

2021/04/23_Windows32位内核_第3课_IDE 编译驱动程序

笔记本: Windows32位内核
创建时间: 2021/4/23 星期五 15:50
作者: ileemi

- [IDE 编译驱动程序](#)
- [调试操作系统](#)
 - [XP](#)
 - [配置调试器界面](#)
 - [调试系统](#)
 - [VS 中调试驱动程序](#)
 - [相关函数的使用](#)
 - [Win7](#)
 - [保持系统崩溃时的信息](#)
 - [调试系统崩溃文件](#)
- [内核API分类](#)

IDE 编译驱动程序

模型:

NT驱动模型

WDM模型 -- 即插即用

KMDF -- 驱动模型的框架由C++编写

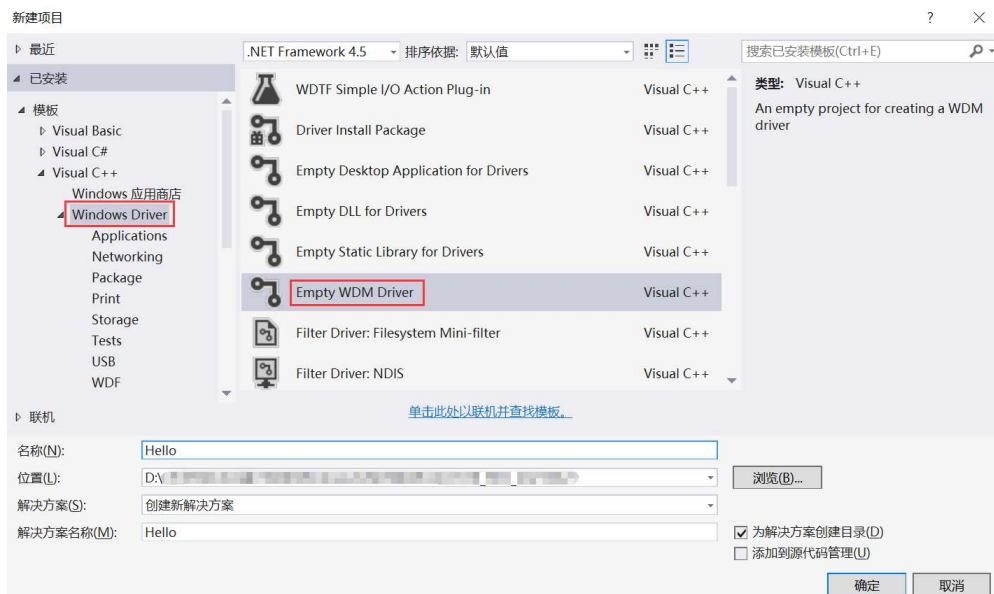
还有一些针对硬件设备的模型

框架模型: NDIS、Mini-filter等

内核驱动开发一般采用 "NT驱动模型", 也可采用框架模型。

使用高本版的VS配合对应版本的WDK编写驱动的步骤:

1. 打开Visual Studio 2013, 新建项目, 在 "Windows Drrver" 中选择 "Empty WDM Driver"后, 输入路径, 名称, 如下图所示:



2. 在空项目中新建 ".c" 或者 ".cpp" 文件, ".cpp" 文件防止函数名出现名称粉碎需要在头文件中使用 "extern "C" {}" 将头文件以及函数声明进行包裹。
3. 编写驱动程序时, 需要将警告设为错误, 所以在项目属性中应该将其打开 (将链接器警告视为错误)。



4. 在工程函数中使用 "UNREFERENCED_PARAMETER" 宏是编译器忽略形参为引用的警告。

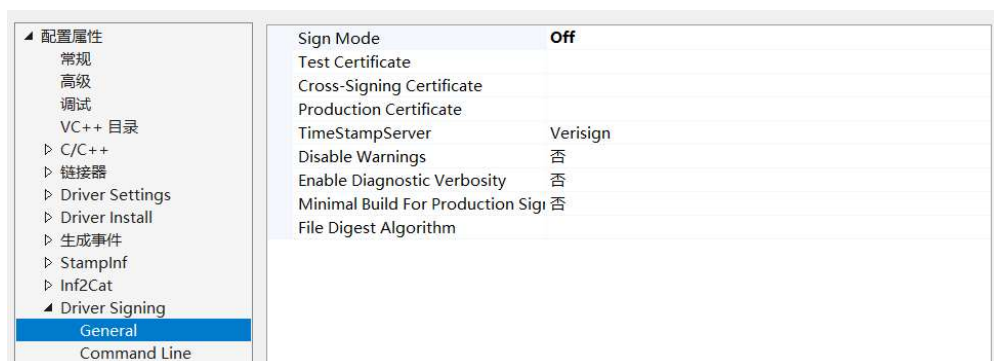


5. 签名文件的错误可以通过工程中的 ".inf" 配置文件进行设置, 不使用签名文件可以在项目属性中进行关闭, 如下图所示:

```

1 ;
2 ; Hello.inf
3 ;
4
5 [Version]
6 Signature="$WINDOWS_NT$"
7 Class=System
8 ClassGuid={4d36e97d-e325-11ce-bfc1-08002be10318}
9 Provider=%ManufacturerName%
10 DriverVer=
11 CatalogFile=Hello.cat
12
13 [DestinationDirs]
14 DefaultDestDir = 12

```



调试操作系统

以 WinXP、Win7为例，为其配置调试状态。使用Windbg进行调试时，对应的系统需要以调试模式打开。

XP

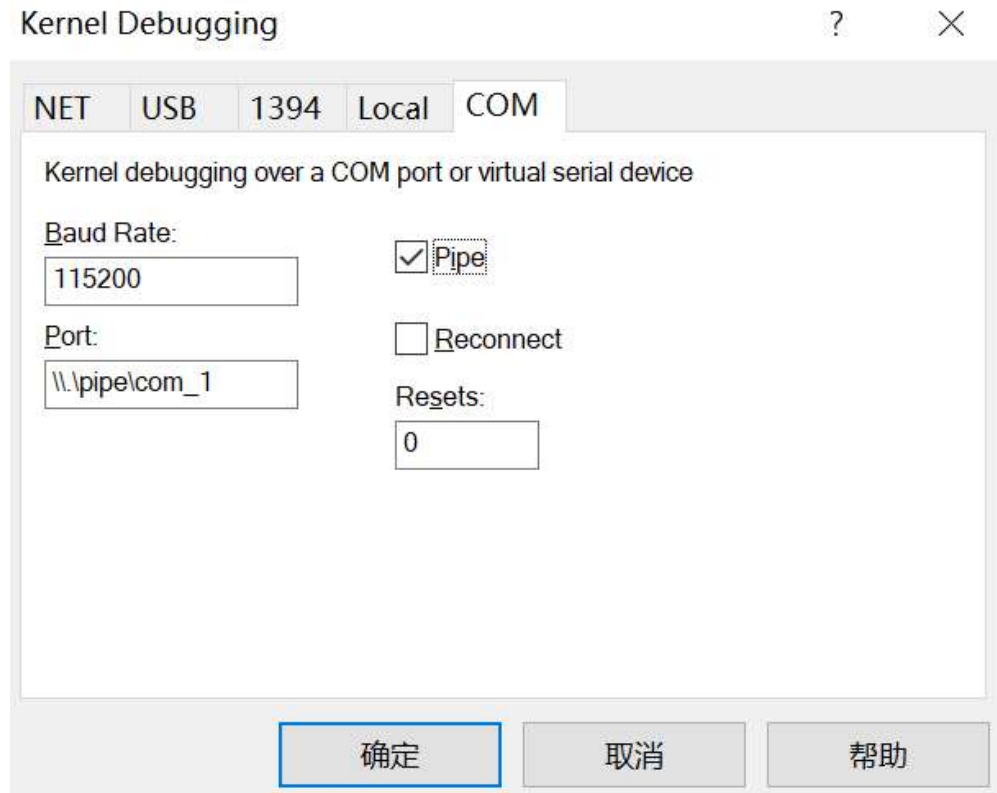
调试模式的配置如下：

1. 在XP系统的C盘中找到 "boot.ini" 文件（没有该文件，通过将 "文件夹选项" 中的 "隐藏受保护的操作系统文件" 取消打勾、"显示所有文件和文件夹" 打勾操作即可显示）进行修改，用来配置系统启动的选项（修改该文件前注意进行下备份）。
2. 在 "boot.ini" 文件中添加新的启动项，需要配置 "DEBUGPORT:port" 以及 "BAUDRATE:baud"，用来和硬件（串口）进行通讯。配置参数如下：
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Debug" /noexecute=alwaysoff /fastdetect /debug /DEBUGPORT:com1 /BAUDRATE:115200
3. 编辑虚拟机设置，添加一个 "串行端口"，并进行如下配置：

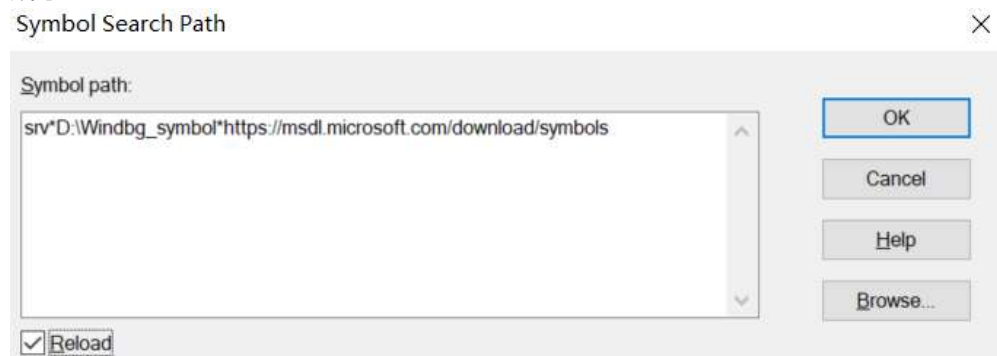


4. 在物理机上打开 Windbg 进行相应的配置，需要通过COM端口或虚拟串行设备进行内核调试。点击 "File" --> "kernel Debugging" --> "COM"，做如下配

置：



5. 配置 Windbg 的 "Symbols Search Path", 配置需要加载的符号路径, 如下图所示:

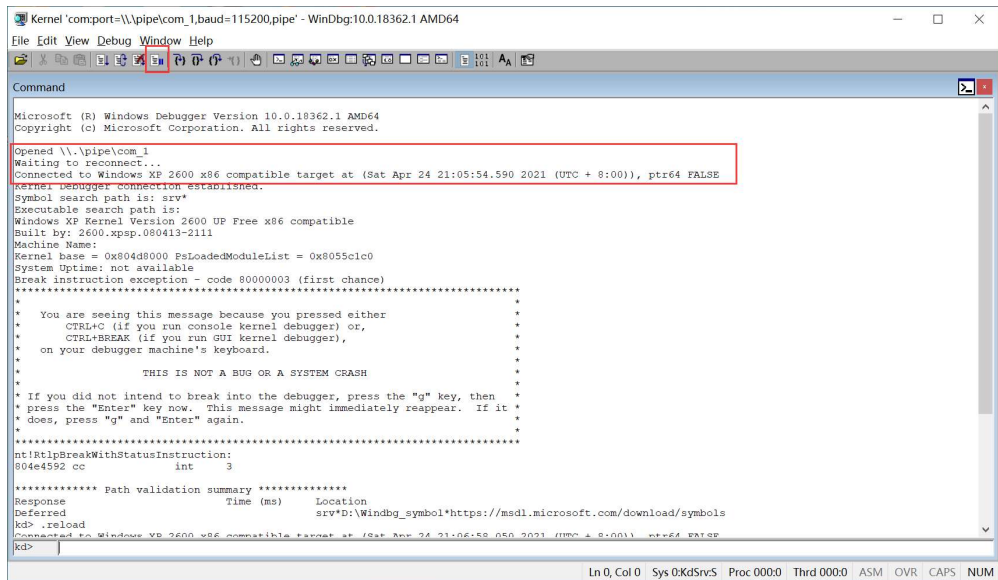


Symbols Path示例:

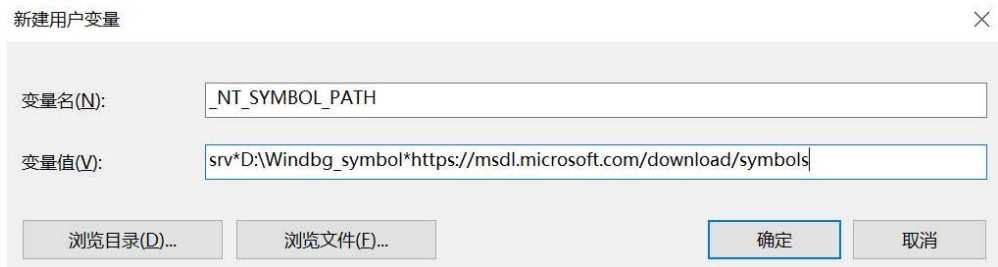
srv*D:\Windbg_symbol*https://msdl.microsoft.com/download/symbols

[官方文档](#)

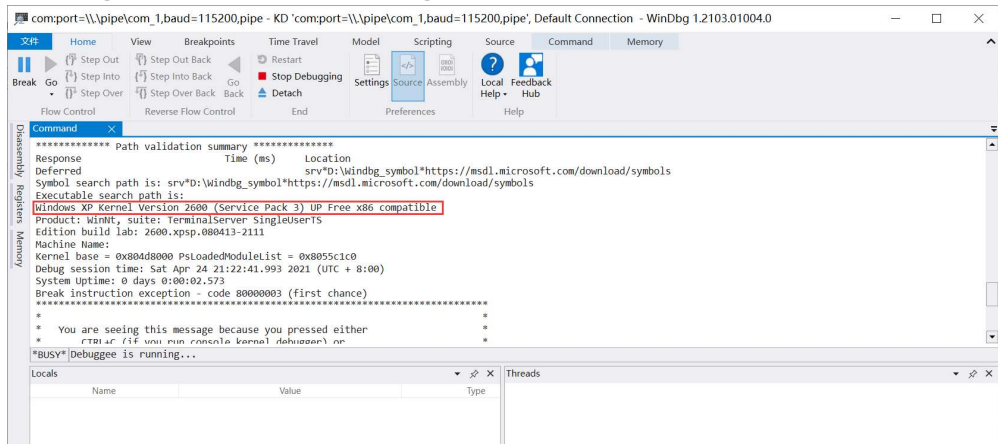
6. 虚拟机以刚才配置的调试模式启动, 打开配置好的 Windbg 进行连接, 连接成功效果图如下图所示:



7. 连接成功后，系统处于被调试状态，运行操作系统，在 WinDbg 中输入 "g" 命令后，被调试的系统也就运行起来。
8. 每次调试操作系统，都需要配置符号路径，为了方便，需要在环境变量中进行设置，操作如下图所示：



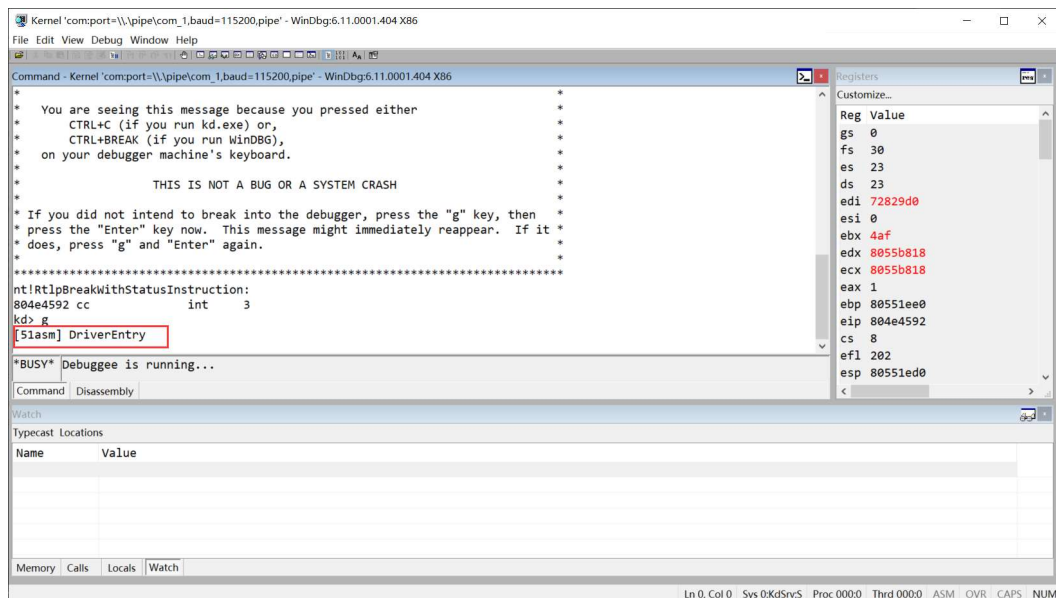
9. WinDbg Preview 配置和 WinDbg 的配置类似，连接成功效果如下：



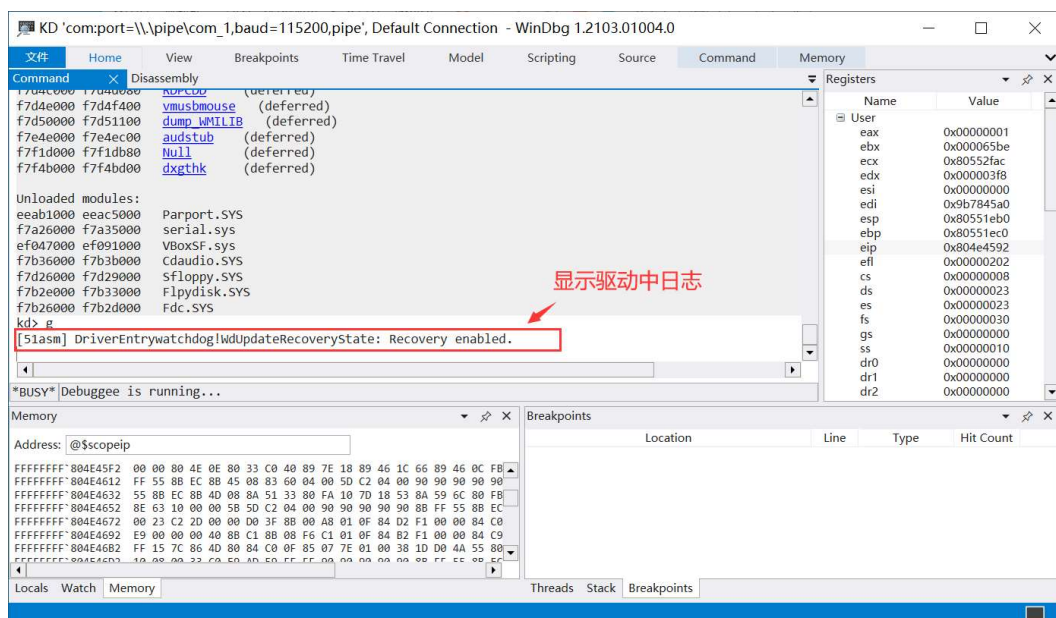
10. "一键连接" 需要右键点击 "Windbg" 属性，在 "快捷方式" 页面的 "目标(T):" 原路径的后面添加 " -b -k com:pipe,port=\\.\pipe\com_1,baud=115200,pipe" 配置信息，之后按照之前的步骤打开虚拟机等待物理机连接调试，在物理机上双击 Windbg 的图片，即可建立连接。

配置调试器界面

Windbg 6.11:



Windbg Preview:



界面配置完成后，需要保存 WinDbg 的工作空间，便于下次直接使用。当系统处于断点状态时，可为其设置快照，方便系统出现问题或者快速打开。

调试系统

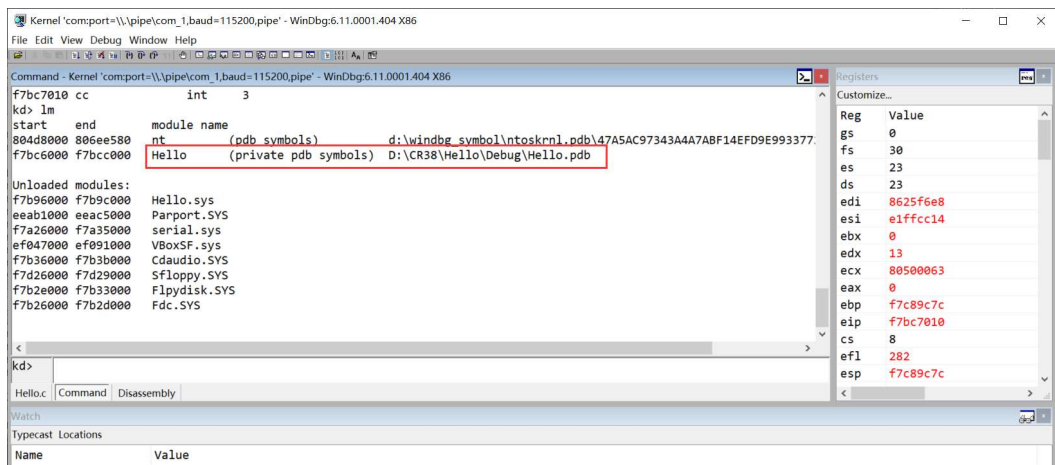
WinDbg 调试程序时常用命令：

- ".detach"：可以断开物理机和被调试的操作系统。
- ".lm"：查看驱动对应的 ".pdb" 文件是否被加载
- ".reload"：重新加载符号

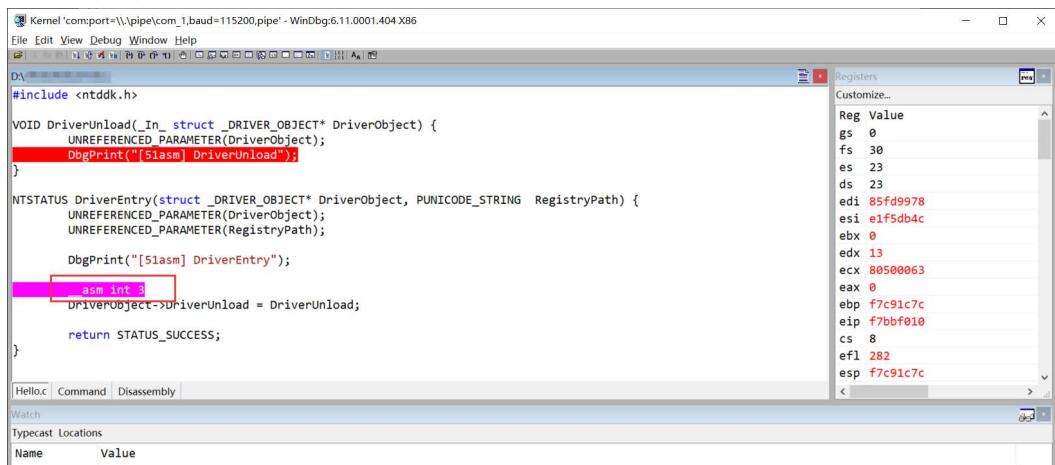
驱动的编写有对应的 ".pdb"，所以在目标系统中运行驱动，可以在物理机中打开 WinDbg，将对应的 ".pdb" 路径作为符号路径添加到 WinDbg 中。



如果 ".pdb" 文件没有加载成功，可以在驱动入口函数代码中写上内联汇编
 "__asm int 3"，驱动入口函数运行后，会产生断点异常，可通过 "lm" 查看驱动
 加载信息。操作系统会自动加载驱动对应的 ".pdb" 文件后，如下图所示：



打开驱动对应的源码可以在对应的代码位置上 "F9" 下断点，运行操作系统，等待断
 点断下，即可调试程序：



在Debug版本的驱动文件内部保存了其对应的 ".pdb" 的绝对路径，如下图所示：


000006E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	RSDS="/="=ù0}A
000006F0	00 00 00 00 52 53 44 53	3D 2F 22 3D F9 30 7D 41	%Žóýİă D:\C
00000700	BC 8E D3 FD CE E5 11 02	03 00 00 00 44 3A 5C 43	R38\Hello\Debug\
00000710	52 33 38 5C 48 65 6C 6C	6F 5C 44 65 62 75 67 5C	Hello.pdb
00000720	48 65 6C 6C 6F 2E 70 64	62 00 00 00 00 00 00 00	T .text\$mn
00000730	00 10 00 00 54 00 00 00	2E 74 65 78 74 24 6D 6E	T * .tex
00000740	00 00 00 00 54 10 00 00	2A 00 00 00 2E 74 65 78	t\$s .ida
00000750	74 24 73 00 00 20 00 00	08 00 00 00 2E 69 64 61	

VS 中调试驱动程序

1. 在 VS 2019 中选择 "菜单" --> "Driver" --> "Test" --> "Configure Devices" -> "Add New Device"

2. 配置调试信息:

Device Configuration ×

 **Adding a new target device**

Display name:

Network host name:

Provisioning Options


☐ **Provision device and choose debugger settings**
Installs and configures components on the target to allow deploying drivers and running tests from Visual Studio. The specified debug settings will also be set on the target, and the connection validated.

☒ **Manually configure debuggers and do not provision**
Sets debuggers settings in Visual Studio to connect to the target, but does no validation or provisioning.

[Show help](#)

3. 进行 "Configure debugger settings", 配置和 "Windbg" 基本类似:

Device Configuration ×

 **Configure debugger settings**

WinXP

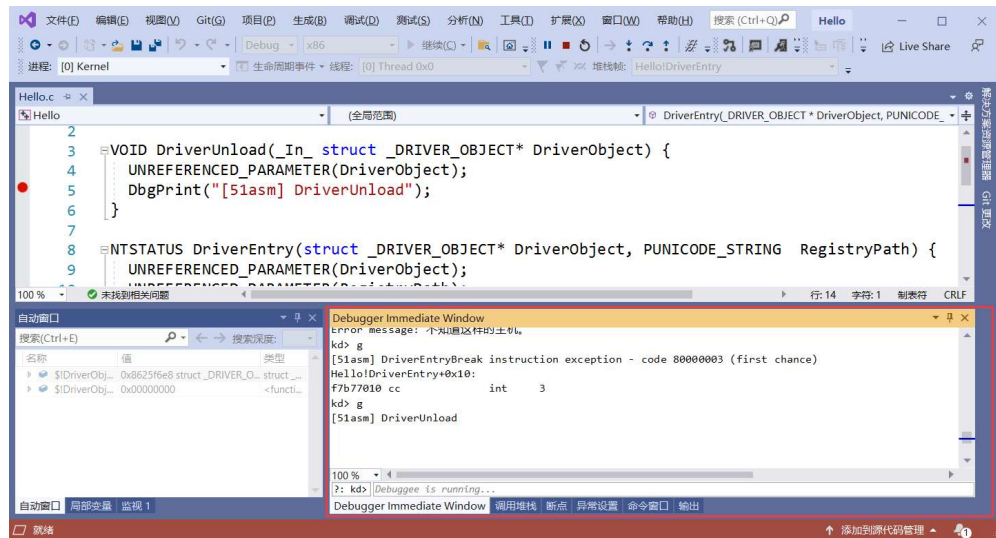
- ▶ Windows Debugger - User Mode
- ▲ Windows Debugger - Kernel Mode

Connection Type	Serial
Baud Rate	115200
Pipe	<input checked="" type="checkbox"/>
Reconnect	<input type="checkbox"/>
Pipe name	com_1
Target Port	\\\\pipe\\com_1

The serial port used on the target device for communication between the target device and host computer.

[Show help](#)

4. 配置完成后, 在VS界面中点击 "Debugging Tools for Widnows - Kernel Debugge" 即可



推荐使用WinDbg，运行效率要比VS高。

相关函数的使用

- DbgBreakPoint(): 内核提供的断点宏，在调试操作系统时不会导致操作系统出现蓝屏，但是不在调试状态就会导致操作系统出现蓝屏。
- KbBreakPoint(): 编译为 Release 时，断点不存在，为空宏；编译为 Debug 和 "DbgBreakPoint()" 功能一样。

```
#if DBG

#define KdPrint(_x_) DbgPrint _x_
#define KdPrintEx(_x_) DbgPrintEx _x_
#define vKdPrintEx(_x_) vDbgPrintEx _x_
#define vKdPrintExWithPrefix(_x_) vDbgPrintExWithPrefix _x_
#define KbBreakPoint() DbgBreakPoint()

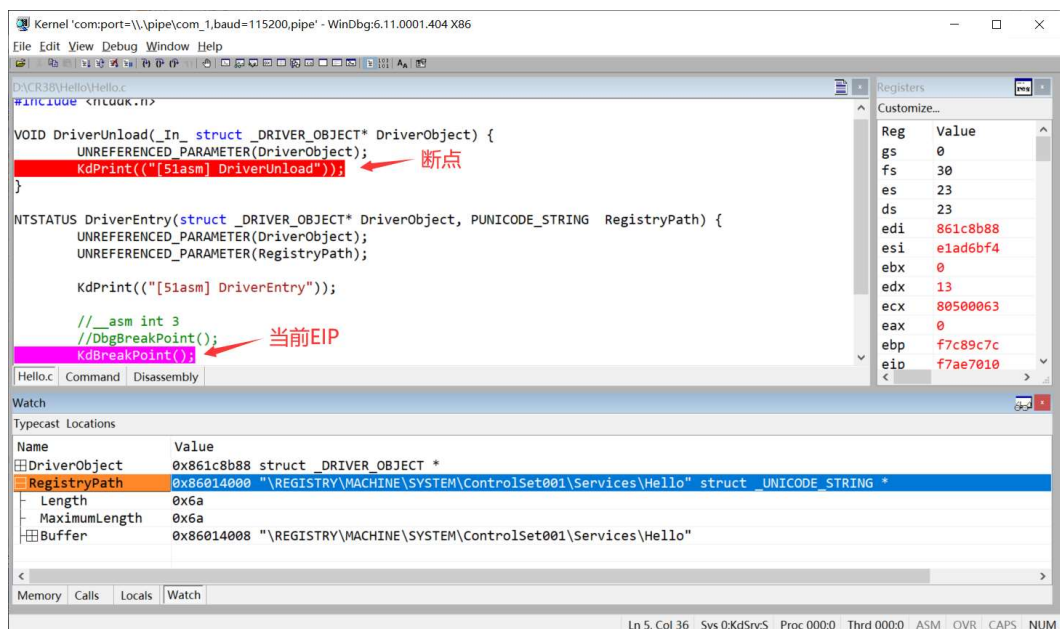
#define KdBreakPointWithStatus(s) DbgBreakPointWithStatus(s)

#else

#define KdPrint(_x_)
#define KdPrintEx(_x_)
#define vKdPrintEx(_x_)
#define vKdPrintExWithPrefix(_x_)
#define KbBreakPoint()

#endif
```

- KbPrint(): 编译为 Release 时，不会打印日志；编译为 Debug 和 "DbgPrint()" 功能一样。使用示例：
KdPrint(("[Test] DriverEntry"));



Win7

不在提供 "boot.int" 文件。

使用 "bcdedit" 配置双击调试以及系统的启动选项。输入 "bcdedit /?" 查看相关的命令说明。

操作如下：

1. 管理员身份运行 "cmd", 输入以下命令, 拷贝当前的启动项的GUID, 设置新的启动项。

```
bcdedit /copy {current} /d "Win7 DUG"
```

```
C:\Users\ileemi>bcdedit /copy {current} /d "Win7DUG"
已将该项成功复制到 {652232ff-9093-11eb-888a-f6f1eb124313}。
```

2. 设置调试启动:

```
bcdedit /debug {652232ff-9093-11eb-888a-f6f1eb124313} ON
```

```
C:\Users\ileemi>bcdedit /debug {652232ff-9093-11eb-888a-f6f1eb124313} ON
操作成功完成。
```

3. 设置串口调试接口:

```
bcdedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:115200
```

下列命令设置全局调试程序设置在 com1 上以 115,200 波特进行串行调试:

```
bcdedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:115200
```

下列命令设置全局调试程序设置使用通道 23 进行 1394 调试:

```
bcdedit /dbgsettings 1394 CHANNEL:23
```

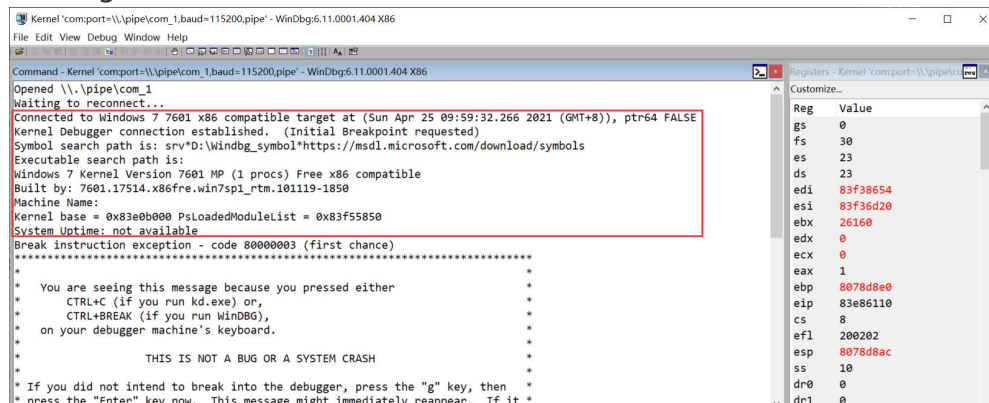
下列命令设置全局调试程序设置使用目标名称 DEBUGGING 进行 USB 调试:

```
bcdedit /dbgsettings USB TARGETNAME:DEBUGGING
```

```
C:\Users\ileemi>bcdedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:115200
操作成功完成。
```

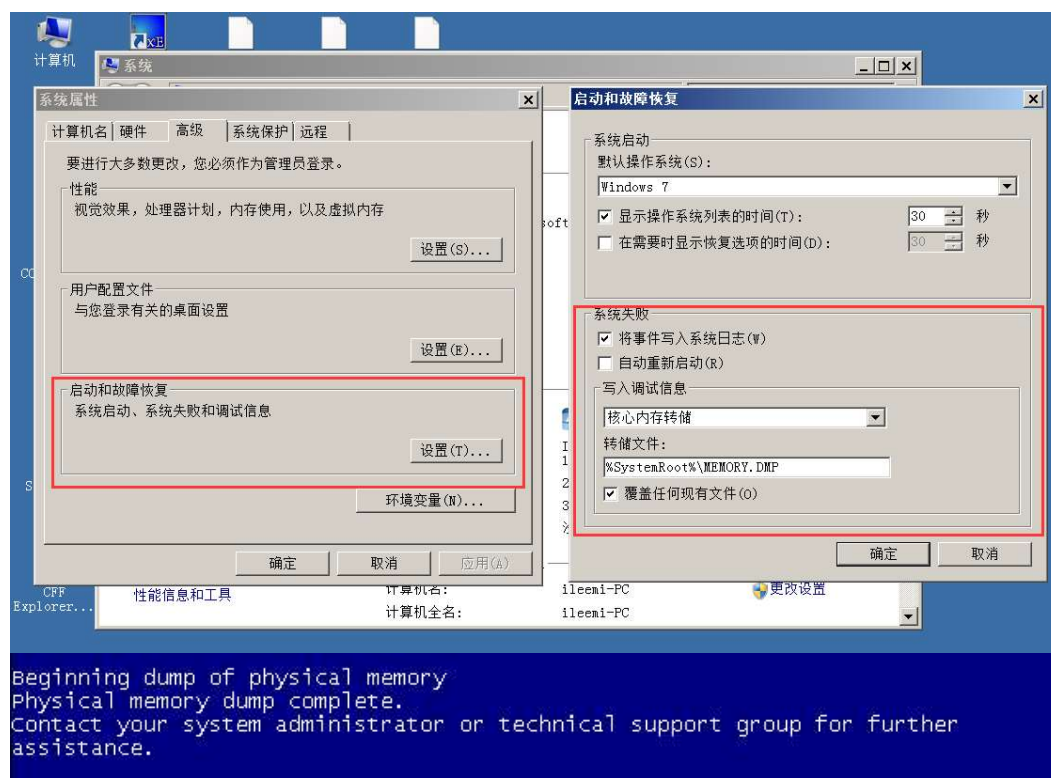
4. 配置完成, 关闭Win7虚拟机, 添加新的串口, 进项相应的配置, 操作和WinXP一致。

5. Windbg连接成功的效果如下：



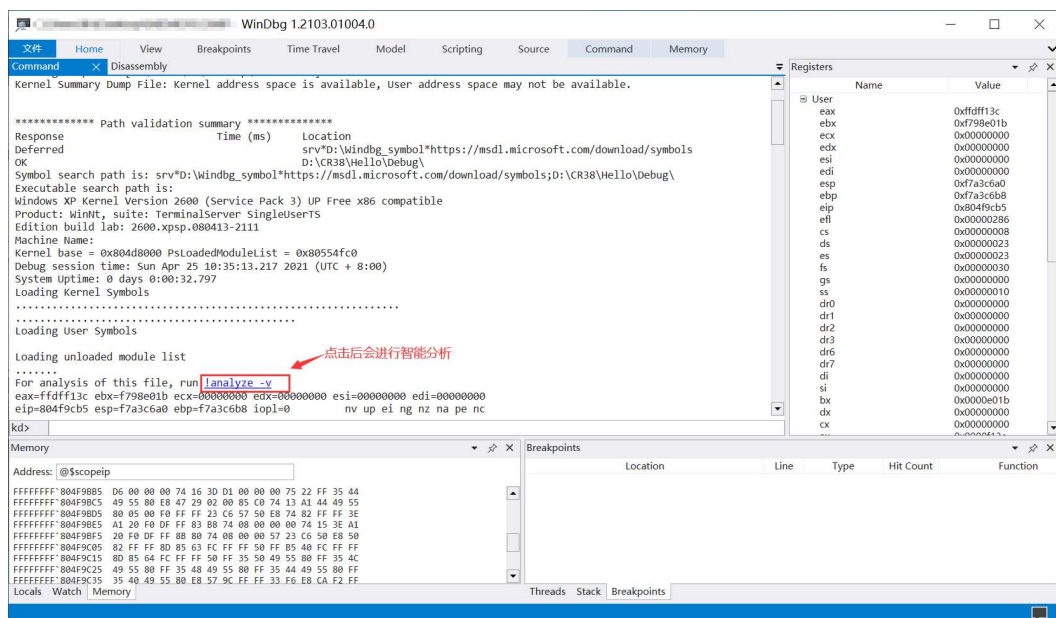
保持系统崩溃时的信息

在操作系统属性中设置 "写入调试信息" 为 "核心内存转储", 当操作系统出现故障（蓝屏时）会保存当时的环境信息，方便调试进行调试，找出当时导出系统崩溃的原因。

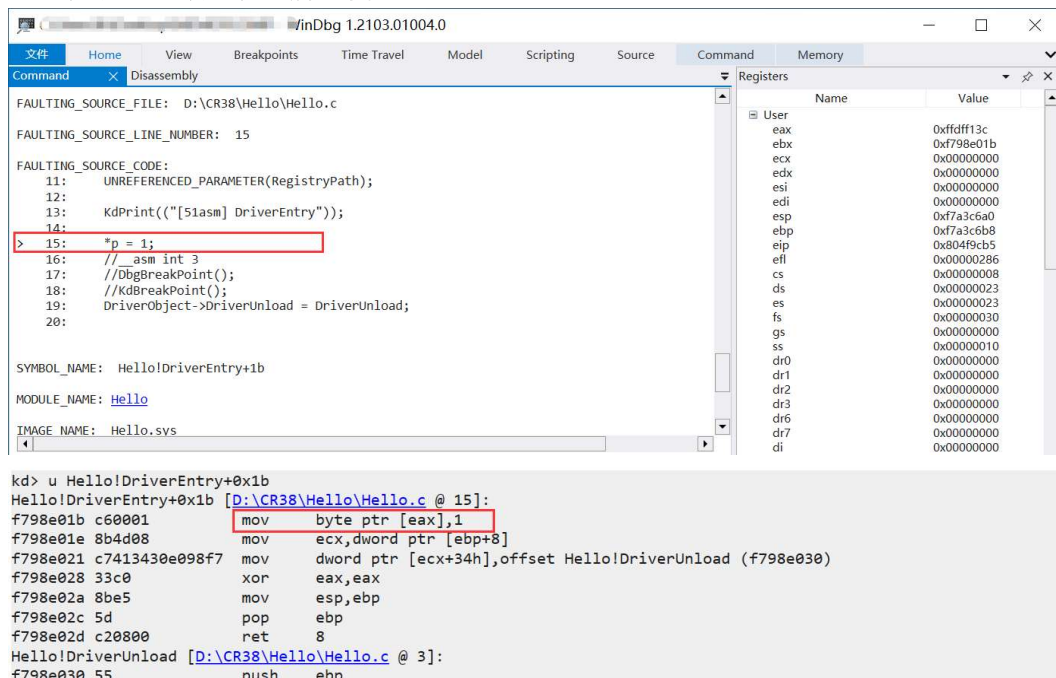


调试系统崩溃文件

在崩溃操作系统中的C盘，Windows下找到对应的 ".dmp" 文件，拷贝到物理机，通过 Windbg打开（File --> Open Crash Dump）来还原当时的操作系统崩溃时的环境，进行调试。



点击或者输入 "!analvze -v" 会对 ".dmp" 文件进行智能分析，如下图所示，Windbg 已经标识了导致系统蓝屏的代码：



物理机的Windbg和虚拟操作系统连接时，如果虚拟机中的驱动导致操作系统出现了异常，调试器会收集导致其出现异常的原因，调试器会自动进行dump。

内核API分类

- Io... (I/O管理器)：负责对硬件、文件等设备进行操作。获取和报告有关驱动程序设备和当前平台的硬件配置信息。
IoCreateDevice：创建驱动
IoAttachDevice
IoCallDriver
IoDeleteDevice
...
- Cm...：配置管理器(注册表)

- Ob...: 对象相关
ObReferenceObject
ObDereferenceObject
ObReferenceObjectByHandle
..
- Mm...: 内存相关
MmIsAddressValid: 判断内存是否可以访问
MmMapIoSpace: 内存映射
MmGetPhysicalAddress: 获取虚拟内存地址在物理内存条中的位置
...
- Ex...: 执行层相关
ExAllocatePool: 申请内存
ExFreePool: 释放内存
InterlockedExchangeAdd: 自加锁
...
- Ke...: 核心层相关
KeBugCheck: 会导致操作系统蓝屏
KeEnterCriticalRegion: 进入临界区
KeGetCurrentProcessorNumber: 获取CPU的信息
KeInitializeEvent: 事件对象
...
- Zw...: 与RING3功能相似的函数
ZwCreateEvent: 创建同步对象
ZwCreateFile: 创建文件
...
- Ps...: 进程线程相关
PsGetCurrentProcessId: 获取内核中当前正在执行进程PID
PsGetCurrentThreadId: 获取内核中当前正在执行线程PID
PsGetVersion: 获取系统版本
...
- RTL: 更安全的C运行库