例如，鼠标和键盘设备就是这样。假定 DriverEntry 枚举了所有鼠标或键盘硬件并为它们创建了设备对象，但如果 DriverEntry 例程运行的太快，那么这些驱动程序将不能正常工作。因此，它们必须使用 IoRegisterDriverReinitialization 函数寄存一个例程，之后 I/O 管理器在某个驱动程序检测到新硬件存在时再回调这个寄存例程。最后“再初始化例程”运行，同时也把自身寄存为下一次回调的函数。

WDM 驱动程序不需要寄存再初始化例程，因为它们不需要用自己的代码去检测硬件。PnP 管理器自动把新硬件匹配到正确的 WDM 驱动程序上，并调用该驱动程序的 AddDevice 例程，再由 AddDevice 例程做所有必要的初始化工作。

4.2.5 USB 驱动程序的卸载

在 WDM 驱动程序中，DriverUnload 例程的作用就是释放 DriverEntry 例程在全局初始化过程中申请的任何资源，但它几乎没什么可做。如果你在 DriverEntry 中备份了 RegistryPath 串，应该在这里释放备份所占用的内存：

```
VOID DriverUnload(PDRIVER_OBJECT DriverObject)
{
  RtlFreeUnicodeString(&servkey);
}
```

如果 DriverEntry 例程返回一个失败状态代码，系统将不再调用 DriverUnload 例程。所以，不能让 DriverEntry 例程出错后产生任何副作用，必须在它返回错误代码前消除副作用。

4.3 INF 文件

INF 是 Device Information File 的英文缩写，是 Microsoft 公司为硬件设备制造商发布其驱动程序推出的一种文件格式，INF 文件中包含硬件设备的信息或脚本以控制硬件操作。在 INF 文件中指明了硬件驱动该如何安装到系统中，源文件在哪里、安装到哪一个文件夹中、怎样在注册表中加入自身相关信息等等。安装监视器、调制解调器和打印机等设备所需的驱动程序，都是通过 INF 文件，正是 INF 的功劳才使得 Windows 可以找到这些硬件设备的驱动并正确安装。当我们通过“开始→控制面板→添加删除程序→Windows 安装程序”来添加系统组件的时候，INF 文件将会自动调用。而在其他场合下，则需要在 INF 文件上点击鼠标右键，然后选择“安装”，才能顺利安装应用程序。在 C：/Windows/INF 文

件夹中存放有大量的 INF 文件。该文件夹一般处在隐含状态，如果想查看该文件夹，可选择“查看→文件夹选项→查看”，然后在“文件和文件夹”选项下选择“显示所有文件”。

INF 文件含有安装一个 WDM 设备驱动程序需要的所有必要的信息，包括要复制的文件列表、要创建的注册表项、设备的 ID 和兼容 ID 等。INF 文件是一个文本文件，它由节组成，每一节从节名称开始，后面是节内容。当发现新的设备时(系统启动时，在安装热插拔设备时，或者从控制面板安装新设备时)，就调用 Windows 的"添加新设备向导"执行。这个向导扫描所有可用的 INF 文件，试图找到合适的驱动程序。Windows 首先选择硬件 ID 匹配的设备的 INF 文件，否则它选择其兼容 ID 与设备 ID 最佳匹配的 INF 文件，若仍未找到提示用户选择驱动程序 INF 文件。然后根据 INF 文件的指令安装驱动程序，驱动程序可执行文件被复制到正确的位置，通常是 Windows System32\Drivers 目录，然后创建各种注册表项，驱动程序被装入内存，并执行它的 DirverEntry 例程。对新的设备调用 AddDevice 例程，给设备分配 I/O，DMA，中断等资源(USB 设备不需要分配资源)，启动设备，然后正常的 I/O 操作就可以继续进行。使用后的 INF 文件复制到 Windows INF 子目录。

INF 文件的结构

INF 文件其实是一种纯文本文件，可以用任意一款文本编辑软件来打开进行编辑，如：记事本、写字板等。INF 文件有一整套的编写规则，每一个 INF 文件都是严格按照这些规则来编写的。

规则一：INF 文件是分节的，每一个 INF 文件有许多的节组成，节名用方括号括起来。这些节名有些是系统定义好的，有一些是用户自定义的。每一个节名最长为 255 个字符（Windows 2000/XP/2003 操作系统中）或 28 个字符（Windows 98 操作系统中）。节与节之间没有先后顺序的区别，另外，同一个 INF 文件中如果出现两个同样的节名，则系统会自动将这两个节名下面的条目合并到一起。

规则二：在节与节之间的内容叫条目，每一个节又是由许多的条目组成的，每一个条目都是由形如“signature="$CHICAGO$"”的形式组成的。如果每一个条目的等号后有多个值，则每一个值之间用“，”号分隔开。

规则三：INF 文件对大小写不敏感。

规则四：“；”号后面的内容为注释。

规则五：如果一个条目的内容过多，在一行无法书写完全，则用“\”将一行内容书写为多行。

1．Version 节

每一个 INF 文件都包含一个这样的节，该节中的条目主要是描述此 INF 文件支持的设备类型和适用的操作系统。在该节中如果出现“signature="$CHICAGO$""这样的条目则表示该 INF 文件适用于 Windows 98 之后的所有操作系统，如果包含“signature="$Windows NT$""这样的条目则表示该 INF 文件适用于 Windows 2000/XP/2003 操作系统，而且两者必具其一。

另外，该节中“CLASS”条目很重要，它表明了设备的类型，常见的类型有：Display（显示设备，如显卡）、Media（多媒体设备，如声卡）、Net（网络设备，通常是网卡）、Modem（调制解调器）、Printer（打印设备）、Image（图像捕获设备，如摄像头）。

2．Manufacturer 节

该节中的条目主要是描述 INF 文件可以识别的所有硬件设备，其中包含有设备的生产厂家，以便设备的正确安装。如“%ATi%=ATi”指明设备的生产商为“ATi”，这个大家都不会陌生吧，是显卡的生产商。

3．SourceDisksNames 节

该节主要指明安装文件所在的介质。如:“1="ATi Drivers Release CD""表明所有驱动程序都在零售版的 CD 盘上。

4．SourceDiskFiles 节

驱动程序文件列表及被安装的位置，该节必须结合[SourceDisksNames]节才能知道具体的位置。如“atinbtxx.sys=1”，则你必须到[DestinationDirs]节去查看“1”具体代表那个位置。

5．DestinationDirs 节

INF 文件会指示安装程序在安装的过程中，将一些文件复制到硬盘上，或者将硬盘上的一些文件删除、重命名等。该节即指定了为实现上述目的的文件所在的目的路径。

6．DefaultInstall 节和 Install 节

在这一节中描述了设备驱动程序与硬件设备的实际属性。默认情况下，首先执行[DefaultInstall]节，该节指定了要复制或删除的文件，注册表的更新，INF文件的更新等等信息，同时又包含指向其他节的指针。

7．String 节

这一节中定义了字符串变量，当某些字符串频繁地出现在 INF 文件中，为简化输入可以在该节中定义一个字符串变量，代表该字符串出现在 INF 文件中。

4.4 USB 应用程序设计

在 Win32 系统中，把每一个设备都抽象为文件，此时的应用程序只需要通过几条简单的文件操作 API 函数，就可以实现与驱动程序中某个设备通信。一个驱动程序可以驱动多个设备，并且此驱动程序可能为 Windows 系统中已有的，也可能为用户安装的。通常，这些 Win32 API 函数有以下几种：

（1）CreateFile 函数。打开一个设备，返回一个与设备相关的句柄。如果调用成功，那么该函数返回打开设备的句柄。

（2）ReadFile 函数。从设备中读取数据。

（3）WriteFile 函数。向设备写数据。

（4）DeviceIoControl 函数。对设备进行一些自定义的操作，比如更改设置等。

（5）CloseFile 函数。关闭一个由 CreateFile 打开的设备。

这些 API 函数的执行，都对应着驱动程序的一些分发例程，表 4-1 是常用 API 函数和驱动程序的 IRP 对应关系表。例如，当应用程序调用函数 CreateFile 来打开设备对象时，操作系统代替应用程序，向驱动程序发送系统 I/O 控制消息 IRP_MJ_CREATE，从而驱动程序响应这个消息，对应的例程被调用。如果驱动程序没有提供该例程，CreateFile 调用就会失败。

表 4-1 常用 API 函数和驱动程序的 IRP 对应关系表

API 函数	IRP	说 明
CreateFile	IRP_MJ_CREATE	打开设备
ReadFile	IRP_MJ_READ	从设备获取数据
WriteFile	IRP_MJ_WRITE	向设备发送数据
CloseFile	IRP_MJ_CLOSE	关闭设备
DeviceIoContrlo	IRP_MJ_DEVICE_CONTROL	控制操作