驱动开发学习笔记 1

很久没有网了，出了一段时间的差，近来，莫名的就有点郁闷！前不久在大富翁上发了一份帖子是关于 delphi 程序员的发展，大家的反应并不都是很好。于是开始觉得可以考虑换个方向。以前我是做 MIS 开发的。换哪个方向呢？人越多的方向，好像越是没有前途。想想当初上大学，那可是越多人考的学校，学费越贵啊！可现在的职业呢？越多人干的事，越是没有前途了。考虑来考虑去，决定学习一下驱动程序的开发吧！于是从网上查找了一些资料，看的懂的觉得蛮不错适合我这种小学生的就贴了出来，算是学习笔记吧！

用户模式与内核模式

从 Intel80386 开始，出于安全性和稳定性的考虑，该系列的 CPU 可以运行于 **ring0~ring3** 从高到低四个不同的权限级，对数据也提供相应的四个保护级别。运行于较低级别的代码不能随意调用高级别的代码和访问较高级别的数据，而且也只有运行在 ring0 层的代码可以直接对物理硬件进行访问。由于 WindowsNT 是一个支持多平台的操作系统，为了与其他平台兼容，它只利用了 CPU 的两个运行级别。一个被称为**内核模式**，对应 80x86 的 ring0 层，是操作系统的核心部分，设备驱动程序就是运行在该模式下；另一个被称为**用户模式**，对应 80x86 的 ring3 层，操作系统的用户接口部分（就是我们通常所说的 win32 API）以及所有的用户应用程序都运行在该级别。操作系统对运行在内核模式下的代码是不设防的，所以不管是建设还是破坏内核模式下的编程都是值得去研究的。

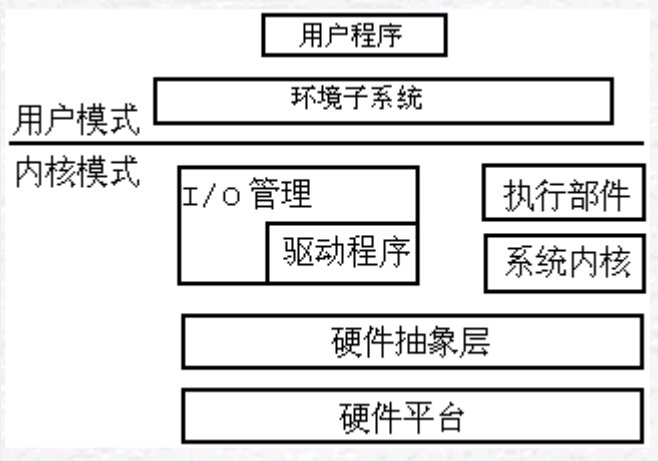

图 1-WIN2000 系统的分层结构

在物理硬件与系统核心之间有一个硬件抽象层（HardwareAbstractionLayer），它屏蔽了不同平台硬件的差异，向操作系统的上层提供了一套统一的接口。从图中我们还可以看到，设备驱动程序（DeviceDriver）是被 I/O 管理器(I/OManager)包围起来的，即驱动程序与操作系统上层的通信全部都要通过 I/O 管理器。这给驱动程序的编写带来了很大的便利，因为很多诸如接收用户的请求、与用户程序交换数据、内存映射、挂接中断、同步等等麻烦的工作都由 I/O 管理器代劳了。

驱动程序的分类

驱动程序并不像所有人想的那样一定要和硬件打交道，我粗略的把他分为两类：硬驱动和软驱动。硬驱动就是对硬件直接编程进行控制，这类驱动通常必须遵守硬件的通信协议，直接对硬件进行端口访问、中断响应、DMA 传输。它包括：串、并行口，键盘，文件系统，SCSI，网络等驱动程序；另外一种软驱动呢？不需要直接对硬件就行操作。我认为他可以理解为它是在硬驱动之上的一层更为高级的驱动。我想学习的主要是软驱动。

一般来说，设备驱动程序的任务主要有两个：第一，接受来自用户程序的读写请求，把用户的数据传送给设备，或把从设备接收到的数据传送给用户；第二，轮询设备或处理来自设备的中断请求，完成数据传输。

驱动程序的结构

在这里，我主要介绍 WDM 的结构。WDM(Windows driver module)是什么东西呢？在 Windows98\95 下面，也许你听得最多的是 VXD，我只知道 VXD 是一种驱动程序，和 WDM 差不多的东西。只是因为 Windows2000 是 WindowsNT 那条线过来的东西，要加上两个主要的新功能：即插即用（Plug and Play）和电源管理（Power Menage），又不能用 Windows98\95 那一套，所以就搞出一个叫 WDM 这么个东西，来支持 PNP 和 PM.。其实想想，现在的技术名词还不是一般的多啊！总之 wdm 大家都叫它 **windows 驱动程序模型**。

Windows2000 里有叫**即插即用管理器**和 **I\O（此 I\O 非彼 I\O 端口）管理器**的两个东西。比如说我在机器上插了一张符合 PCI 规范的 PCI 卡。即插即用管理器会发现这张卡插在第 N 个插槽上，然后即插即用管理器会说它找到了这样一张卡，它就去找有没有现成的驱动程序，如果没有找到，它会告诉我们，我找到了这样一张卡，请你插入这张卡的驱动程序盘。好，我们就把驱动程序盘给它，即插即用管理器会去找驱动程序盘上的 INF 文件，找到后它会比较 PCI 卡上的标志和 INF 文件里的标志是否相同，如果相同，它就会依照 INF 文件里提供的路径去找驱动程序，找到之后就可以交给 I\O 管理器，**I\O 管理器会装载这个驱动程序**。I\O 管理器在做了一些接口的工作后，即插即用管理器会先分配好相关的资源给 PCI 卡，比如说 I\O 端口空间、内存空间和中断向量，然后告诉这张卡的驱动程序，我给你分配了这些资源，你看怎么的。如果你没有怎么的或不敢怎么的，那就赶快记下这些资源，以备后用。

下面说 I\O 管理器这个东西。上面我们讲到 I\O 管理器装载这个驱动程序，驱动程序有一个大门，还有 N 多的小门。I\O 管理器先从大门进去（因为 I\O 管理器只找得到大门，I\O 管理器是不是很傻，NO，当然有它的道理，你别问我：I\O 管理器怎么找到大门的？驱动程序无非就是一些文件，I\O 管理器把这么些文件加载到系统中去），去找一样东西：进小门的地图。我们要在大门进去的房间里放这张地图（驱动程序都是我们造的，我们当然有驱动程序的地图啦）。I\O 管理器找到了地图，就可以自由进出大小门了。———这些大小门说白了就是函数（不要问我函数是什么东东），小门的地图就是函数的地址。I\O 管理器知道了这些函数的地址，当然就可以调用这些函数啦。还有一个叫 **IRP** 的东西，中文名叫 I\O 请求包。我们这样来理解它：在用户的应用程序这一端，要和驱动程序对话，它们之间不是简单的调用函数（至于为什么，我现在也不知道），应用程序和驱动程序之间有 I\O 管理器隔着，应用程序对驱动程序的操作，首先由 I\O 管理器处理成一个包，这个包里面有应用程序请求的操作内容、传送的数据等等一些东西，然后 I\O 管理器把这个包扔给驱动程序，驱动程序依照包里的请求，完成操作，把该回传的数据放进包里，再把包扔还给 I\O 管理器，I\O 管理器再把数据返

回给应用程序。——这里所说的包，就是 IRP。

这里说的只是 WDM 结构的一部分，但是有了这一部分知识，其它部分就不难懂了。通过上面的介绍你看见了什么。你可以想象得出驱动程序是什么样子的吗？我是这样想的。驱动程序好像就是一个函数库，只不过在大门的地方放了一张地图。而这个大门与地图我们见到过吗？好像有点像 dll 文件呢。在早些时候我学 dll 的时候，我就只会用 dll 来存放函数。

驱动程序开发——工具篇

因为我学习的时候是在 win2000 下进行的，所以一切以我学习时的配置为准。

第一：安装 win2000 操作系统，我安装是 win2000 高级服务器版本。

第二：安装 Vc++6.0，我装的是英文版。

第三：安装 win2000DDK；

通常驱动程序的调试都是用 ddk 在 cmd 中完成的。这部分我暂时略过。下面先介绍如何设置 vc++6.0 在 Visual Studio 6.0 集成环境中开发设备驱动程序的方法。

在 Windows 上，Windows DDK 提供的开发环境是基于命令行的，操作起来极为不便，而 Visual Studio 6.0 给我们提供了非常友好易用的集成环境，让我们有如虎添翼之感。

那么，能否利用 Visual Studio 的集成环境来开发驱动程序呢？答案是可以的。通过对 Visual Studio 集成环境的简单设置，创建好自己的驱动开发集成环境就可以了。

首先要求系统已安装 DDK 和 Visual C++6.0(安装时选上所有工具)，

1、接下来需要改造 ddk\bin\setenv.bat 把要求 mstools 的有关语句注释掉(若想在命令行环境开发驱动则还需加入 call VC_DIR\VC98\Bin\Vcvars32.bat)，以便能在命令行使用 vc 的相关工具；若只想在 IDE 环境开发就不必调用 Vcvars32.bat，因为相关工具的路径信息可以在 vc 环境中设置.)

2、创建一个目录 DriverEnv（目录名随意）,作为你开发驱动的大本营

3、在该目录下创建一个批处理文件 MakeDrvr.bat,内容如下：

```
@echo off
if "%1"=="" goto usage
if "%3"=="" goto usage
if not exist %1\bin\setenv.bat goto usage
call %1\bin\setenv %1 %4
%2

cd %3
build -b -w %5 %6 %7 %8 %9
```

```
goto exit

:usage
echo usage MakeDrvr DDK_dir Driver_Drive Driver_Dir free/check
ed [build_options]echo eg MakeDrvr %%DDKROOT%% C: %%WDMBOOK%%
free -cef
:exit
```

该批处理首先对传递的参数作一些检查，然后调用 ddk 的 setenv 命令设置环境变量，然后改变目录为源程序所在驱动器和目录，并最后调用 build，-b 保证显示完全的错误信息，-w 保证在屏幕上输出警告，在 vc ide 里的 output 窗口中可以看到这些错误和警告。

4.建立一个空白工程
选 File 的 new 菜单项,然后选 project 栏的 makefile,然后输入路径，一路 next 下去即可,visual studio 提供两种配置 win32 debug 和 win32 release.

5.修改这两种配置
    选 project 的 settings 菜单项 win32 debug:
    在 Build Command Line 一栏填入
MakeDrvr DDK_DIR SOURCE_DRIVE SOURCE_DIR checked [build option
s]
    在 Rebuild all options 一栏填入 -nmake /a
    在 output file 一栏填入与 sources 文件中的 TARGETNAME 相同的文件名
    在 Browse info file name 一栏填入 obj\i386\checked\(与
TARGETNAME 相同的文件名，见下述）.bsc

    win32 release:
    在 Build Command Line 一栏填入
MakeDrvr DDK_DIR SOURCE_DRIVE SOURCE_DIR free [build options]
    在 Rebuild all options 一栏填入 -nmake /a
    在 output file 一栏填入与 sources 文件中的 TARGETNAME 相同的文件名
    在 Browse info file name 一栏填入 obj\i386\free\(与 TARGETNAME 相同的文件名).bsc
    注：DDK_DIR 一般可以写成%BASEDIR%，build options 一般为-cef 即已足够

6.添加源文件到工程
    可以新建，也可以添加，这和普通的 win32 开发一样。

7.添加资源文件
    选 INSERT 的 RESOURCE 菜单项即可

8.把文件 makefile 放入源程序目录，其内容总是
#

```
# DO NOT EDIT THIS FILE!!! Edit .\sources. if you want to add
a new source
# file to this component. This file merely indirects to the re
al make file
# that is shared by all the driver components of the Windows N
T DDK
#

!INCLUDE $(NTMAKEENV)\makefile.def
```

9.把文件 Sources 放入源程序目录，内容为

```
    TARGETNAME=RamDrive//这是要生成的驱动程序.sys 文件的名字
    TARGETPATH=obj //.sys 文件所在目录的上层目录，（由于 ddk 的 bug)应手
工在 obj 目录下创建 checked 和 free 目录，以作为.sys 的最终存放目录
    TARGETTYPE=DRIVER //驱动程序的类型，一般不变
    INCLUDES=$(BASEDIR)\inc //ddk 包含文件路径，一般不变
    SOURCES=RamDrive.cpp RamDrive.rc //源文件(不要头文件)，资源文件
    BROWSER_INFO = 1 //若想要浏览信息，则要有本行；否则可无
```

10.因为 MakeDrvr.bat 在 DriverEnv 目录，所以应把该目录添加到 vc 的 Executable files 里面

选 tools 的 options 菜单项，然后选 directories 页,在 show directories for 一栏选择 Executable files，然后添加即可.

至此，环境设置完毕，你可以按 F7, build 你的驱动程序了。

驱动程序开发——Hello Word!

看了好多天的书！特别到书店买了《Windows 200/xp wdm 设备驱动开发》这本书，在这里我不想怎么评论它！对于高手来说，我觉得她一定不能满足，但是对于像我这样想入门的人来说，仿佛看了半天，还是不知道从何下手。什么原理、模型、分层等等讲不讲，讲！绝对应该讲！但是你得快点告诉我怎么先弄一个像“Hello Word!”的什么简单来不能再简单的完整的例子给我呀！到网上找阿找啊！那些高手啊！也不为我们新手写点图文并茂的上手资料。没办法！结合自己的需要再参考一些别人的东东，算是自己的一点不成熟的想法吧！

我觉得下面这个介绍非常不错！我能看懂，所以贴了出来。

我道为什么找不到“Hello Word!”呢？原来在驱动开发的例子里是没有所谓的“Hello World”程序的。这主要还是因为网络上的 WDM 资料太少造成的。但是程序的入口点呢？c 语言有 Main()，用 Vc 的常看见的是 WinMain()，Delphi 开发的是 Program 里的 Begin，但是驱动开发呢？那也是应该有程序的入口点啊。后来我才明白了，那就是 DriverEntry()函数。还有一个问题让我怀疑了老半天，那就是驱动开发的源程序中

需不需要 include 头文件呀？为什么会怀疑呢？那是因为我看了半天的书都没有看到一个完整的驱动程序结构。真的是郁闷。下面是我看到的一个完整的结构，我先放上来，让大家看看驱动开发的结构吧。

```
/*************************************************************
*****
程序名称：Hello World for WDM
文件名称：HelloWDM.cpp
日期：2002-8-16
*************************************************************
****/

//一定要的头文件，声明了函数模块和变量：
#include "HelloWDM.h"

/*************************************************************
*****
函数名称：DriverEntry()
功能描述：WDM 程序入口（原来的 WinMain 被换成了 DriverEntry，也是驱动程序的大门）
*************************************************************
****/
//extern "C"是必须的，表示"用 C 链接"。如果你的文件名是 HelloWDM.c 的话，这句可以省略。
extern "C"
NTSTATUS DriverEntry(    IN PDRIVER_OBJECT DriverObject, //IN 是一个关键字表示这是一个输 入参数，PDRIVER_OBJECT 是一个数据结构的指针，就像 PCHAR 一样，这个数据结构是什么样子的，后面我会列出来。她描述了一个驱动设备对象。
                         IN PUNICODE_STRING RegistryPath)//参数 RegistryPath 指定了驱动程序注册表健的路径，因为驱动程序安装后总会在系统注册表里留下一点东西的。
{
   //指定"添加设备"消息由函数"HelloWDMAddDevice()"来处理：
   DriverObject->DriverExtension->AddDevice = HelloWDMAddDevice;
   //指定"即插即用"消息由函数"HelloWDMPnp()"来处理：
   DriverObject->MajorFunction[IRP_MJ_PNP] = HelloWDMPnp;

   //返回一个 NTSTATUS 值 STATUS_SUCCESS。几乎所有的驱动程序例程都必须返回一个 NTSTATUS 值，这些值在 NTSTATUS.H DDK 头文件中有详细的定义。
   return STATUS_SUCCESS;
}

//NTSTATUS 也是一个数据类型，上面我所说的消息有点不准确的，准确地说是"I/O 请求包"，不过如果像我们以前理解消息那样来理解也无不可，我觉得两者太想了。无
```

非就是上层的应用程序通过它来告诉驱动程序，你要给我什么服务吧！IRP_MJ_PNP 就是即插即用处理的请求。你发没发觉上面其实是在制造进入各个房间的“小门”

```
/*************************************************************
*****
函数名称：HelloWDMAddDevice()
功能描述：处理“添加设备”消息
*************************************************************
****/
NTSTATUS HelloWDMAddDevice(IN PDRIVER_OBJECT DriverObject,
                           IN PDEVICE_OBJECT PhysicalDeviceObject)
{
    //定义一个 NTSTATUS 类型的返回值：
    NTSTATUS status;
    //定义一个功能设备对象（Functional Device Object）：
    PDEVICE_OBJECT fdo;

    //创建我们的功能设备对象，并储存到 fdo 中：
    status = IoCreateDevice(
        DriverObject,               //驱动程序对象
        sizeof(DEVICE_EXTENSION),   //要求的设备扩展的大小
        NULL,                       //设备名称，这里为 NULL
        FILE_DEVICE_UNKNOWN,        //设备的类型，在标准头文件 WDM.H 或
NTDDK.H 中列出的 FILE_DEVICE_xxx 值之一
        0,                          //各种常量用 OR 组合在一起，指示可删除介质、只读
等。
        FALSE,                      //如果一次只有一个线程可以访问该设备，为 TRUE，
否则为 FALSE
        &fdo);                      //返回的设备对象

    //NT_SUCCESS 宏用于测试 IoCreateDevice 内核是否成功完成。不要忘记检查
对内核的所有调用是否成功。NT_ERROR 宏不等同于!NT_SUCCESS，最好使
用!NT_SUCCESS，因为除了错误外，它还截获警告信息。
    if( !NT_SUCCESS(status))
        return status;

    //创建一个设备扩展对象 dx，用于存储指向 fdo 的指针：
    PDEVICE_EXTENSION dx =
(PDEVICE_EXTENSION)fdo->DeviceExtension;
    dx->fdo = fdo;

    //用 IoAttachDeviceToDeviceStack 函数把 HelloWDM 设备挂接到设备栈：
    dx->NextStackDevice = IoAttachDeviceToDeviceStack(fdo,
PhysicalDeviceObject);
```

```
    //设置 fdo 的 flags。有两个“位”是必须改变的，一个是必须清除
DO_DEVICE_INITIALIZING 标志，如果在 DriverEntry 例程中调用
IoCreateDevice()，就不需要清除这个标志位。还有一个是必须设置 DO_BUFFER_IO
标志位：
    fdo->Flags |= DO_BUFFERED_IO | DO_POWER_PAGABLE;
    fdo->Flags &= ~DO_DEVICE_INITIALIZING;

    //返回值：
    return STATUS_SUCCESS;
}


/**********************************************************************
函数名称：HelloWDMPnp()
功能描述：处理“即插即用”消息
**********************************************************************/
NTSTATUS HelloWDMPnp(IN PDEVICE_OBJECT fdo,
                     IN PIRP Irp)
{
    //创建一个设备扩展对象 dx，用于存储指向 fdo 的指针：
    PDEVICE_EXTENSION dx=(PDEVICE_EXTENSION)fdo->DeviceExtension;

    //首先要通过函数 IoGetCurrentIrpStackLocation()得到当前的 IRP，并由此得
到 Minor Function：
    PIO_STACK_LOCATION IrpStack = IoGetCurrentIrpStackLocation(Irp);
    ULONG MinorFunction = IrpStack->MinorFunction;

    //然后把这个 Minor Function 传递给下一个设备栈：
    IoSkipCurrentIrpStackLocation(Irp);
    NTSTATUS status = IoCallDriver( dx->NextStackDevice, Irp);

    //处理“即插即用”次功能代码：
    //当 Minor Function 等于 IRP_MN_REMOVE_DEVICE 时，说明有设备被拔出或
卸下，这时要取消资源分配并删除设备：
    if( MinorFunction==IRP_MN_REMOVE_DEVICE)
    {
        //取消设备接口：
        IoSetDeviceInterfaceState(&dx->ifSymLinkName, FALSE);
        RtlFreeUnicodeString(&dx->ifSymLinkName);

        //调用 IoDetachDevice()把 fdo 从设备栈中脱开：
```

```
        if (dx->NextStackDevice)
            IoDetachDevice(dx->NextStackDevice);
        //删除 fdo：
        IoDeleteDevice(fdo);
    }

    //返回值：
    return status;
}



/**********************************************************************
*****
程序名称：Hello World for WDM
文件名称：HelloWDM.h
作者：罗聪
日期：2002-8-16
**********************************************************************
****/

//头文件，只是声明一些函数和变量，比较简单就不多说了，请读者自行研究：

#ifdef __cplusplus

extern "C"
{
#endif

#include "ntddk.h"

#ifdef __cplusplus
}
#endif

typedef struct _DEVICE_EXTENSION
{
    PDEVICE_OBJECT    fdo;
    PDEVICE_OBJECT    NextStackDevice;
    UNICODE_STRING    ifSymLinkName;

} DEVICE_EXTENSION, *PDEVICE_EXTENSION;

NTSTATUS HelloWDMAddDevice(IN PDRIVER_OBJECT DriverObject,
```

```
                    IN PDEVICE_OBJECT PhysicalDeviceObject);

NTSTATUS HelloWDMPnp(IN PDEVICE_OBJECT fdo,
                     IN PIRP Irp);
```

好了，第一个 WDM 版的“Hello World”就介绍到这里，虽然实际上它什么都没有做，但是由于它包含了完整的框架，所以对于向我这样的新手来说还是很有参考价值的。至于怎么编译及安装只有下次再说了

2005 年 9 月 25 日 15:32

驱动程序开发——编译前传

好啦，辛辛苦苦终于写完了程序，让我们编译运行吧！按下 Ctrl+F5（嘿嘿，让我们先假设你习惯用 VC 来写程序），我等啊等……疑？怎么毫无动静的？再看看 Output 窗口，哇！有几百个错误啊！！不禁头大——这是怎么回事呢？

原来，WDM 程序编译出来的并不是我们常见的.exe，而是.sys 文件，在未经设置编译环境之前，是不能直接用 VC 来编译的（这就是为什么会有几百个错误了）。这种类型的文件你可以在 WINNT\System32\Drivers 里面找到很多。其实驱动程序也是一种 PE 文件，它同样由 DOS MZ header 开头，也有完整的 DOS stub 和 PE header，同样拥有 Import table 和 Export table——……那跟普通的 PE 文件有什么不一样呢？那么就让我们先来做个小剖析，加深对.sys 文件的认识吧

首先祭出 Delphi 里附带的 tdump.exe 程序。让我们键入：
C:\WINNT\System32\Drivers>tdump ccport.sys -em -ee
参数-em 是列出 Import table，-ee 是列出 Export table。回车之后，屏幕列出一大堆东西：

```
C:\WINNT\SYSTEM32\DRIVERS>tdump ccport.sys -em -ee
Turbo Dump  Version 5.0.16.12 Copyright ? 1988, 2000 Inprise Corporation
                  Display of File CCPORT.SYS

IMPORT:     NTOSKRNL.EXE={hint:011Fh}.'memcpy'
IMPORT:     NTOSKRNL.EXE={hint:003Dh}.'IoDeleteDevice'
IMPORT:     NTOSKRNL.EXE={hint:0030h}.'IoAttachDeviceToDeviceStack'
IMPORT:     NTOSKRNL.EXE={hint:008Eh}.'KeSetEvent'
IMPORT:     NTOSKRNL.EXE={hint:0068h}.'IofCallDriver'
IMPORT:     NTOSKRNL.EXE={hint:0095h}.'KeWaitForSingleObject'
IMPORT:     NTOSKRNL.EXE={hint:0074h}.'KeInitializeEvent'
IMPORT:     NTOSKRNL.EXE={hint:003Fh}.'IoDetachDevice'
```

```
IMPORT:     NTOSKRNL.EXE={hint:00D3h}.'RtlFreeUnicodeString'
IMPORT:     NTOSKRNL.EXE={hint:0077h}.'KeInitializeSpinLock'
IMPORT:     NTOSKRNL.EXE={hint:0129h}.'strcpy'
IMPORT:     NTOSKRNL.EXE={hint:0121h}.'memset'
IMPORT:     NTOSKRNL.EXE={hint:003Ch}.'IoCreateUnprotectedSymbolic
Link'
IMPORT:     NTOSKRNL.EXE={hint:0038h}.'IoCreateDevice'
IMPORT:     NTOSKRNL.EXE={hint:00C2h}.'RtlAnsiStringToUnicodeString'
IMPORT:     NTOSKRNL.EXE={hint:0069h}.'IofCompleteRequest'
IMPORT:     NTOSKRNL.EXE={hint:0124h}.'sprintf'
IMPORT:     NTOSKRNL.EXE={hint:003Eh}.'IoDeleteSymbolicLink'
IMPORT:     NTOSKRNL.EXE={hint:0042h}.'IoFreeIrp'
IMPORT:     NTOSKRNL.EXE={hint:004Dh}.'IoInitializeIrp'
IMPORT:     NTOSKRNL.EXE={hint:002Dh}.'IoAllocateIrp'
IMPORT:     NTOSKRNL.EXE={hint:0027h}.'InterlockedExchange'
IMPORT:     NTOSKRNL.EXE={hint:0025h}.'InterlockedCompareExchange
'
IMPORT:     NTOSKRNL.EXE={hint:0035h}.'IoCancelIrp'
IMPORT:     NTOSKRNL.EXE={hint:012Ah}.'strlen'
IMPORT:     NTOSKRNL.EXE={hint:0126h}.'strcat'
IMPORT:     NTOSKRNL.EXE={hint:0114h}.'atoi'
IMPORT:     NTOSKRNL.EXE={hint:0128h}.'strcmp'
IMPORT:     NTOSKRNL.EXE={hint:0034h}.'IoBuildSynchronousFsdReque
st'
IMPORT:     NTOSKRNL.EXE={hint:00D5h}.'RtlInitAnsiString'
IMPORT:        HAL.DLL={hint:0006h}.'KfAcquireSpinLock'
IMPORT:        HAL.DLL={hint:0009h}.'KfReleaseSpinLock'

EXPORT ord:0001='Vcomm_DriverControl'
```

看到了吗？它主要调用了 NTOSKRNL.EXE 和 HAL.DLL 文件（实际上你会发现，几乎所有的 WDM 驱动程序都会调用 NTOSKRNL.EXE 文件，从它的名字你可以看出为什么了吧？），并且输出了一个函数"Vcomm_DriverControl"。这表明，其实.sys 跟.exe 文件一样，都是一种 PE 文件来的。不同的是，.sys 文件 Import 的通常是 NTOSKRNL.EXE，而.exe 文件 Import 的通常是 KERNEL32.DLL 和 USER32.DLL。

知道了这些有什么用呢？实际上，由于.sys 通常不调用 KERNEL32.DLL 和 USER32.DLL，所以你是不能在设备驱动程序里面调用任何 C、C++和 Win32 函数的，而且也不能用 C++关键字 new 和 delete 等（可以用 malloc 和 free 来代替），而必须使用大量的内核函数。另外，你应该也能看到她调用了像 IoDeleteDevice、IoAttachDeviceToDeviceStack 等等函数，这些你以前可能没有见过的函数都是些内核函数。为了读者的方便，下面我列出一些常见的驱动程序可用的内核函数：

```
Ex…        执行支持
Hal…       硬件抽象层（仅 NT/Windows 2000）
Io…        I/O 管理器（包括即插即用函数）
Ke…        内核
Ks…        内核流 IRP 管理函数
Mm…        内存管理器
Ob…        对象管理器
Po…        电源管理
Ps…        进程结构
Rtl…       运行时库
Se…        安全引用监视
Zw…        其他函数
```

最后让我们再来看看，写设备驱动程序时必须注意的一些问题：

1、内核宏

如果查看 DDK 头文件，会发现有几个内核函数是以宏的方式实现的。这种宏中有几个宏的定义是相当糟糕的。例如，我们看到 RemoveHeadList 的定义如下：

```
#define RemoveHeadList(ListHead)
        (ListHead)->Flink;
        {RemoveEntryList((ListHead)->Flink)}
```

如果以以下方式调用 RemoveHeadList，则将编译错误的代码：

```
if(SomethingInList)
        Entry = RemoveHeadList(list);
```

使这个调用安全的唯一方法是使用花括号：

```
if(SomethingInList)
   {
      Entry = RemoveHeadList(list);
   }
```

所以我们切勿为了贪图一时的方便，而使用不太规范的写法，最好是在所有的 if、for 和 while 等语句中使用花括号。

2、驱动程序函数名称

跟 C/C++的 main()函数一样，设备驱动程序也有一个必须存在，而且只能以 DriverEntry()为名称的入口函数。然而，除此之外，我们可以使用任何名字来给其他函数命名——只要你自己记得就行了，当然，最好符合某些特定的规范啦，例如匈牙利命名法……

3、安装时的问题

·在 Windows98 中驱动程序可执行文件必须是 8.3 文件名。（别问我为什么，我也不知道，我只能建议你去问比尔该死）

·如果 INF 文件中含有非法节的详细资料，Windows 将不使用这个 INF 文件。

本节罗罗嗦嗦讲了一大堆，跟实际的编程却并没有太大的关系，前传嘛！就是这样的啦！

驱动开发——编译正传

我在前面也讲过了一些关于编译环境及工具的。在这里结合本例子我再说一下：

DDK 分为 98 DDK 和 2000 DDK 两种，它们工作起来是大同小异的，不过有些驱动程序只能在 2000 DDK 中使用。由于 Win98 注定是一种即将被淘汰的操作系统了，所以我学习的时候也没有过多的关注，我用的是 2000 的 DDK，所以以下的所有内容都是针对 2000 DDK 的。

·准备工作

1、确定你已经安装了 Visual C++

2、安装 2000 DDK

3、安装 2000 DDK 成功后，在"**开始**"->"**程序**"里应该有"**Development Kits**"->"**Windows 2000 DDK**"的项目。

（注意一定要先安装好 VC，然后才安装 DDK，这个顺序决不能颠倒！！）

4、保证 DDKROOT 环境变量设置为 Windows 2000 DDK 的基目录，如果不是的话，请在控制面板"**系统**"属性的"**高级**"标签环境变量编辑器中设置好这个环境变量。

·编写必需的文件

编译 WDM 程序的时候，有两个文件是必须要有的，它们是：

1、**makefile**

（这个是什么啊？你可能会问。）对于比较年轻的程序员来说，有可能没有见过这个文件吧。其实在 VC 这些 IDE 出现之前，我们都必须使用 makefile 来确定项目中哪些文件需要重新编译，现在的 IDE 都把这个工作自动做好了

我们要做的工作很简单，就是提供这样一个文件，它的内容是：

```
#
# DO NOT EDIT THIS FILE!!!  Edit .\sources. If you want to add a new source
# file to this component.  This file merely indirects to the real make file
# that is shared by all the driver components of the Windows NT DDK
#

!INCLUDE $(NTMAKEENV)\makefile.def
```

正如它所述，不要编辑这个文件。事实上每个 WDM 程序所需要的 makefile 的内容都是一样的，也就是说，我们只需要简单地 copy 一个 makefile 到新的项目中就可以了

2、**Sources**

```
TARGETNAME=HelloWDM //编译出来的驱动程序的名称
TARGETTYPE=DRIVER       //编译的类型是驱动程序编译
DRIVERTYPE=WDM          //驱动程序的类型是 WDM 驱动程序
TARGETPATH=OBJ          //生成的文件存放在 OBJ 目录中

INCLUDES=$(BASEDIR)\inc;\   //这是需要引入的头文件
         $(BASEDIR)\inc\ddk;\

TARGETLIBS=$(BASEDIR)\lib\*\free\usbd.lib\  //这是需要引入的库文件

SOURCES=HelloWDM.cpp\    //这是源码文件
```

这个文件指定了驱动程序目标名是 HelloWDM.sys，是一个 WDM 驱动程序，生成的文件存放在 OBJ 目录中。值得注意的是，“=”前后不能有空格，否则编译的时候会出错。

·开始编译

娃哈哈，前面罗罗嗦嗦讲了一大堆，现在终于到重点了。WDM 程序的编译过程比较特殊，它不是在 VC 里面按 F7 来编译的（尽管你可以通过设置来达到这一目的），而是通过一个 DDK 实用工具 build 来完成。下面我们来讲讲具体步骤：

1、“Debug”版的生成

首先，我们假设你的源代码放在 D:\HelloWDM 里面。请跟着以下步骤：

“开始”->“程序”->“Development Kits”->“Windows 2000 DDK”->“Checked Build Environment”

屏幕将显示：（有“回车”的那行是需要读者你亲自打进去的）

```
New or updated MSVC detected.  Updating DDK environment….

Setting environment for using Microsoft Visual C++ tools.
Starting dirs creation…Completed.

D:\NTDDK>cd\HelloWDM    （回车）

D:\HelloWDM>build    （回车）
```

如果源代码没有错误的话，生成的 HelloWDM.sys 将存放在 objchk\i386 目录中。

2、“Release”版的生成

请跟着以下步骤：

“开始”->“程序”->“Development Kits”->“Windows 2000 DDK”->“Free Build Environment”

随后的步骤跟“Debug”版相同，不同的是生成的HelloWDM.sys将存放在objfre\i386目录中。

2005 年 9 月 25 日 16:45

评论

 # 驱动开发——编译正传 2006-2-23 12:55 CnYouth

驱动开发——编译正传

 # 回复：驱动开发——编译正传 2006-4-12 14:15 微笑刺客

BUILD: Computing Include file dependencies:
BUILD: Examining c:\helloworld directory for files to compile.
Compiling c:\helloworld directory **********
'nmake.exe /c BUILDMSG=Stop. -i NTTEST= UMTEST= NOLINK=1 NOPASS0=1 386=1'
.\sources.(5) : U1033: syntax error : '$(BASEDIR)\inc\ddk' unexpected
Stop.
BUILD: nmake.exe failed - rc = 2
Linking c:\helloworld directory **********
'nmake.exe /c BUILDMSG=Stop. -i LINKONLY=1 NOPASS0=1 NTTEST= UMTEST= 386=1'
.\sources.(5) : U1033: syntax error : '$(BASEDIR)\inc\ddk' unexpected
Stop.
BUILD: nmake.exe failed - rc = 2

我的运行结果是这样的，斑竹看看我哪里设置的有问题吗？
谢谢帮忙！！！

驱动开发——安装

作为一个完整的例子，你开发出来驱动还必须要能安装。所以下面我讲一下安装。

如果前面的编译过程没有错误的话，现在我们应该已经得到了一个 HelloWDM.sys 文件，假设它是放在 D:\HelloWDM\objfre\i386 中。

安装 WDM 驱动程序可以用两种方法，一种是利用注册表，还有一种是利用 INF 文件。我们一般是采用 INF 文件（这是微软推荐的）。INF 文件可以在 WINNT\INF 目录中找到很多。为了顺利安装，我在这里先给出 HelloWDM 所需要的 HelloWDM.INF 文件：

```
;; The Win2K DDK documentation contains an excellent INF reference.

;--------- Version Section ---------------------------------------------------

[Version]
Signature="$CHICAGO$"
Provider=LC_Device
DriverVer=8/21/2002,3.0.0.3

; If device fits one of the standard classes, use the name and GUID here,
; otherwise create your own device class and GUID as this example shows.

Class=Unknown
ClassGUID={ff646f80-8def-11d2-9449-00105a075f6b}

;--------- SourceDiskNames and SourceDiskFiles Section
----------------------

; These sections identify source disks and files for installation. They are
; shown here as an example, but commented out.

[SourceDisksNames]
1 = "HelloWDM",Disk1,,

[SourceDisksFiles]
HelloWDM.sys = 1,objfre\i386,

;--------- ClassInstall/ClassInstall32 Section -------------------------------

; Not necessary if using a standard class

; 9X Style
[ClassInstall]
Addreg=Class_AddReg

; NT Style
[ClassInstall32]
Addreg=Class_AddReg

[Class_AddReg]
HKR,,,,%DeviceClassName%
HKR,,Icon,,"-5"

;--------- DestinationDirs Section -------------------------------------------
```

```
[DestinationDirs]
YouMark_Files_Driver = 10,System32\Drivers

;--------- Manufacturer and Models Sections -----------------------------------

[Manufacturer]
%MfgName%=Mfg0

[Mfg0]

; PCI hardware Ids use the form
; PCI\VEN_aaaa&DEV_bbbb&SUBSYS_cccccccc&REV_dd
;改成你自己的 ID
%DeviceDesc%=YouMark_DDI, PCI\VEN_9999&DEV_9999

;---------- DDInstall Sections ---------------------------------------------
; --------- Windows 9X -----------------

; Experimentation has shown that DDInstall root names greater than 19
characters
; cause problems in Windows 98

[YouMark_DDI]
CopyFiles=YouMark_Files_Driver
AddReg=YouMark_9X_AddReg

[YouMark_9X_AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,HelloWDM.sys
HKR, "Parameters", "BreakOnEntry", 0x00010001, 0

; --------- Windows NT -----------------

[YouMark_DDI.NT]
CopyFiles=YouMark_Files_Driver
AddReg=YouMark_NT_AddReg

[YouMark_DDI.NT.Services]
Addservice = HelloWDM, 0x00000002, YouMark_AddService

[YouMark_AddService]
DisplayName = %SvcDesc%
ServiceType = 1 ; SERVICE_KERNEL_DRIVER
```

```
StartType = 3 ; SERVICE_DEMAND_START
ErrorControl = 1 ; SERVICE_ERROR_NORMAL
ServiceBinary = %10%\System32\Drivers\HelloWDM.sys

[YouMark_NT_AddReg]
HKLM, "System\CurrentControlSet\Services\HelloWDM\Parameters",\
"BreakOnEntry", 0x00010001, 0


; --------- Files (common) -------------

[YouMark_Files_Driver]
HelloWDM.sys

;--------- Strings Section ------------------------------------------------

[Strings]
ProviderName="Flying L Co.,Ltd."
MfgName="LC Soft"
DeviceDesc="Hello World WDM!"
DeviceClassName="LC_Device"
SvcDesc="???"
```

注意它可以同时在 Win98 或者 Win2000 中使用（系统会通过这个 INF 文件里面的字段名称，自动选择适合当前系统的安装方法的）。关于 INF 文件的各个字段含义现在我也不知道，所以也没有办法说清楚，如果谁看到这篇文章，而又知道的话，不妨为我一份。

准备好这个 HelloWDM.INF 文件后，让我们打开控制面板，双击“添加/删除硬件”，选择“添加/排除设备故障”->“添加新设备”->“否，我想从列表选择硬件”->“其它设备”->“从磁盘安装”，选择 HelloWDM.INF 所在的路径，然后安装。

当安装完成后，系统就会添加上你写好的驱动程序了。（可以在“设备管理器”中查看到）。然后重启电脑，这个驱动程序就投入使用啦。

关于安装，我也只知道这么多，到底安装驱动程序时，操作系统都作了些什么，我也不是很清楚，等我弄明白了我再贴上。

2005 年 9 月 25 日 16:52