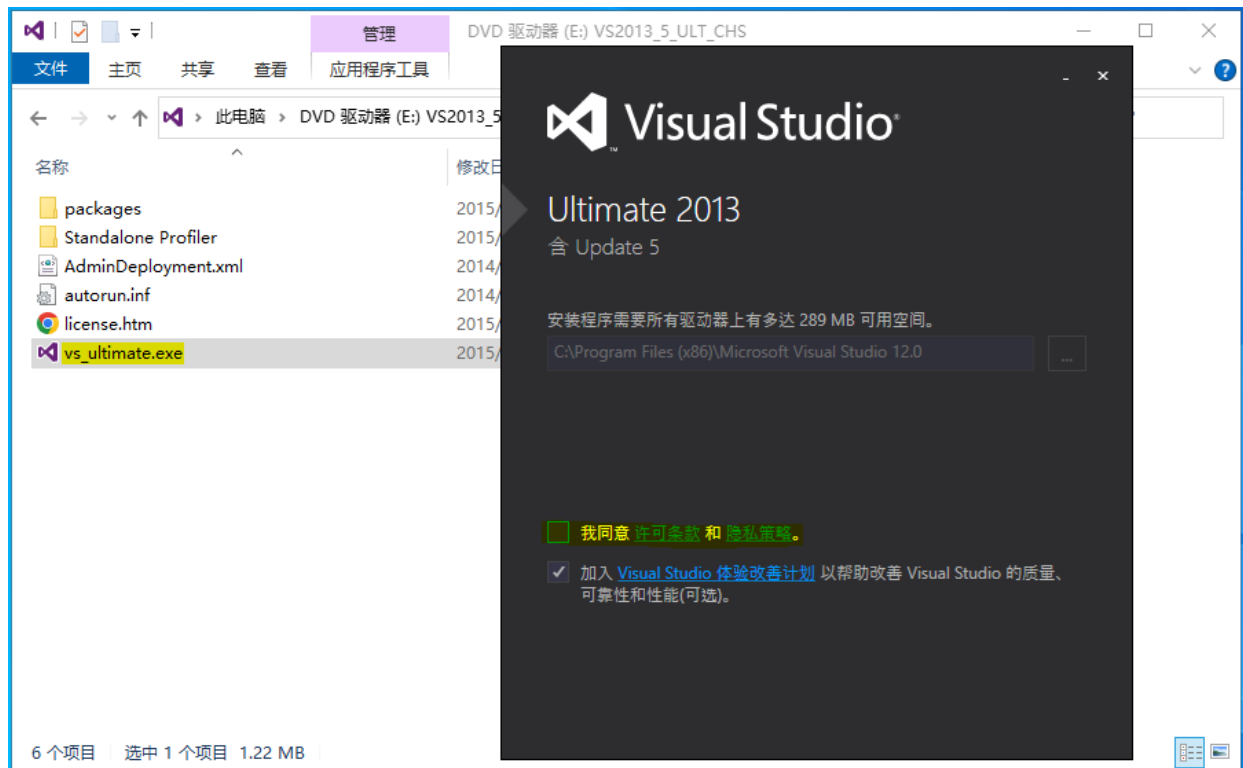


在正式开始驱动开发之前，需要自行搭建驱动开发的必要环境，首先我们需要安装 Visual Studio 2013 这款功能强大的程序开发工具，在课件内请双击 ISO 文件并运行内部的 vs_ultimate.exe 安装包，Visual Studio 的安装非常的简单，您只需要按照提示全部选择默认参数即可，根据机器配置不同可能需要等待一段时间；

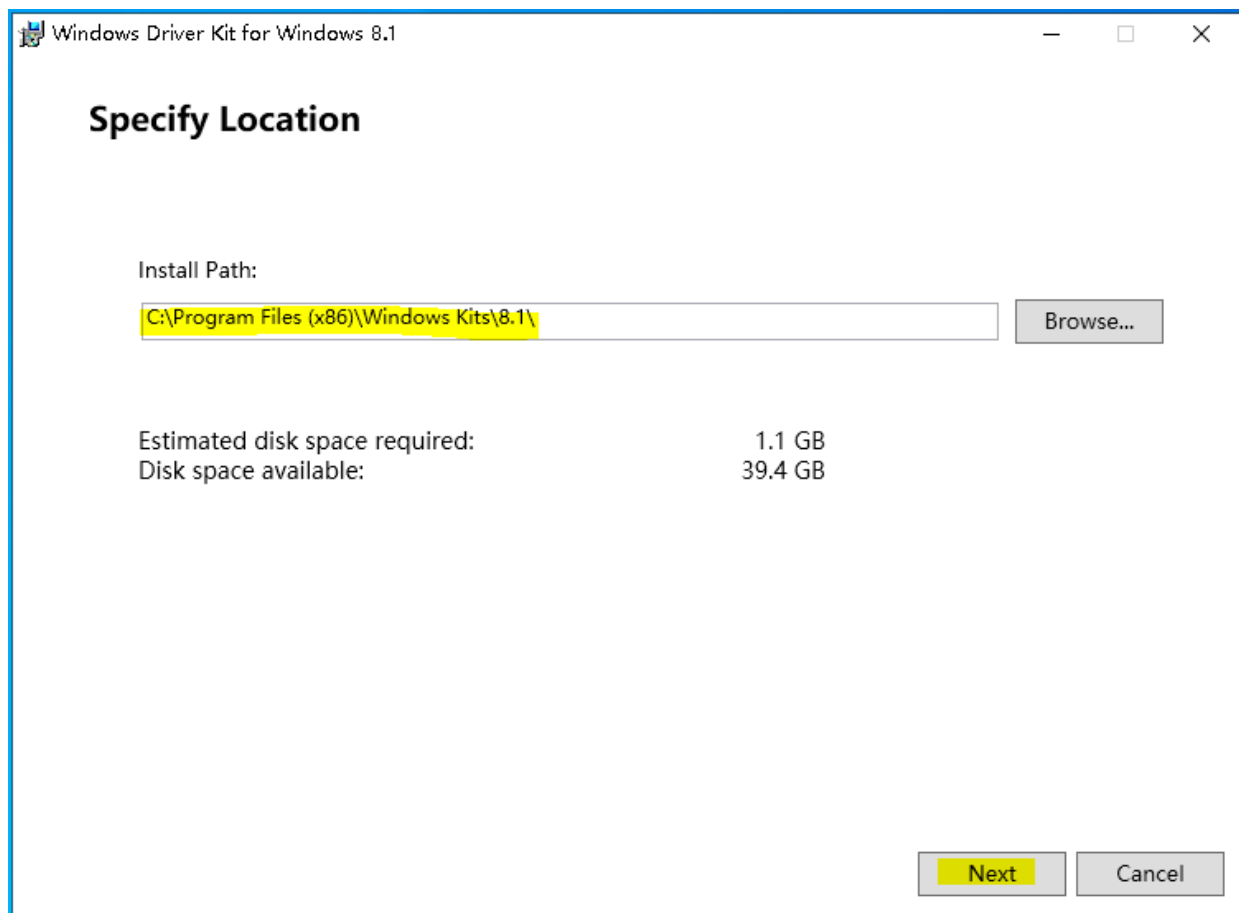
配置驱动开发环境

在正式开始驱动开发之前，需要自行搭建驱动开发的必要环境，首先我们需要安装 Visual Studio 2013 这款功能强大的程序开发工具，在课件内请双击 ISO 文件并运行内部的 vs_ultimate.exe 安装包，Visual Studio 的安装非常的简单，您只需要按照提示全部选择默认参数即可，根据机器配置不同可能需要等待一段时间；

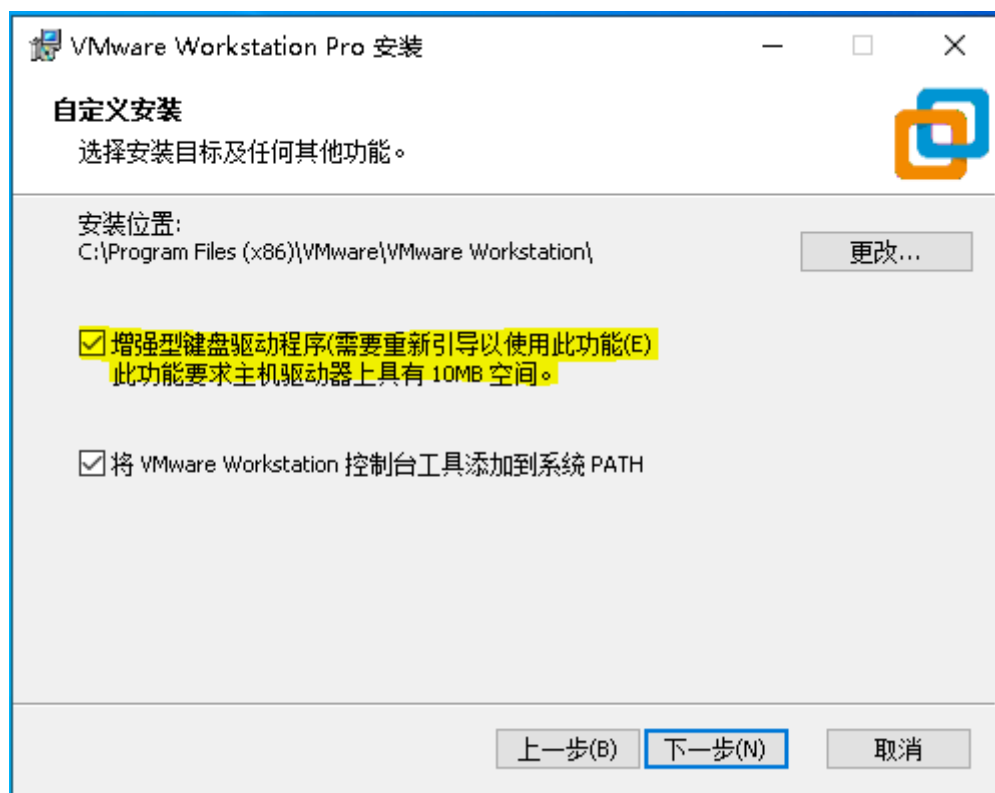


接着读者还需要继续安装 Windows Driver Kit 8.1 工具包，请将该工具包解压缩到桌面，并双击 wdksetup.exe 进行安装，过程中只需要一直下一步，并等待 WDK 工具包安装完成；

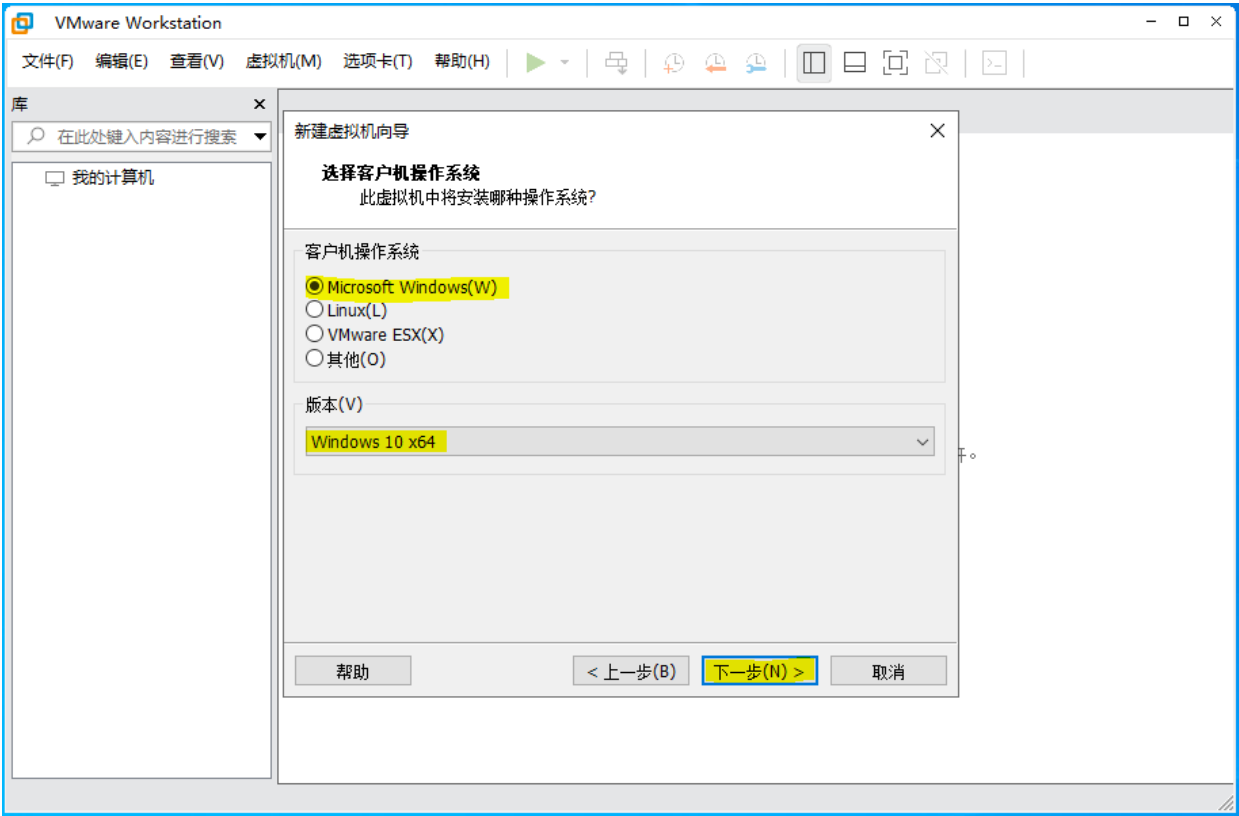
WDK 就是内核编程开发工具包，某些读者可能听说过 DDK 或者 IFSDDK，最典型的开发工具包莫过于 DDK7600，直到目前此类工具包仍然可以正常使用，但并不推荐。



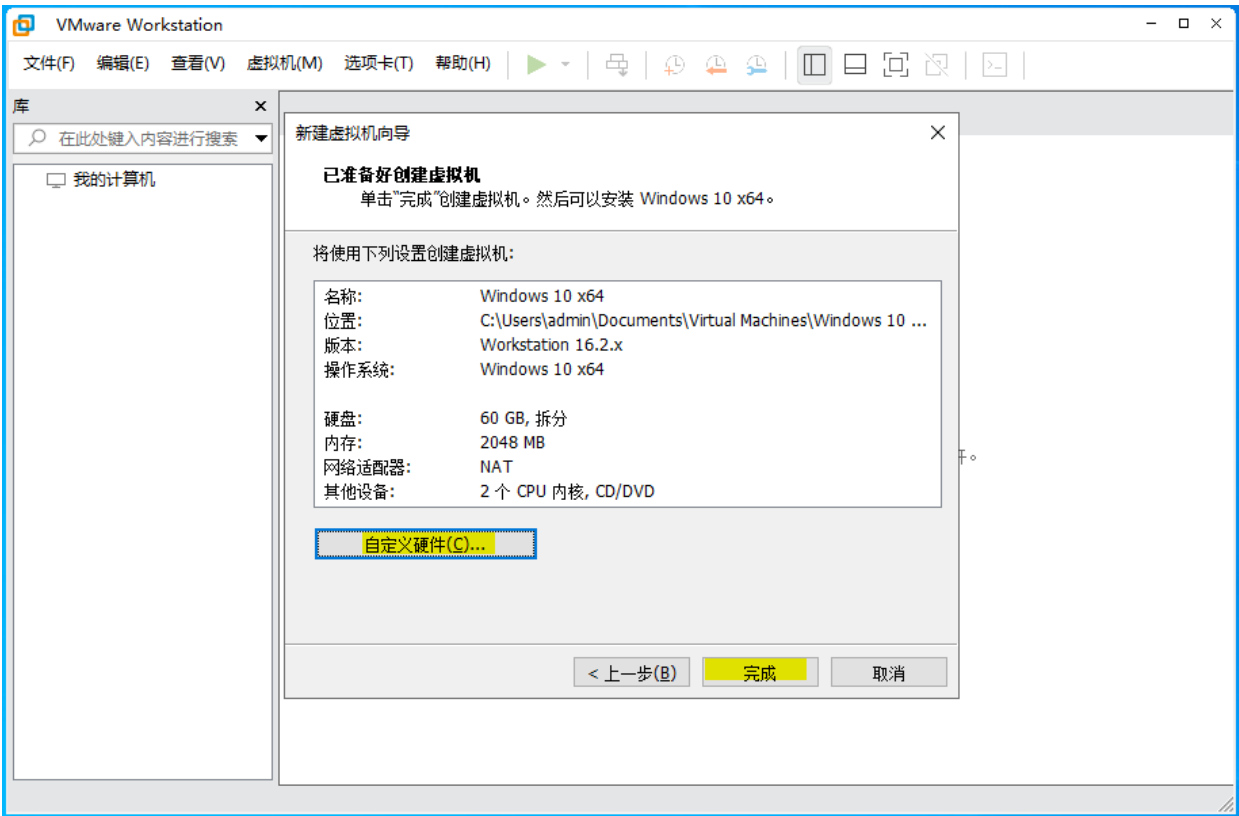
为了能测试驱动程序运行状态，读者需安装 VMware 虚拟机，双击附件中的 vmware-workstation-full-16.2.4-20089737.exe 安装程序一直点击下一步即可，需要注意的是在如下选项中请在增强型键盘驱动程序上打对勾，之后等待安装完毕即可；



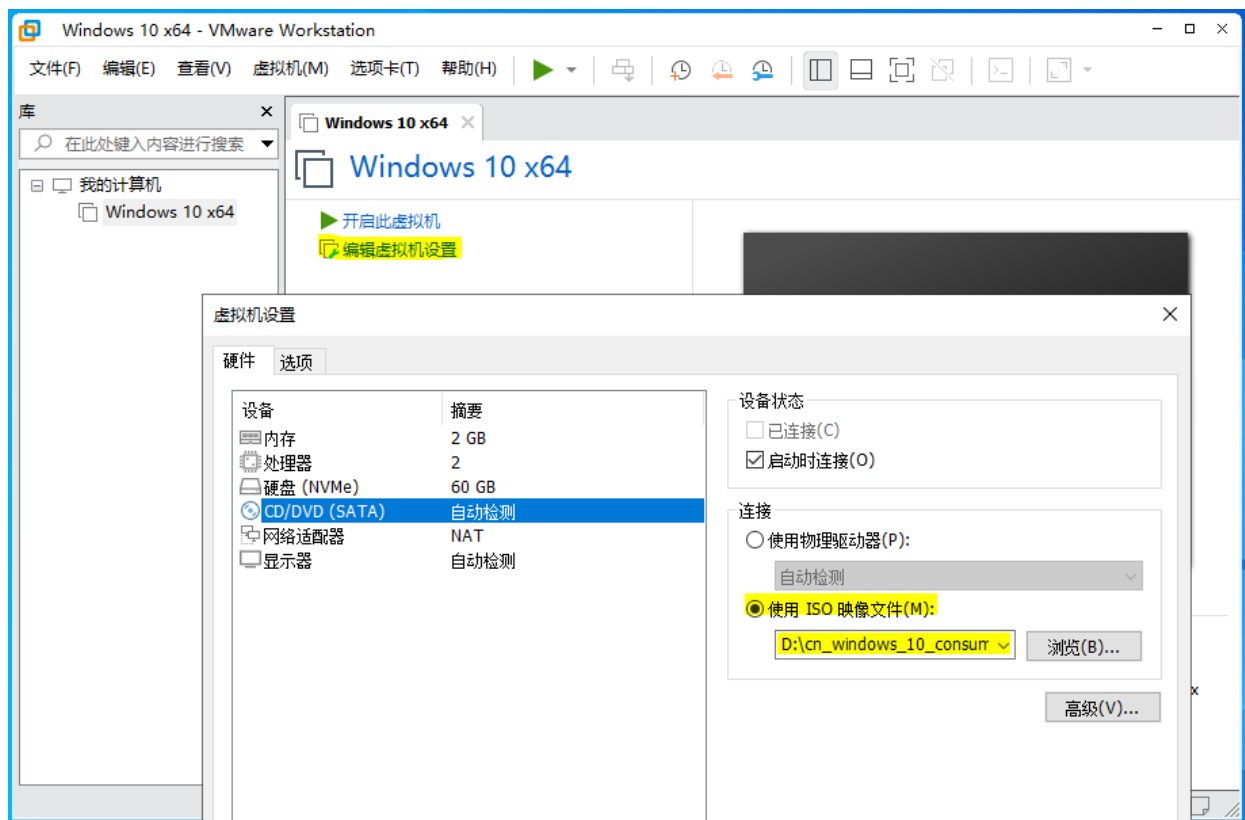
接着打开 VMware 虚拟机，并在【文件】处选择【新建虚拟机】，单机下一步并选中【稍后安装操作系统】，在操作系统选择页面选择【Win10 x64】版本。



在硬件配置处，读者可根据自己电脑的配置灵活的选择，当自定义配置完成后，则虚拟机模板将被创建。

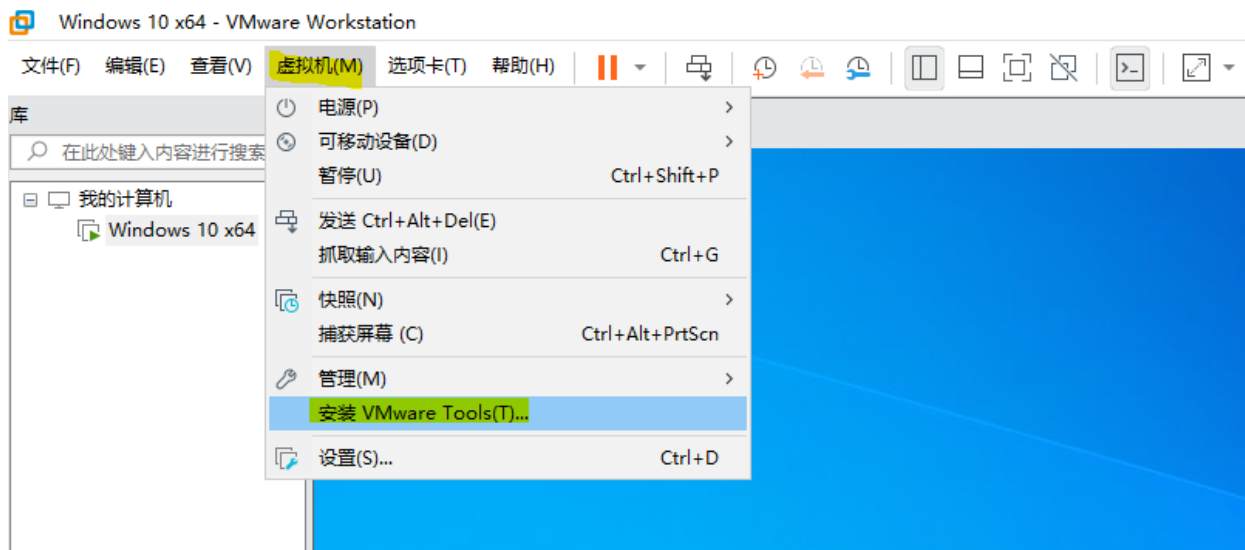


虚拟机模板创建完成后，读者可根据如下配置选择编辑虚拟机设置，并在磁盘位置处将课件中的 `cn_windows_10_consumer_editions_version_1903_x64_dvd_8f05241d.iso` 挂载到虚拟机上；

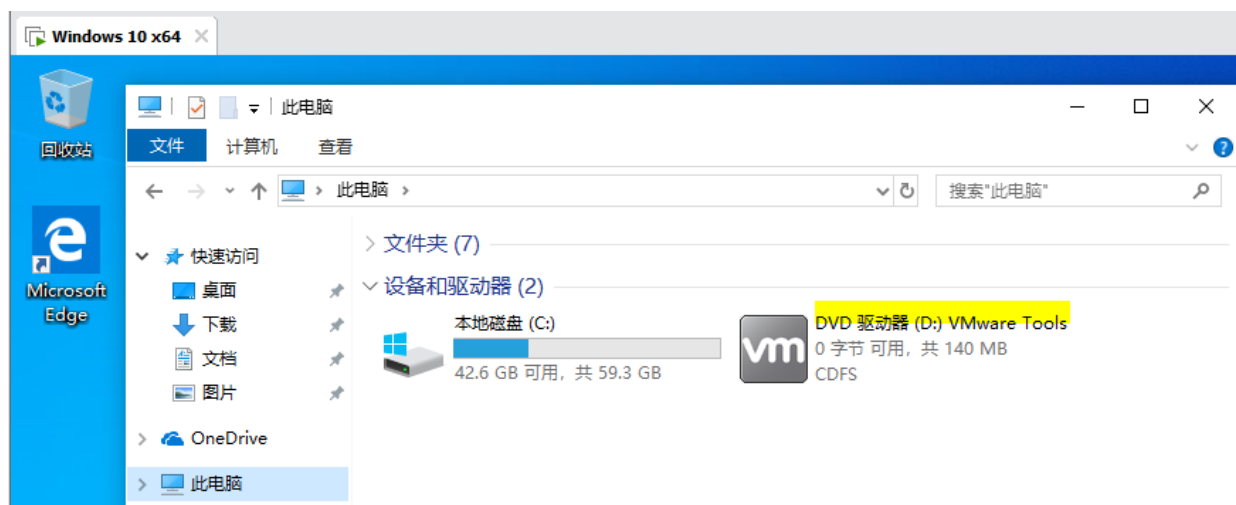


点击开启虚拟机，并按照提示将 windows 系统正确的安装，需要注意的是在选择版本时，读者最好使用 教育版 与笔者开发环境保持一致，至此只需等待系统安装完毕，根据系统差异安装时间可能有所差别，耐性等待即可；

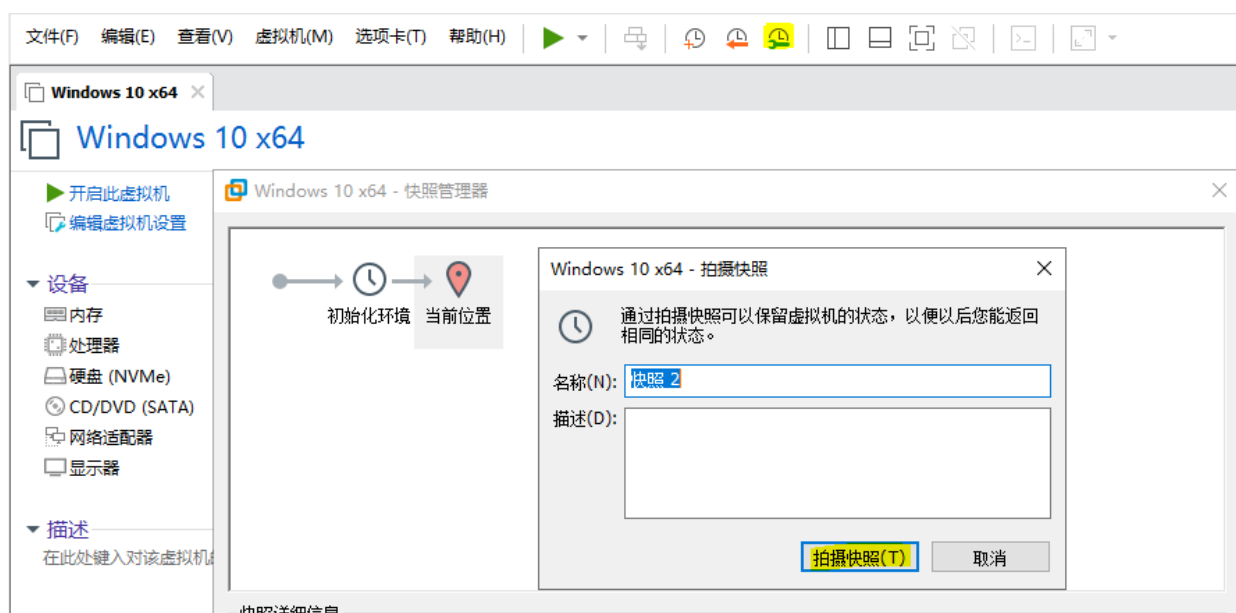
当一切安装就绪后我们需要在系统中安装 VMware Tools 工具，该组件在安装后可让虚拟机具备有拖拽上传文件的功能，且鼠标键盘将可以自由切换，该功能是我们必须要用到的；



安装 VMware Tools 工具很容易，只需要点击安装菜单，后会在虚拟机中出现DVD驱动器，此时双击驱动器并按照要求安装即可，安装完成后重启系统，此时则具备了拖拽上传功能；

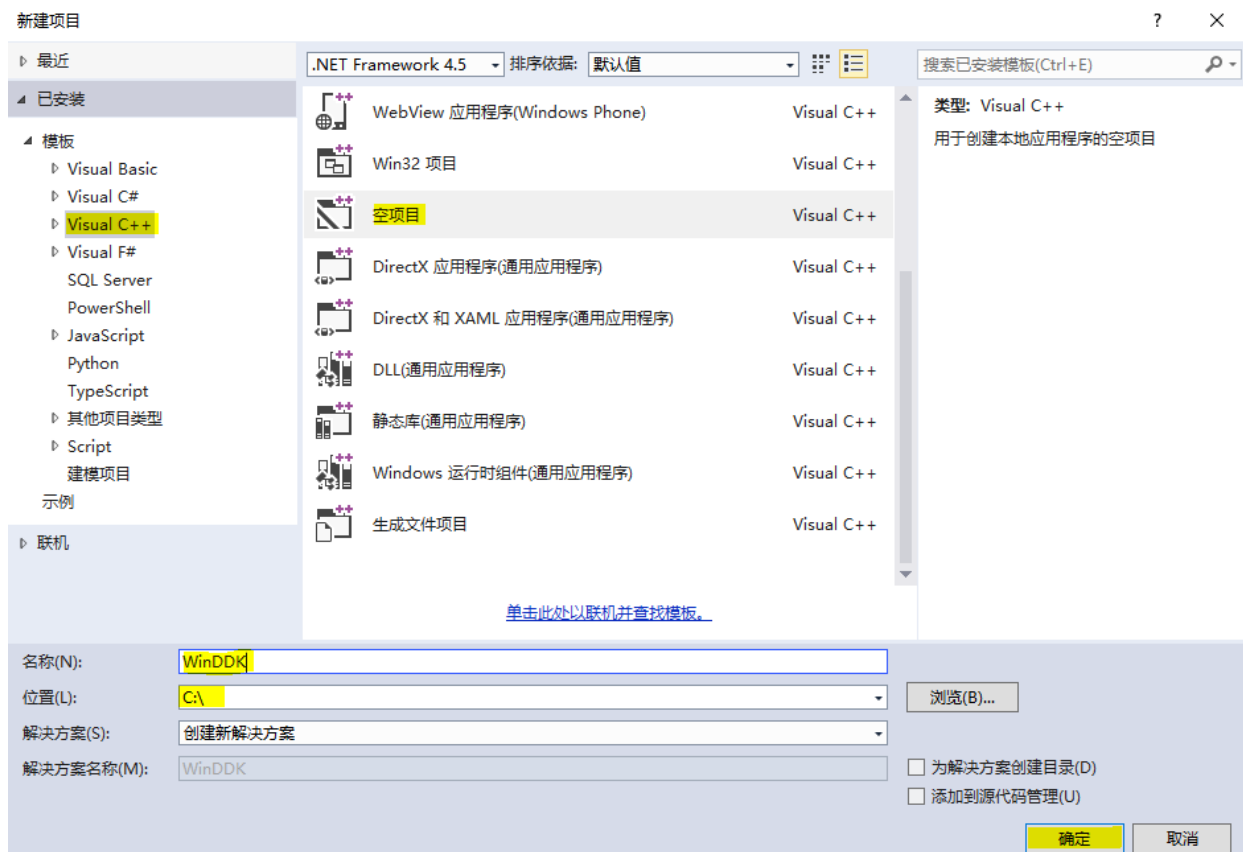


当这些都做好以后，建议用户关闭虚拟机，并点击【虚拟机】菜单，找到【快照】并拍摄一个快照，快照的作用是当虚拟机系统出现问题后可快速恢复到初始模式，避免重装系统，在后续课程中读者会出现无数次的蓝屏，而虚拟机快照的快速恢复功能则是一个很好的选择；

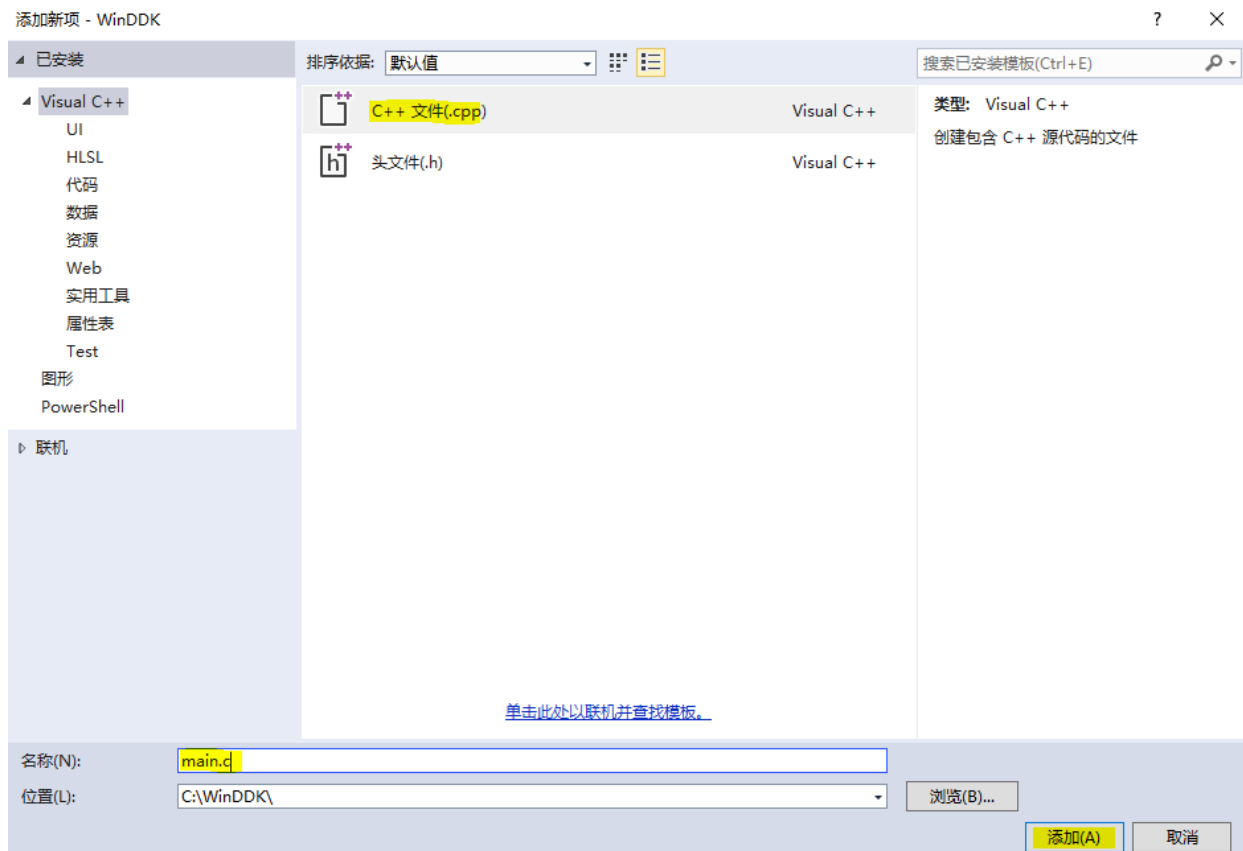


配置驱动开发模板

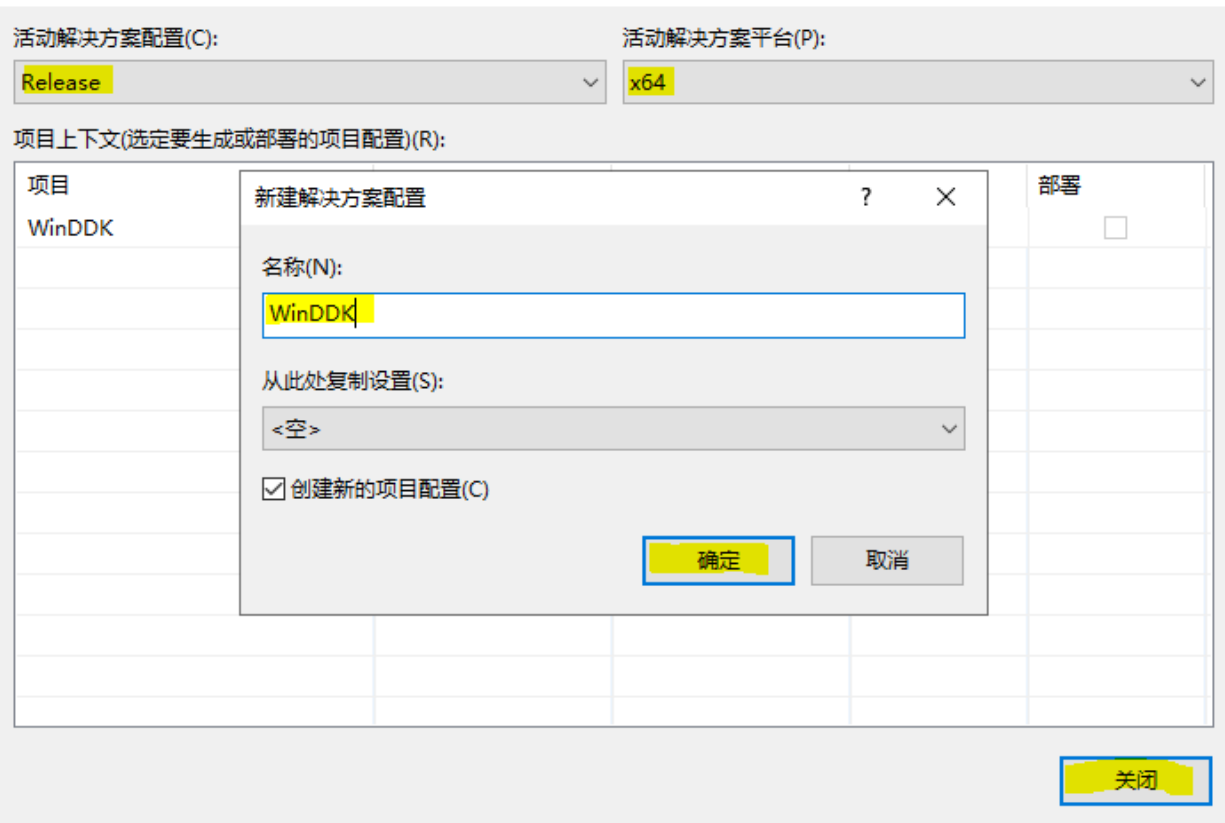
1. 打开 Visual Studio 开发工具，然后选择【文件】菜单新建项目，并在已安装模板中选中【Visual C++】新建空项目，并将项目名称命名为【WinDDK】点击确定。



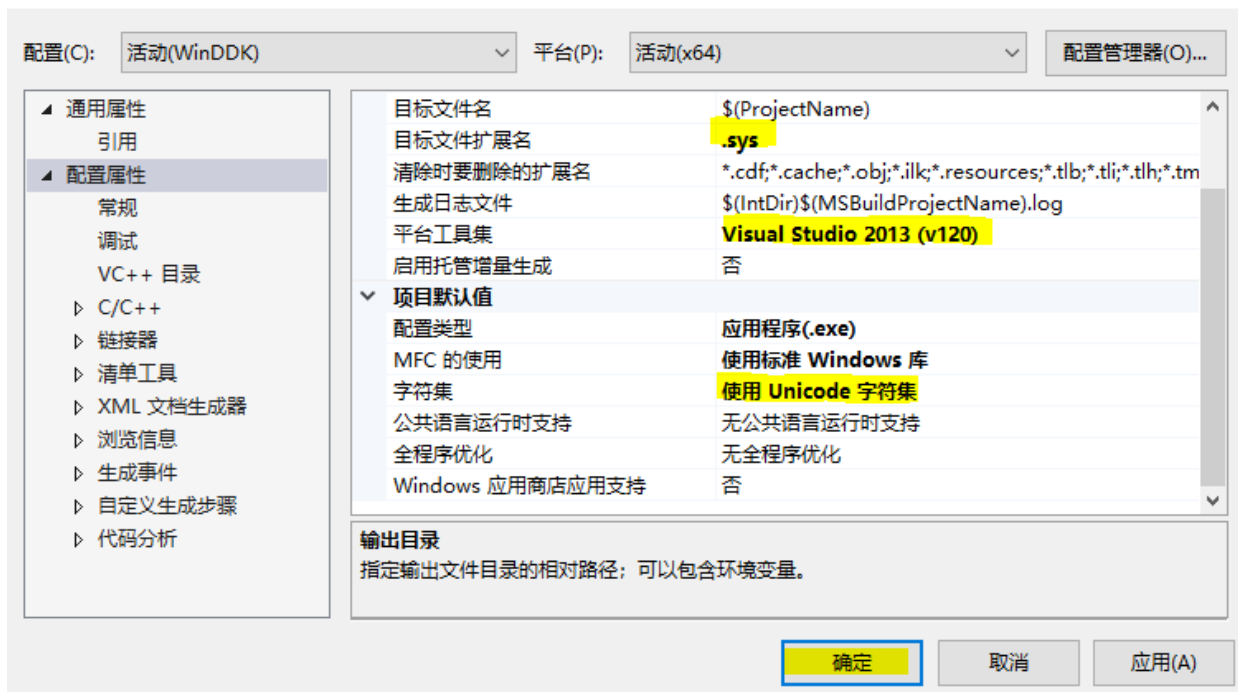
2.依次选择【解决方案视图-源文件-添加新建项】选项卡，或者直接按下 `Ctrl + Shift + A` 快捷打开菜单，并创建 `main.c` 文件。



3.接着需要修改配置管理器，添加自定义配置管理，选择【生成-配置管理器-新建】选项卡，此处我们命名为 `WinDDK` 即可。



4.修改配置属性中的【常规】属性，点击菜单栏中的调试，选择【WinDDK属性-配置-常规】修改为标黄处所示内容即可。



5.配置可执行文件路径与导入库路径，这里我们选择【配置属性-VC++目录】依次将如下信息填入配置项。

可执行目录

C:\Program Files (x86)\windows Kits\8.1\bin\x64

C:\Program Files (x86)\windows Kits\8.1\bin

```
C:\Program Files (x86)\windows Kits\8.1\Include\km
C:\Program Files (x86)\windows Kits\8.1\Include\shared
C:\Program Files (x86)\windows Kits\8.1\Include\um
C:\Program Files (x86)\windows Kits\8.1\Include\wdm\kmdf\1.13
C:\Program Files (x86)\windows Kits\8.1\Include\wdm\umdf\2.0
C:\Program Files (x86)\windows Kits\8.1\Include\winrt
```

C:\Program Files (x86)\windows kits\8.1\Lib\win7\km\x64

```
C:\Program Files (x86)\windows Kits\8.1\Lib\win7\km\x64
C:\Program Files (x86)\windows Kits\8.1\Lib\wdf\kmdf\x64\1.13
C:\Program Files (x86)\windows Kits\8.1\Lib\wdf\umdf\x64\2.0
C:\Program Files (x86)\windows Kits\8.1\Lib\winv6.3\um\x64
C:\Program Files (x86)\windows Kits\8.1\Lib\winv6.3\km\x64
```

WinDDK 属性页

配置(C): 活动(WinDDK) 平台(P): 活动(x64) 配置管理器(O)...

通用属性
引用
配置属性
常规
调试
VC++ 目录
C/C++
链接器
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析

常规

可执行文件目录 C:\Program Files (x86)\Windows Kits\8.1\bin\x64;
包含目录 C:\Program Files (x86)\Windows Kits\8.1\Include;
引用目录 C:\Program Files (x86)\Windows Kits\8.1\Lib\win;
库目录 C:\Program Files (x86)\Windows Kits\8.1\Lib\win;
Windows 运行库目录 \$(WindowsSDK_MetadataPath);
源目录 \$(VC_SourcePath);
排除目录 \$(VC_IncludePath); \$(WindowsSDK_IncludePath); \$(MSI

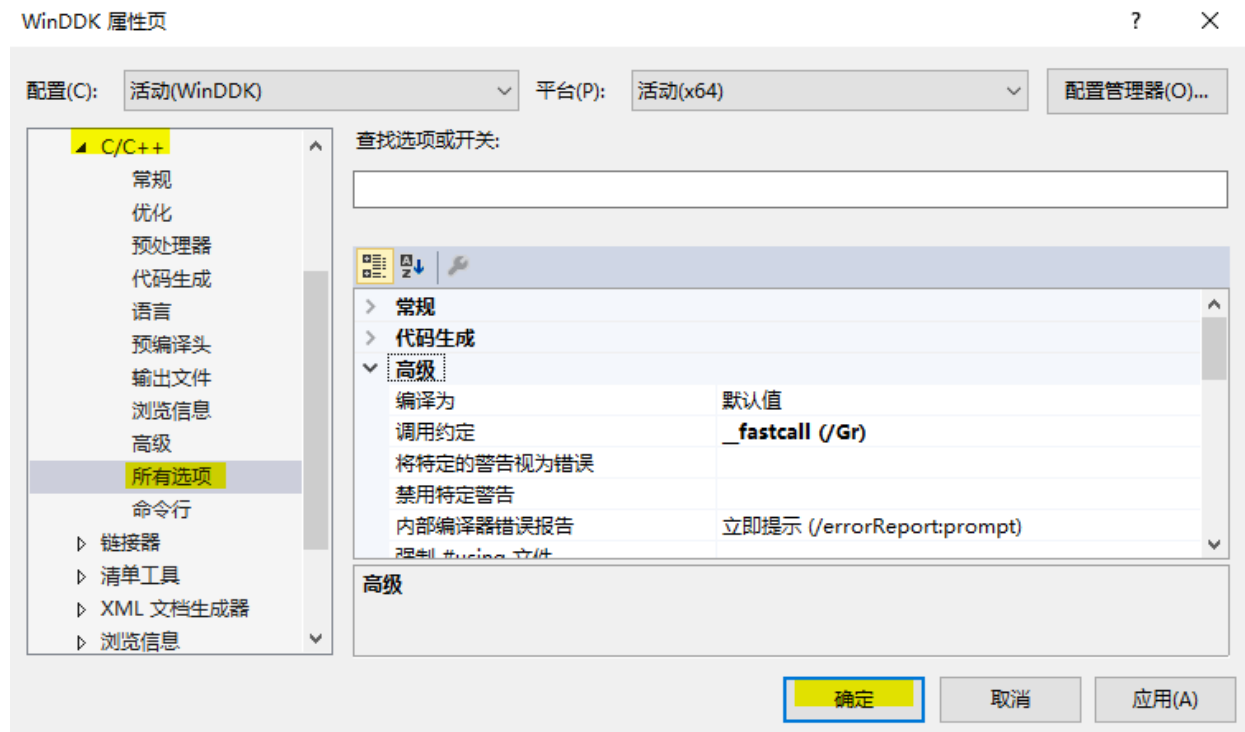
库目录
生成 VC++ 项目期间, 搜索库文件时使用的路径。 与环境变量 LIB 相对应。

确定 取消 应用(A)

6.配置C/C++优化选项，在配置属性中找到【C/C++-所有选项】并依次修改下方几个关键位置。

安全检查	禁用安全检查 (/GS-)
将警告视为错误	否 (/WX-)
警告等级	关闭所有警告
启用C++异常	否
调用约定	<u>_fastcall</u> (/Gr)
优化	<u>已禁用</u> (/Od)
运行库	<u>多线程</u> (/MT)
预处理器定义	<u>_AMD64_</u> ; <u>_DDK_</u> ; <u>_WIN32_WINNT=0x0501</u> ;WINVER=0x0501;_NDEBUG;DBG=0;%
(PreprocessorDefinitions)	

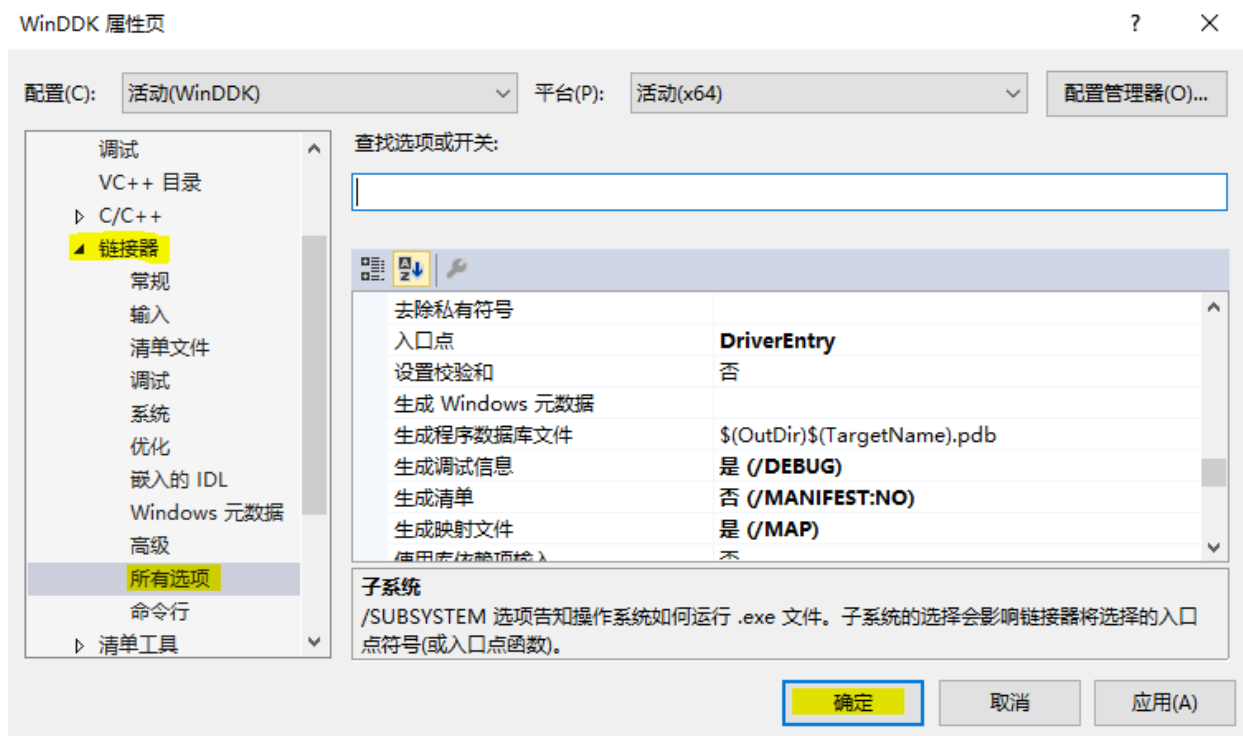
当如上文件配置完成后，最终效果如下图所示；



7.配置连接器选项，选择【连接器-所有选项】依次修改下方几个关键位置。

附加选项	/IGNORE:4078 /safeseh:no
附加依赖项	ntoskrnl.lib;ndis.lib;Hal.lib;wdm.lib;wdmsec.lib;wmilib.lib
固定基址	此处需要清空
忽略所有默认库	是 (/NODEFAULTLIB)
启用增量链接	否 (/INCREMENTAL:NO)
驱动程序	驱动程序 (/Driver)
入口点	DriverEntry
生成清单	否 (/MANIFEST:NO)
生成调试信息	是 (/DEBUG)
生成映射文件	是 (/MAP)
数据执行保护	是 (/NXCOMPAT)
随机基址	此处需要清空
子系统	本机 (/SUBSYSTEM:NATIVE)

当如上文件配置完成后，最终效果如下图所示；



8.上方的配置已经基本完成了，接着我们编写一段驱动初始化代码，然后按下 **F7** 即可完成驱动的编译。

```
// 署名权
// right to sign one's name on a piece of work
// PowerBy: LyShark
// Email: me@lyshark.com

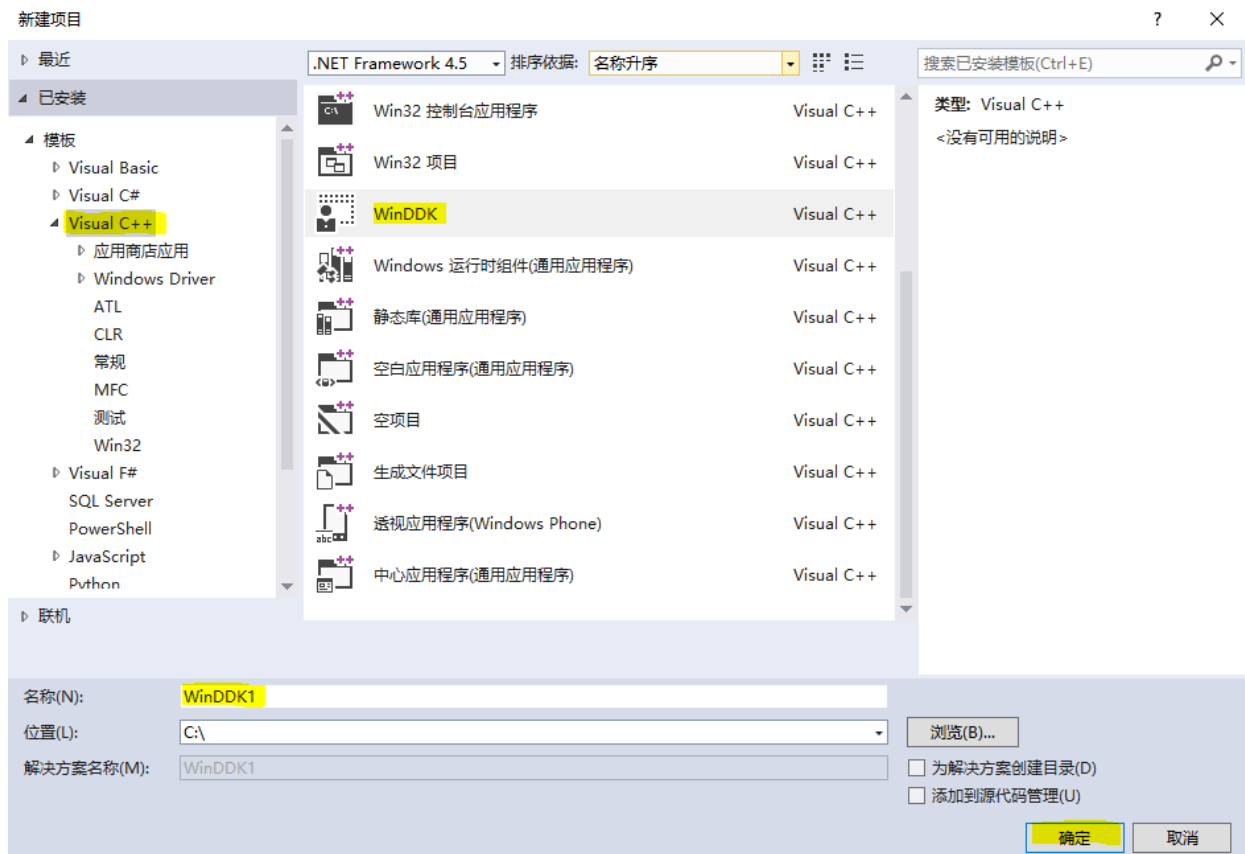
#include <ntifs.h>

// 卸载驱动
NTSTATUS UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("Uninstall Driver Is OK \n");
    return STATUS_SUCCESS;
}

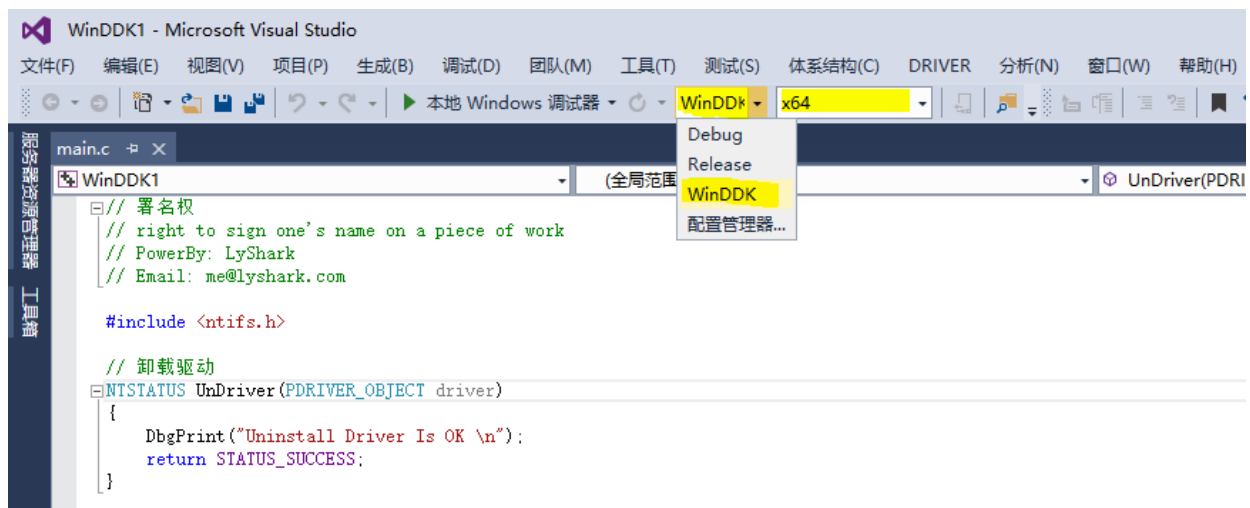
// 驱动入口地址
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("Hello LyShark \n");
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

9.最后生成一个驱动开发模板，依次选择【文件-导出模板-项目模板-下一步-完成】即可完成模板的导出，此时关闭VS工具并再次打开，就能直接使用我们的模板来开发驱动了，当用户需要使用时，不需要每次都配置。

- 模板位置：C:\Users\admin\Documents\Visual Studio 2013\My Exported Templates



读者也应注意，如果用户通过模板创建驱动开发项目则需要手动在配置菜单中切换到 WinDDK 选项的 x64 模式下。

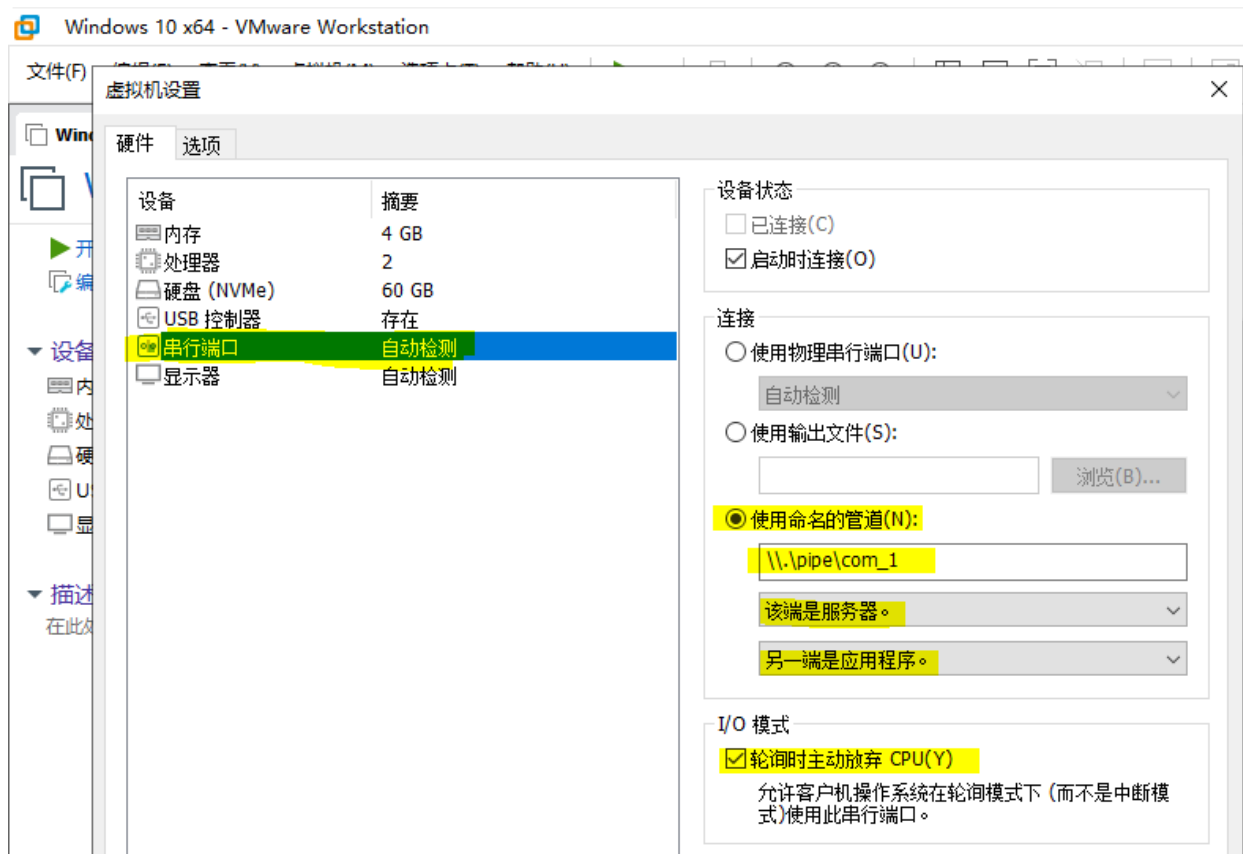


配置驱动双机调试

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，则只有 {current} 启动项以及一个 {bootmgr} 项。

```

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。

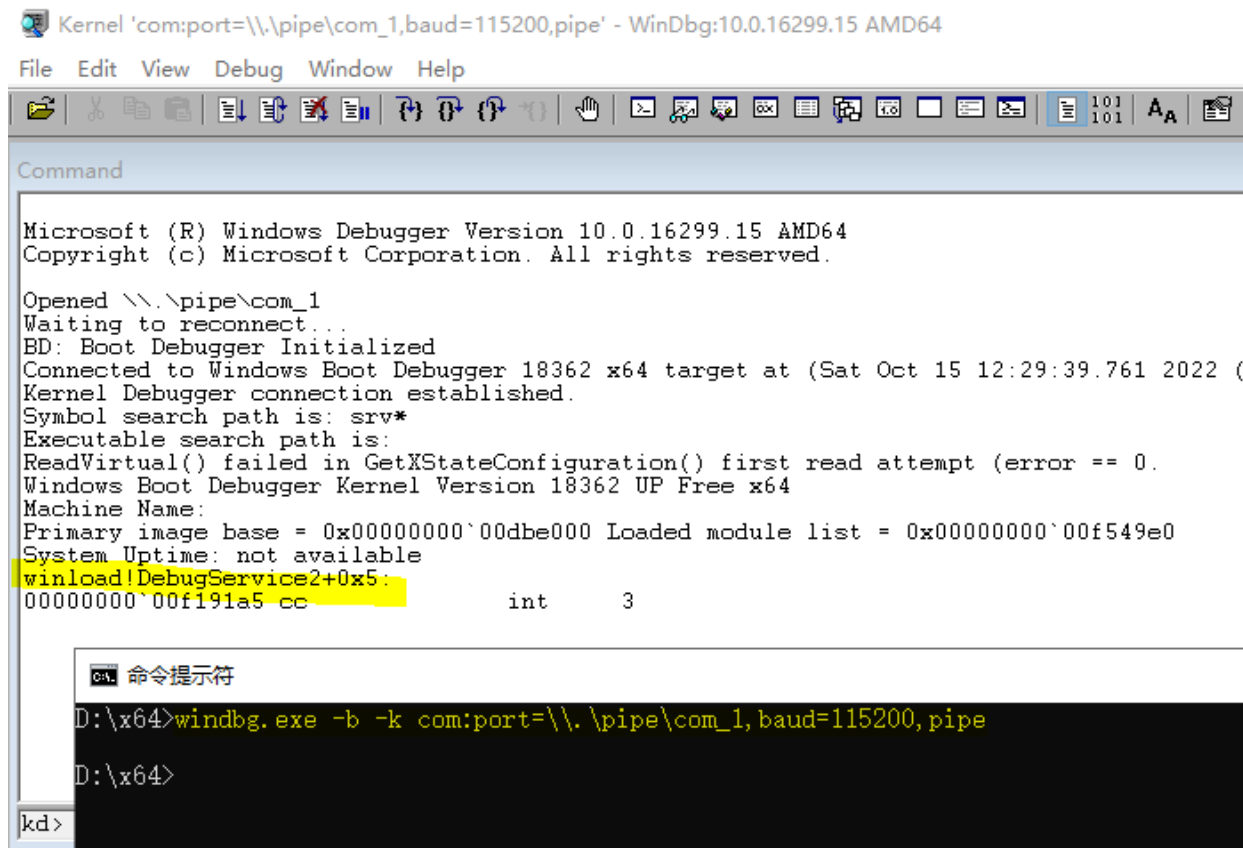
```
管理员: cmd
C:\Windows\system32>bcdedit /dbgsettings
debugtype          Serial
debugport          1
baudrate           115200
操作成功完成。
C:\Windows\system32>
```

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。

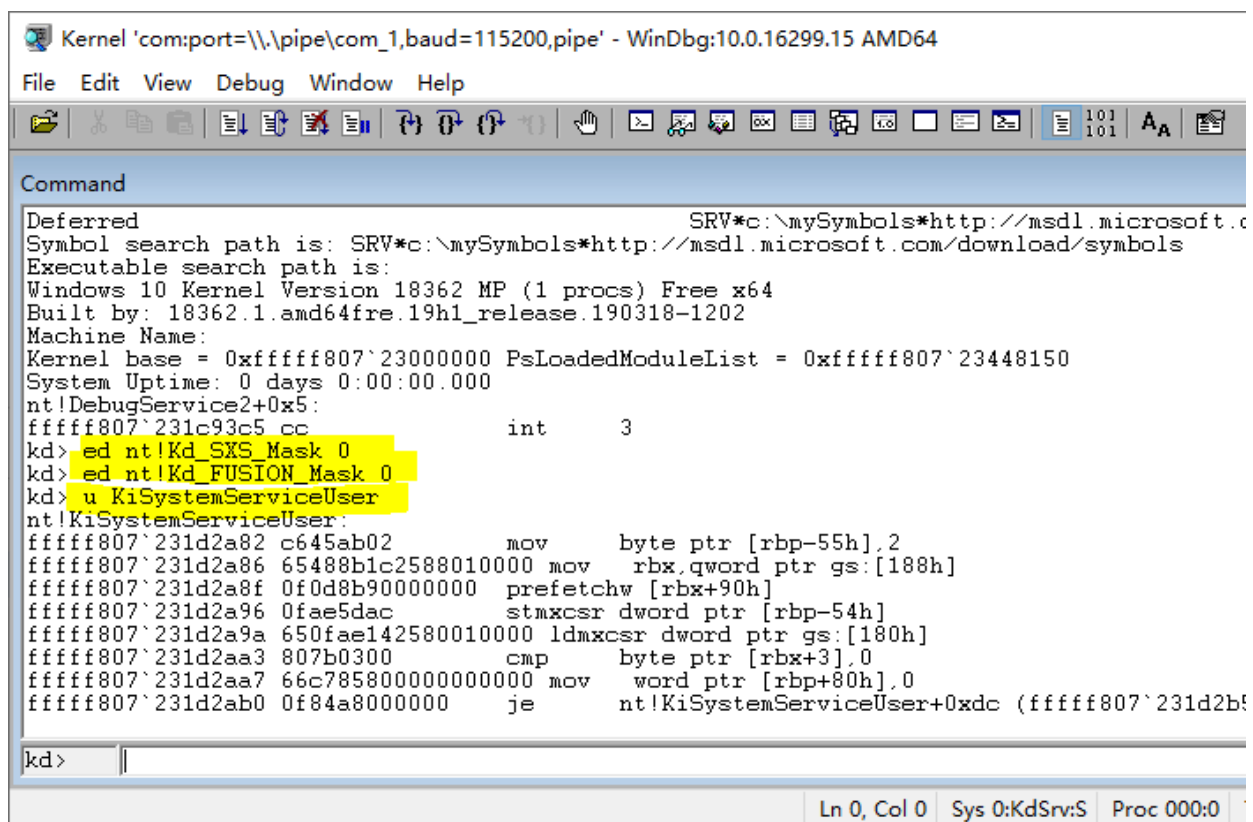
- 执行命令 `windbg.exe -b -k com:port=\\.\pipe\com_1,baud=115200,pipe` 如下图



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。



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。

```
// 署名权
// right to sign one's name on a piece of work
// PowerBy: LyShark
// Email: me@lyshark.com

#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;
    }

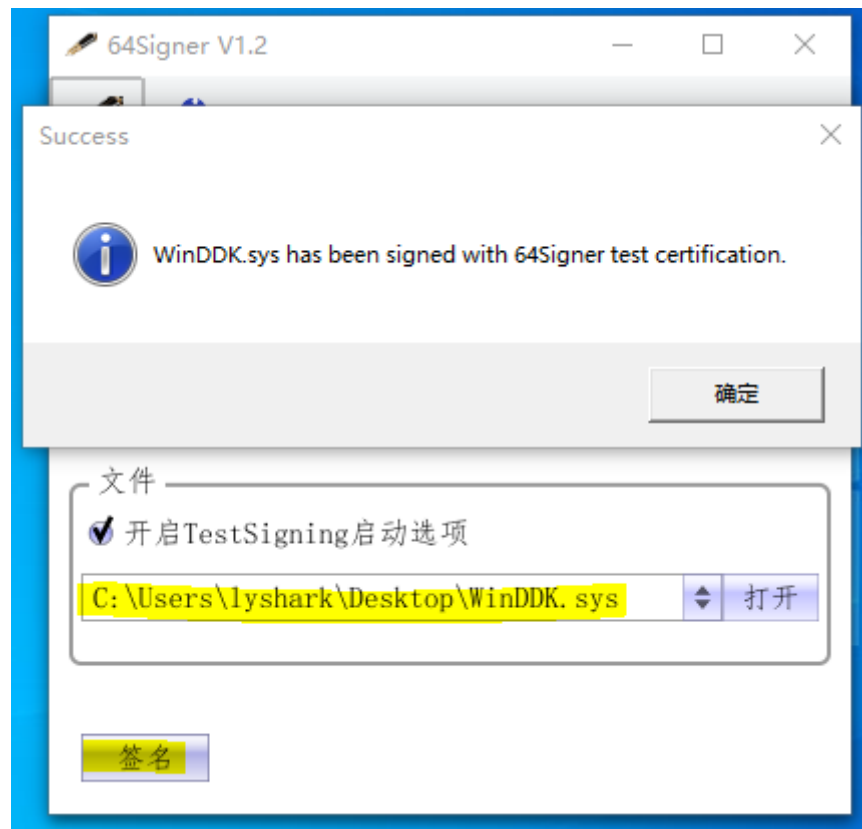
    // 设置断点
    DbgBreakPoint();
    // KdBreakPoint();
    // __debugbreak();

    DbgPrint("驱动已加载 \n");

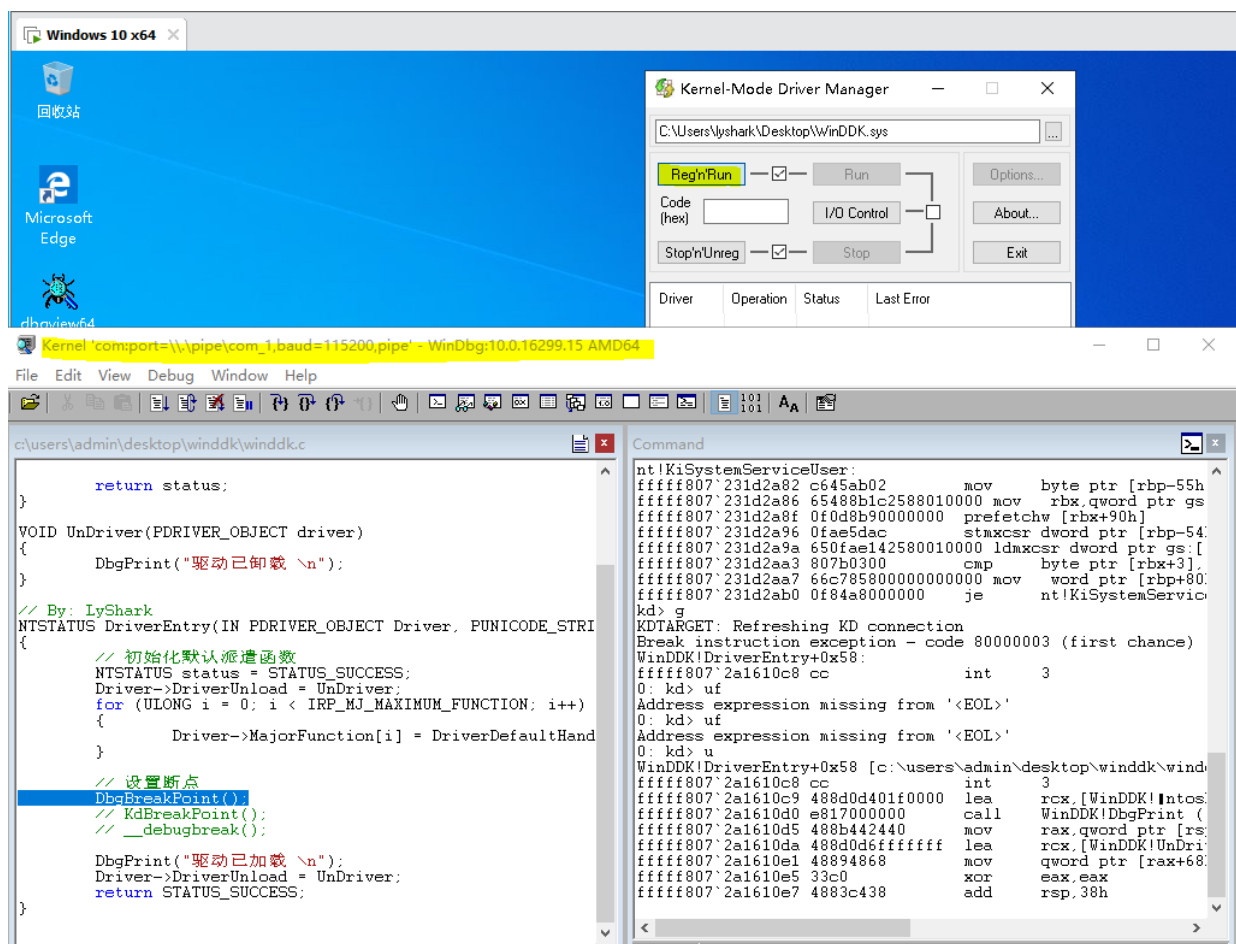
    // 驱动卸载函数
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}

```

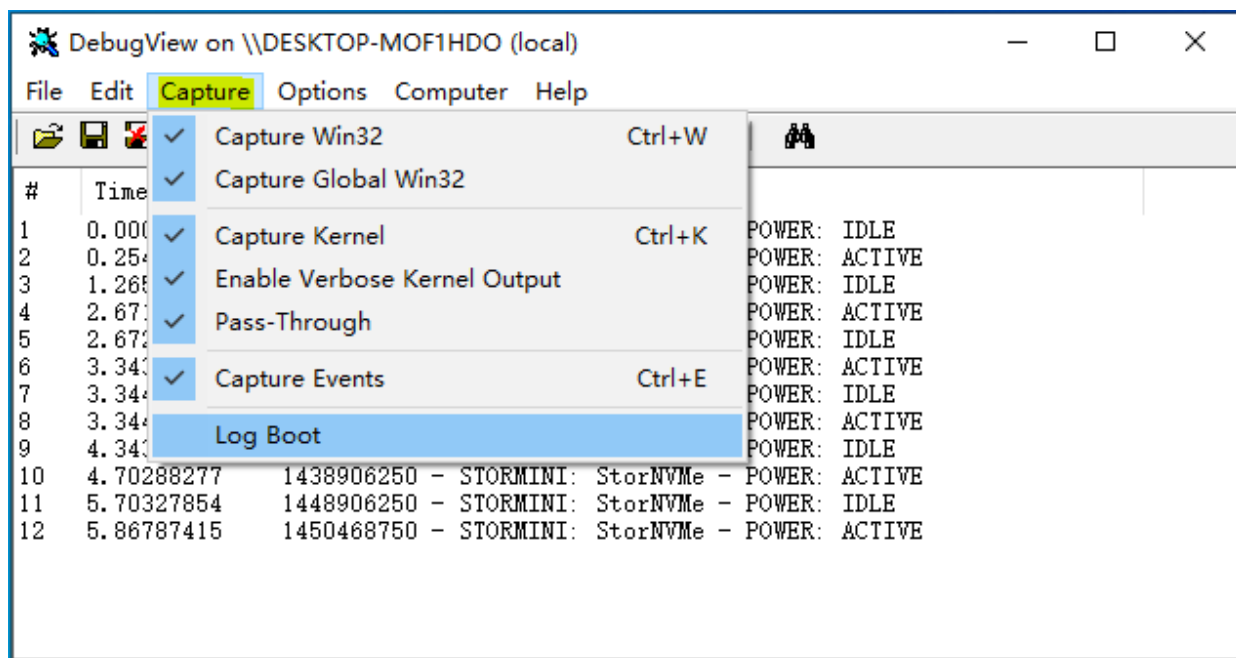
通过 visual studio 工具编译如上代码片段，并在 winDBG 中输入 g 命令 让系统运行起来，将编译好的驱动程序拖入到虚拟机中，并以管理员身份打开 windows 64Signer.exe，使用该工具对驱动程序进行签名，如下图所示；



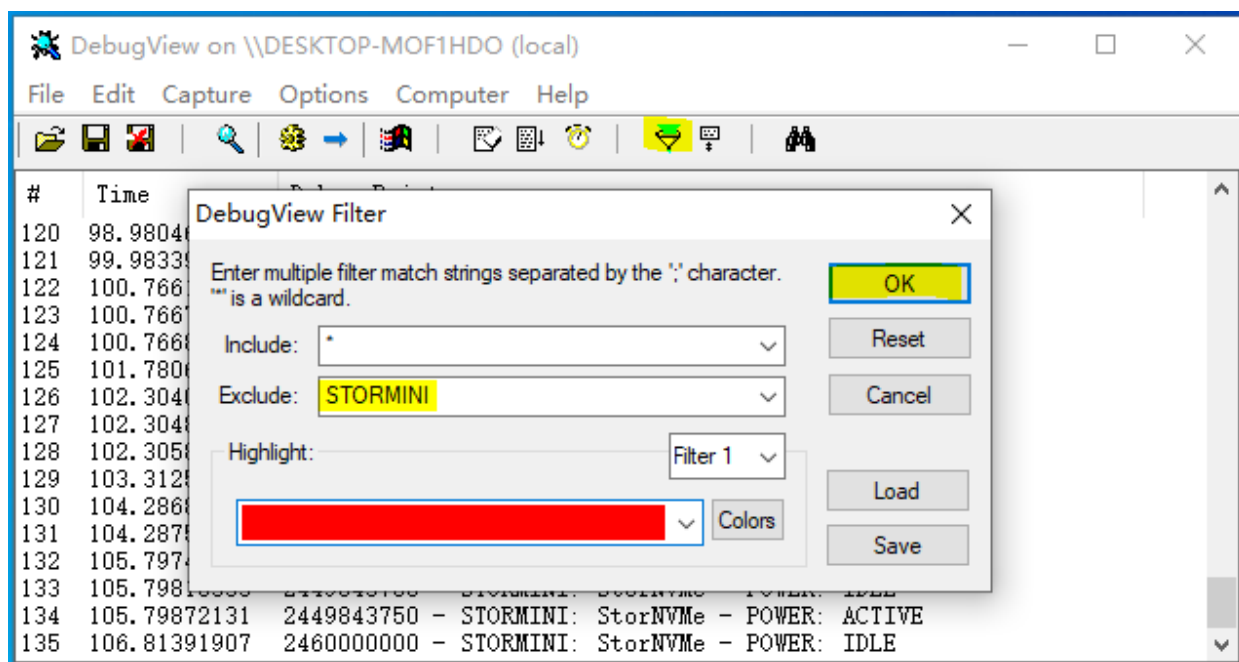
签名完成后将我们的驱动文件 winDDK.sys，拖入到 KmdManager.exe 驱动加载工具中，并通过驱动加载工具加载运行，此时 windows 系统会卡死，回到 winDBG 中发现已经可以进行调试了，如下图所示；



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 dbgvie64.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。