

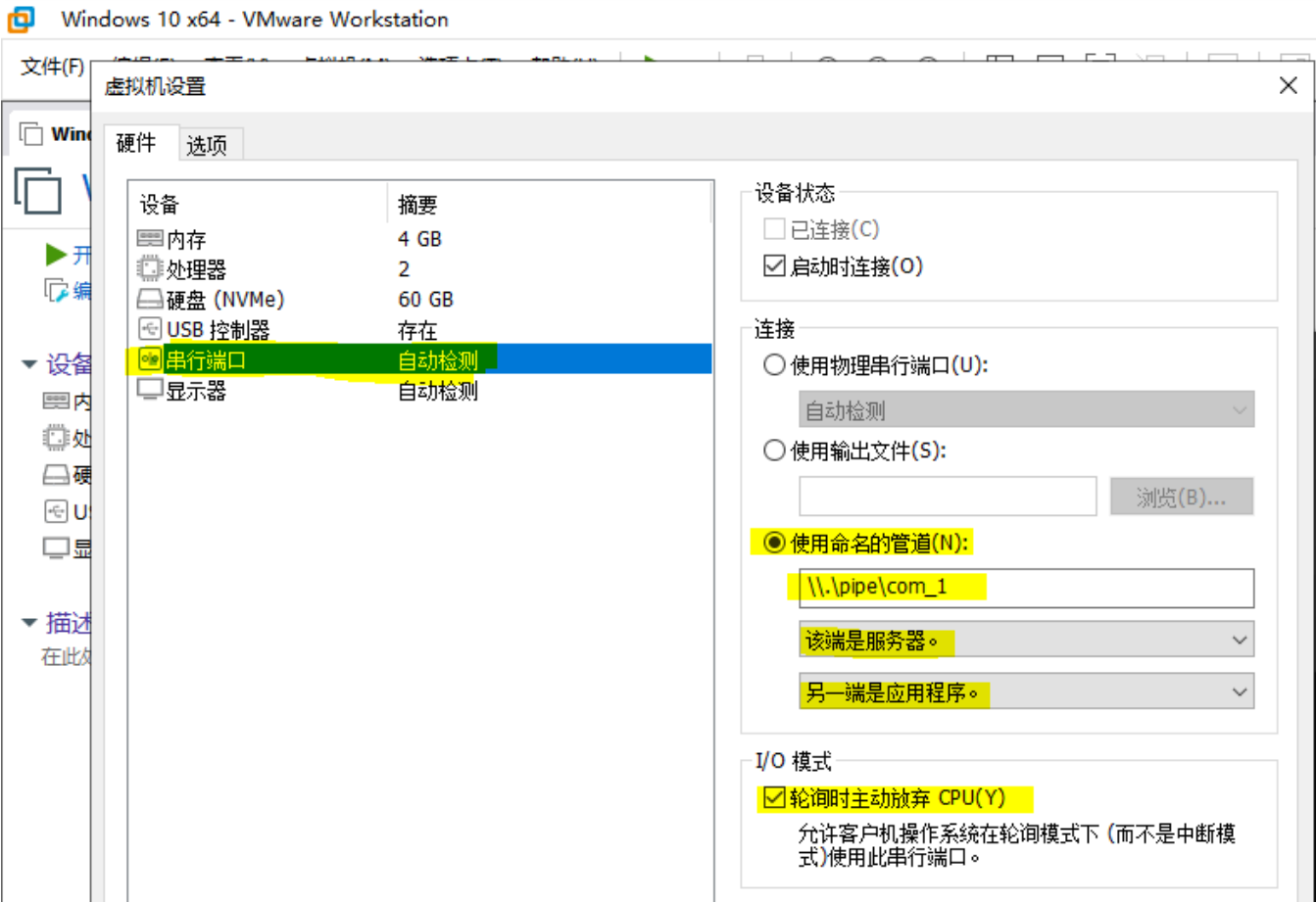
WinDBG是在 windows 平台下，强大的用户态和内核态调试工具，相比较于 visualstudio 它是一个轻量级的调试工具，所谓轻量级指的是它的安装文件大小较小，但是其调试功能却比VS更为强大，WinDBG由于是微软的产品所以能够调试 windows 系统的内核，另外一个用途是可以用来分析 dump 数据，本笔记用于记录如何开启 windows 系统内核调试功能，并使用 WinDBG 调试驱动。

在驱动开发中，需要进行双机调试以确保驱动程序能够正确地与系统交互。为了实现双机调试，您需要在两台计算机之间建立调试连接，并在 visual studio 中配置相应的调试选项。在调试时，您可以使用 Visual Studio 提供的调试工具进行单步调试和调试信息的查看，以便更快地定位问题和解决错误。

1.首先需要在 vmware 虚拟机关闭状态下添加一个 管道虚拟串口，此处需要删除打印机，否则串口之间冲突。

操作步骤： 编辑虚拟机设置 -> 添加 -> 串行端口 -> 完成

参数配置： 使用命名管道 -> \\.\pipe\com\_1 -> 该端是服务器，另一端是应用程序 -> 轮询时主动放弃CPU -> 确定



2.开启虚拟机中的 windows 系统，然后以管理员身份运行CMD命令行，输入 bcdedit 命令，可以查看到系统的当前启动项，如果是新的系统，则只会有 {current} 启动项以及一个 {bootmgr} 项。

管理员: cmd

```
C:\Windows\system32>
C:\Windows\system32>bcdedit

Windows 启动管理器
-----
标识符                {bootmgr}
device                partition=\Device\HarddiskVolume2
path                  \EFI\Microsoft\Boot\bootmgfw.efi
description            Windows Boot Manager
locale                zh-CN
inherit                {globalsettings}
default                {current}
resumeobject           {bdb0b3b2-3f21-11ed-9931-d46011246f28}
displayorder           {current}
toolsdisplayorder      {memdiag}
timeout                30

Windows 启动加载器
-----
标识符                {current}
device                partition=C:
path                  \Windows\system32\winload.efi
description            Windows 10
locale                zh-CN
inherit                {bootloadersettings}
recoverysequence       {bdb0b3b4-3f21-11ed-9931-d46011246f28}
displaymessageoverride Recovery
```

连续执行下方的七条命令，依次建立启动项，激活 windows 系统的调试模式，并开启串口通信，调试端口波特率为 115200

```
bcdedit /set testsigning on
bcdedit -debug on
bcdedit /bootdebug on
bcdedit /set "{current}" bootmenupolicy Legacy           // 修改启动方式为Legacy
bcdedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:115200 // 设置串口1为调试端口波特率为115200
bcdedit /copy "{current}" /d "Debug"                     // 将当前配置复制到Debug启动配置
bcdedit /debug "{<新建的启动配置的标识符>}" on          // 打开调试开关
```

但需要注意 {<新建的启动配置的标识符>} 需替换成 {bdb0b3b6-3f21-11ed-9931-d46011246f28} 标志，如下所示。

管理员: cmd

```
C:\Windows\system32>bcdedit /set testsigning on
操作成功完成。

C:\Windows\system32>bcdedit -debug on
操作成功完成。

C:\Windows\system32>bcdedit /bootdebug on
操作成功完成。

C:\Windows\system32>bcdedit /set "{current}" bootmenupolicy Legacy
操作成功完成。

C:\Windows\system32>bcdedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:115200
操作成功完成。

C:\Windows\system32>bcdedit /copy "{current}" /d "Debug"
已将该项成功复制到 {bdb0b3b6-3f21-11ed-9931-d46011246f28}。

C:\Windows\system32>bcdedit /debug "{bdb0b3b6-3f21-11ed-9931-d46011246f28}" on
操作成功完成。

C:\Windows\system32>
```

3.最后查看一下当前调试配置选项，执行命令 `bcdedit /dbgsettings`，显示出使用的第一个串口，波特率为 115200bps，保持默认不需要修改。

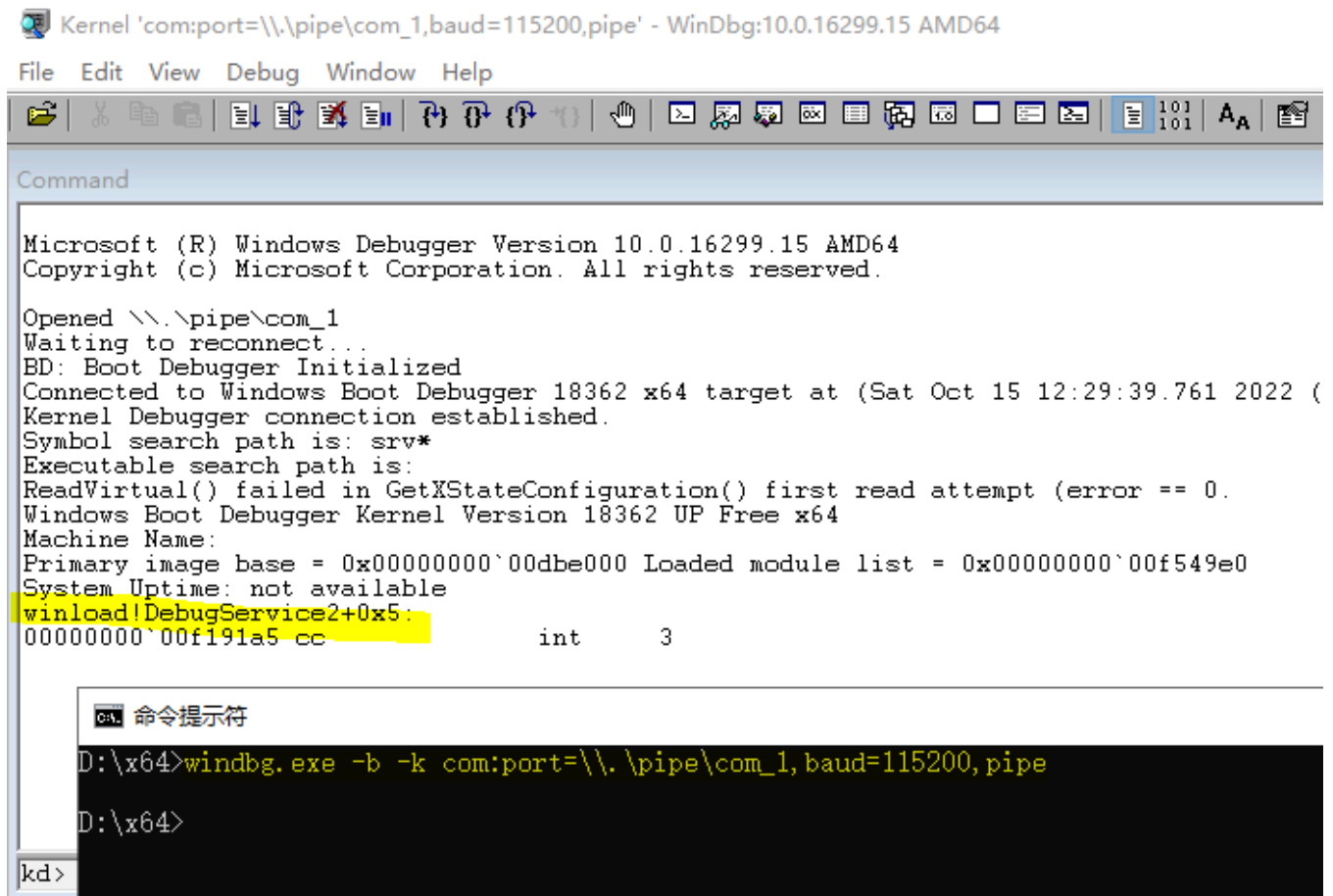
```
管理员: cmd
C:\Windows\system32>bcdedit /dbgsettings
debugtype             Serial
debugport             1
baudrate              115200
操作成功完成。
C:\Windows\system32>
```

4.配置完成后，重新启动系统，在开机的时候选择 Windows10 [启用调试程序] 则系统会黑屏，说明已经正常进入调试模式了。



5.此时回到物理机上面，解压缩课件中的 WinDBG\_10.0.16299.15.zip 到D盘根目录下，我们在命令行中切换到 WinDBG\X64 的根目录下，并执行以下命令，即可连接虚拟机串口进行调试了。

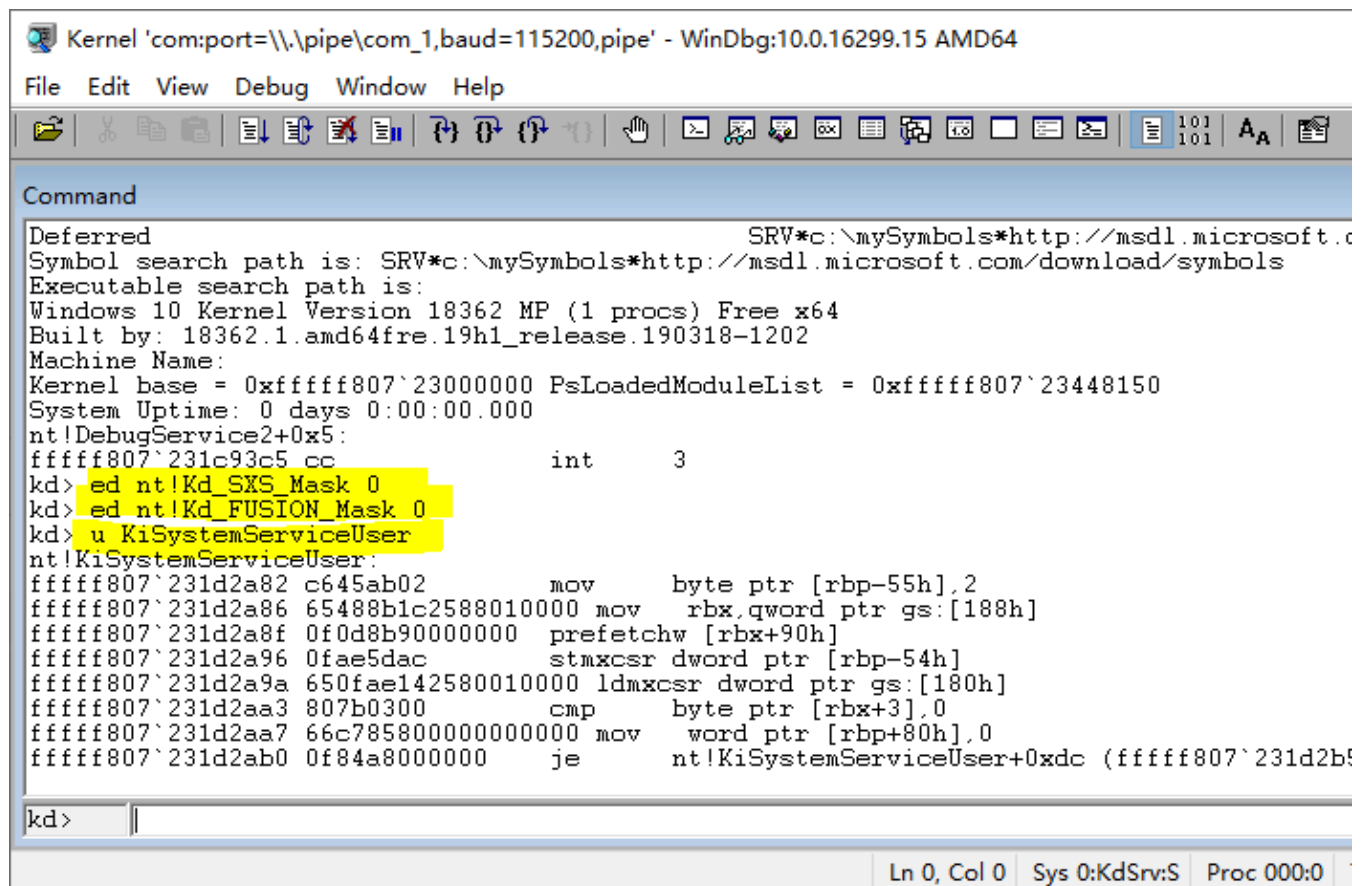
- 执行命令 `windbg.exe -b -k com:port=\\.\pipe\com_1,baud=115200,pipe` 如下图



6.至此我们还需要加载符号，符号的作用是方便我们调试，该符号是由微软官方维护的权威资料，在命令行下依次执行以下命令，配置好符号加载并启动系统。

```
kd> .sympath SRV*c:\mySymbols*http://msdl.microsoft.com/download/symbols
kd> .reload
kd> g
kd> g
kd> ed nt!Kd_SXS_Mask 0
kd> ed nt!Kd_FUSION_Mask 0
kd> u KiSystemServiceUser
```

这样即可完成配置操作，此时系统已被断下等待我们执行操作，如下图所示。



The screenshot shows the WinDbg interface with the title bar 'Kernel 'com:port=\\.\pipe\com\_1,baud=115200,pipe' - WinDbg:10.0.16299.15 AMD64'. The menu bar includes File, Edit, View, Debug, Window, and Help. The toolbar contains various icons for file operations, debugging, and viewing. The Command window displays the following text:

```
Deferred SRV*c:\mySymbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: SRV*c:\mySymbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 10 Kernel Version 18362 MP (1 procs) Free x64
Built by: 18362.1.amd64fre.19h1_release.190318-1202
Machine Name:
Kernel base = 0xfffff807`23000000 PsLoadedModuleList = 0xfffff807`23448150
System Uptime: 0 days 0:00:00.000
nt!DebugService2+0x5:
fffff807`231c93c5 cc int 3
kd> ed nt!Kd_SXS_Mask 0
kd> ed nt!Kd_FUSION_Mask 0
kd> u KiSystemServiceUser
nt!KiSystemServiceUser:
fffff807`231d2a82 c645ab02 mov byte ptr [rbp-55h],2
fffff807`231d2a86 65488b1c2588010000 mov rbx,qword ptr gs:[188h]
fffff807`231d2a8f 0f0d8b9000000000 prefetchw [rbx+90h]
fffff807`231d2a96 0fae5dac stmxcsr dword ptr [rbp-54h]
fffff807`231d2a9a 650fae142580010000 ldmxcsr dword ptr gs:[180h]
fffff807`231d2aa3 807b0300 cmp byte ptr [rbx+3],0
fffff807`231d2aa7 66c78580000000000000 mov word ptr [rbp+80h],0
fffff807`231d2ab0 0f84a8000000 je nt!KiSystemServiceUser+0xdc (fffff807`231d2b0)
```

The command window shows the current command 'kd>' and the status bar at the bottom indicates 'Ln 0, Col 0 Sys 0:KdSrv:S Proc 000:0'.

7.最后我们配置测试一下调试功能，首先编写以下代码，代码中使用 DbgBreakPoint() 设置断点，将会在入口处中断。

```
#include <ntifs.h>

// 驱动默认回调
NTSTATUS DriverDefaultHandle(PDEVICE_OBJECT pDevObj, PIRP pIrp)
{
    NTSTATUS status = STATUS_SUCCESS;
    pIrp->IoStatus.Status = status;
    pIrp->IoStatus.Information = 0;
    IoCompleteRequest(pIrp, IO_NO_INCREMENT);

    return status;
}
```

```

// 驱动卸载函数
VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动已卸载 \n");
}

// 驱动入口地址
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    // 初始化默认派遣函数
    NTSTATUS status = STATUS_SUCCESS;
    for (ULONG i = 0; i < IRP_MJ_MAXIMUM_FUNCTION; i++)
    {
        Driver->MajorFunction[i] = DriverDefaultHandle;
    }

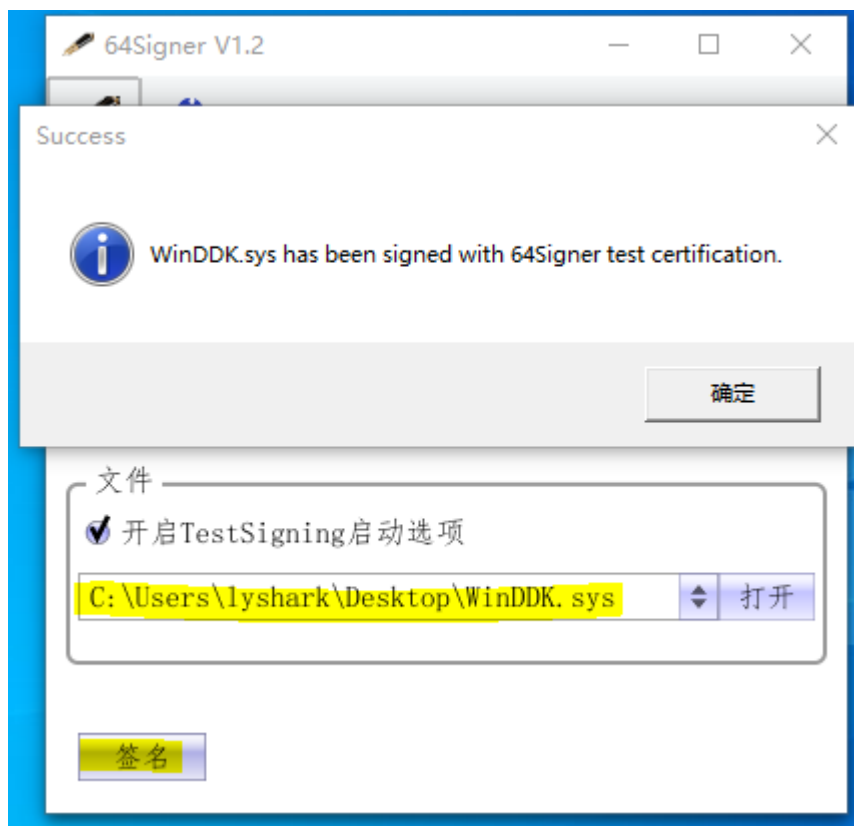
    // 设置断点
    DbgBreakPoint();
    // KdBreakPoint();
    // __debugbreak();

    DbgPrint("驱动已加载 \n");

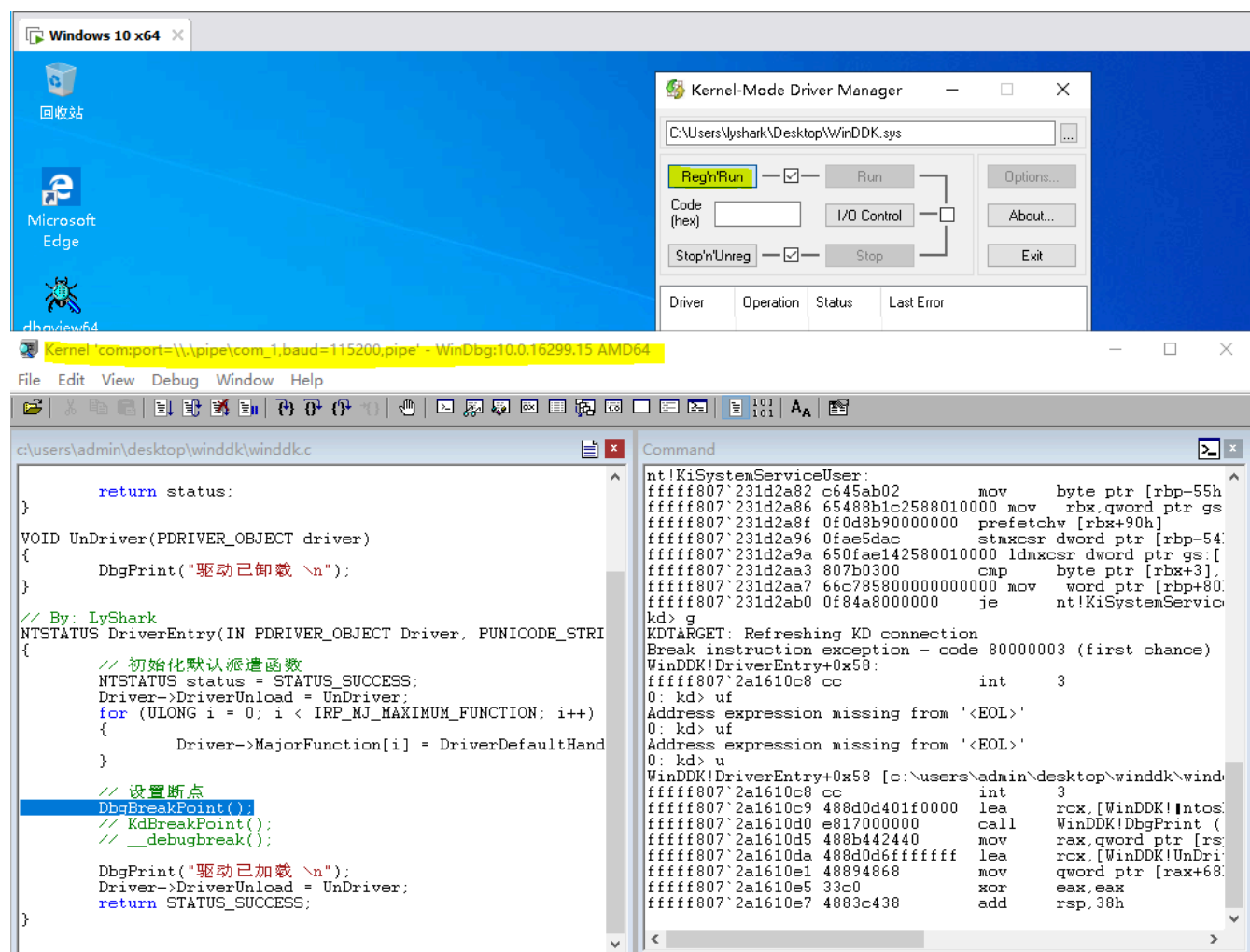
    // 驱动卸载函数
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}

```

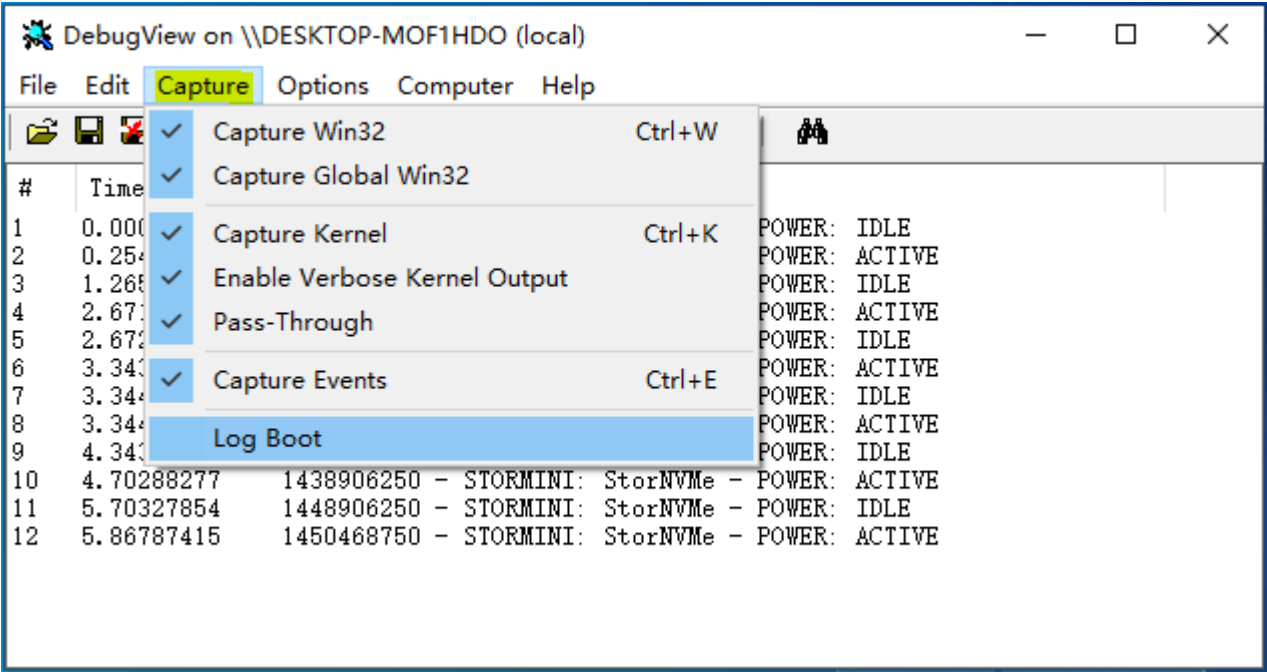
通过 Visual Studio 工具编译如上代码片段，并在 WinDBG 中输入 `g` 命令让系统运行起来，将编译好的驱动程序拖入到虚拟机中，并以管理员身份打开 `Windows 64Signer.exe`，使用该工具对驱动程序进行签名，如下图所示；



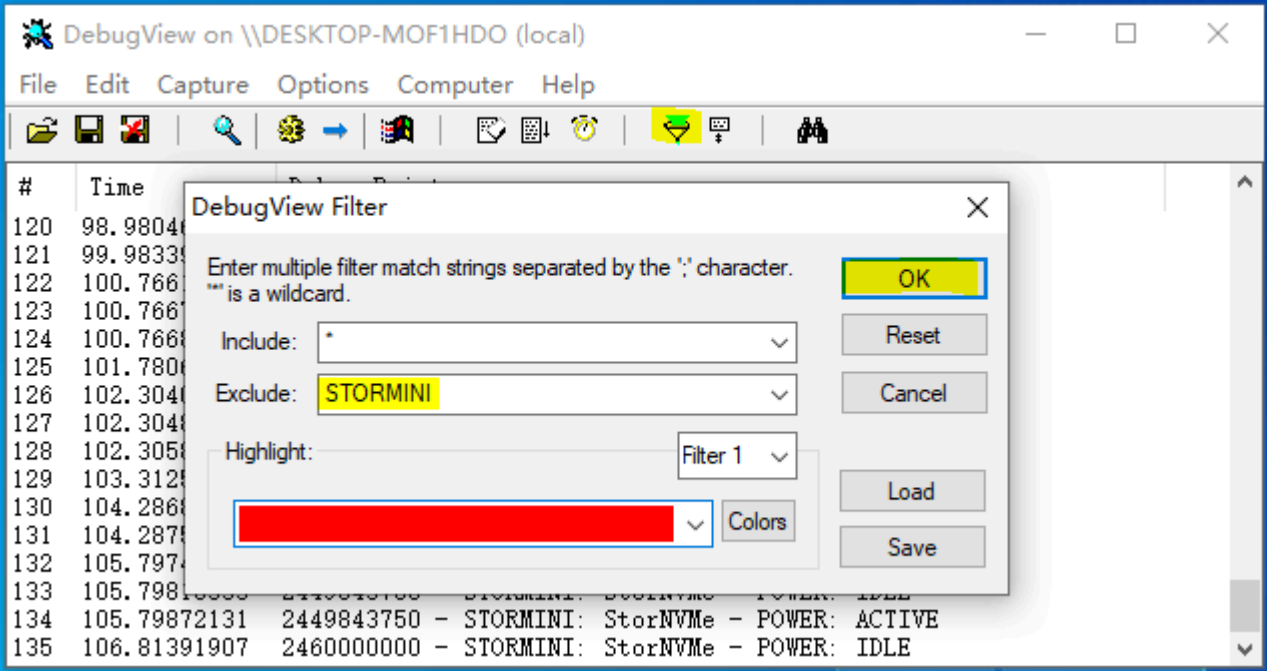
签名完成后将我们的驱动文件 `winDDK.sys`，拖入到 `KmdManager.exe` 驱动加载工具中，并通过驱动加载工具加载运行，此时 Windows 系统会卡死，回到 WinDBG 中发现已经可以进行调试了，如下图所示；



此处需要扩展一个知识点，如果不使用 winDBG 工具而想要获取到 DbgPrint() 函数输出结果，则你可以使用课件中提供的 dbgview64.exe 程序，不过此程序需要注意几点，该程序需要使用管理员身份运行，且运行后需要将 Capture 菜单中的属性全部打对勾，如下图所示；



此时 DebugView 会出现很多的无用输出，则你需要打开过滤器按钮，输入 STORMINI 将此类输出屏蔽掉，如下图所示；



至此再次使用 kmdManager 工具加载 winDDK 驱动，则可以无干扰的输出我们所需结果。