先卖一个关子。但这个数值可以使用 calc_ctl_code.exe 算出来，既输入 801，可以输出 222004），点击“I/O Control”按钮，如果成功会有类似的输出：



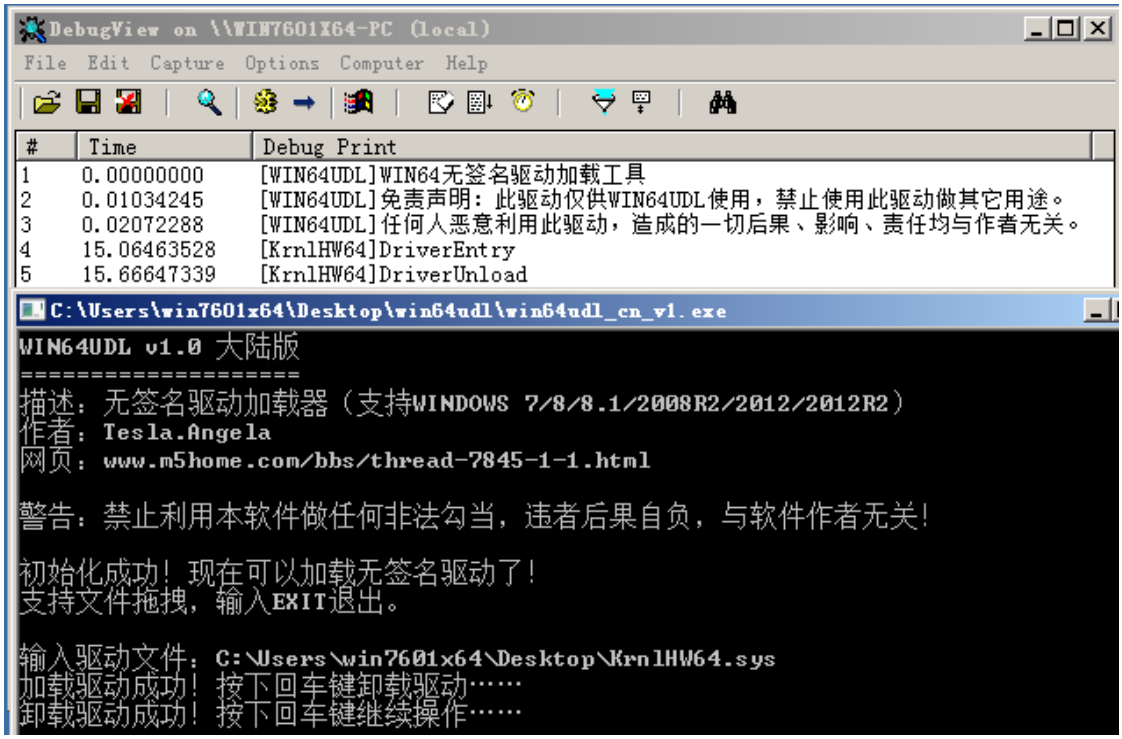
6. 点击“Unregister”和“Stop”按钮，如果成功会有类似的输出：



很显然，用标准方法测试一个驱动是很麻烦且很耗时的。双机调试非常占用系统资源，虽然我的电脑配置较好（2600K+16GB 内存），但是在操作虚拟机时，仍然感到了明显的卡顿。下面介绍一种用特殊工具测试驱动的方法，无需双机调试，甚至无需使用虚拟机。

用 WIN64UDL 测试驱动：

1. 运行 WIN64UDL。
2. 把驱动文件拖进 WIN64UDL 里，然后按下 ENTER 加载。
3. 再按一次 ENTER 卸载驱动。

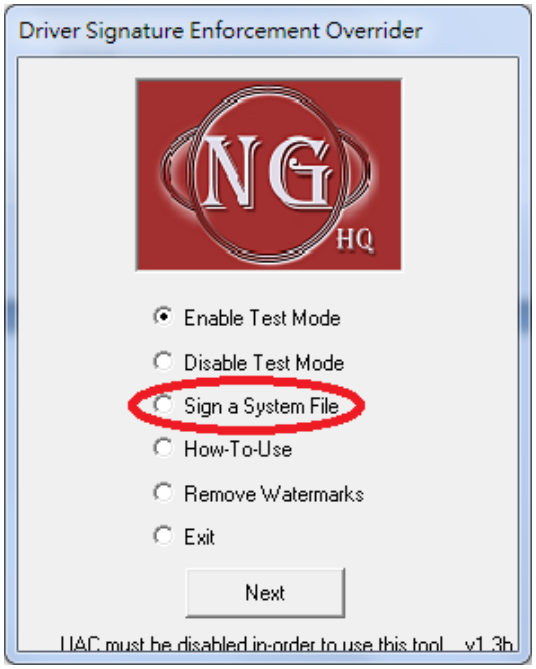


最后，再补充一种非常麻烦的方法，此方法也算是标准方法之一，适用于没有虚拟机或无法进行双机调试的时候。由于非常麻烦，所以不推荐使用。说实话，谁用此方法测试驱动，绝对是脑门被驴踢了。

1. 开启测试模式。管理员权限运行 CMD，输入：bcdedit -set testsigning on。

2. 重启计算机

3. 用 dseo13b（下载地址：http://files.ngohq.com/ngo/dseo/dseo13b.exe）给驱动程序添加测试签名。方法很简单，运行 deso13b，一路 NEXT，当出现这个对话框时，选择“Sign a System File”再点 NEXT：



4. 用任意工具加载驱动。