备功能驱动程序, PC 机应用程序

USB 控制器固件程序实现设备的枚举以及端点与主机的通信。控制器的接收应用程序接收主机下载的文件。 设备功能驱动程序为应用程序和底层驱动程序之间提供接口。当一个应用程序启动一个 API 调用后，Windows 把调用传递给设备驱动程序，设备驱动程序把请求传递到底层驱动程序，底层驱动程序对硬件进行相应的操作。PC 机应用程序功能是实现主机文件的下载。

4.2.2 USB 驱动程序相关概念

设备：这里的设备仅指在编写驱动程序时将设备看成一个整体。同一个设备可以有几种不同的配置

配置：对设备的若干种配置方法中的一种，在驱动程序中，配置用一些结构来表示。从一个配置结构中，可以知道设备有多少接口。

接口：设备中功能相关或相近的一组端点的集合。在编写驱动程序时，可以从接口描述符中获取相关信息。

端点：从用户的角度看，可以直接进行 IO 数据流操作的设备中的基本单位。端点是单向的，如果要对设备进行双向的 IO，必须至少两个端点。

管道：一个端点与客户程序进行 IO 操作时使用的中介。管道与端点是一一对应的，端点侧重于静态的概念，管道侧重于动态的概念。

URB（USB Request Block）：USB 请求块。对 USB 进行操作的请求都应调用系统例程将其转化为一个 URB 结构，然后使用系统级的 IRP 将其提交。

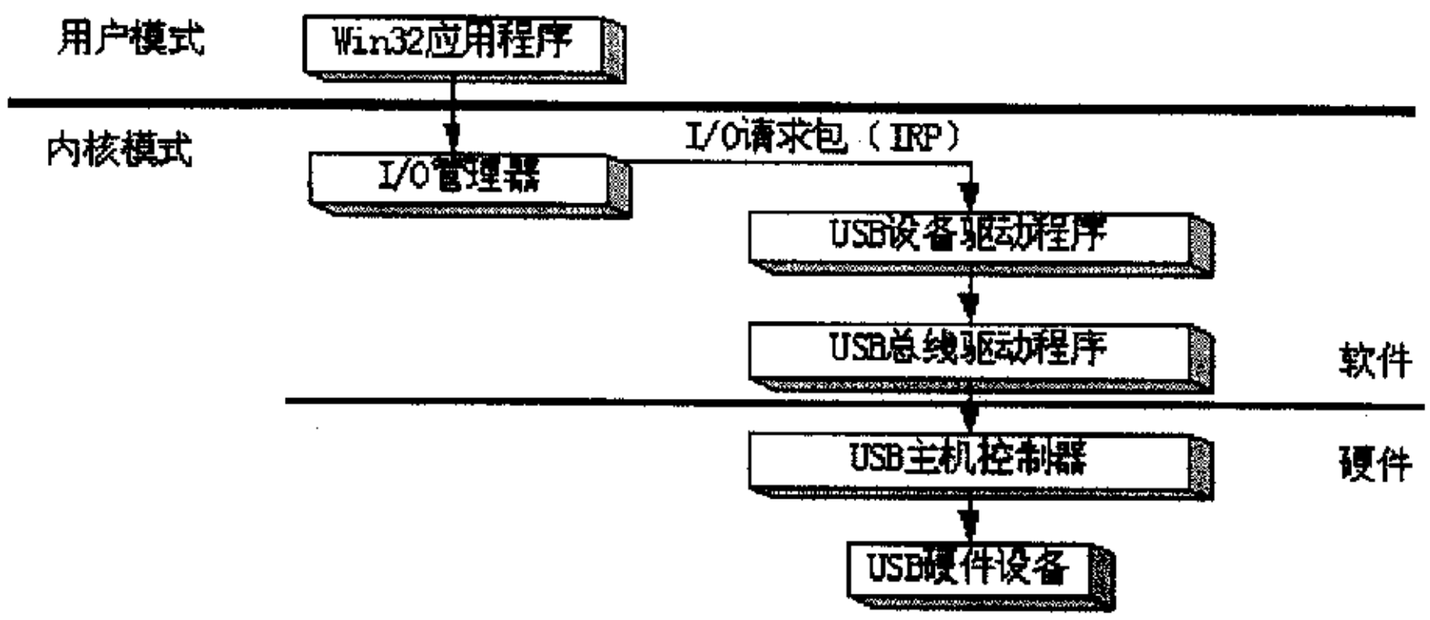

图 4-3 USB 接口软件结构

4.2.3 USB 驱动程序的入口

DriverEntry 是内核模式驱动程序主入口点常用的名字。I/O 管理器按下面

方式调用该例程：

```
extern "C" NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject,
                                IN PUNICODE_STRING RegistryPath)
{
   ...
}
```

DriverEntry 的第一个参数是一个指针，指向一个刚被初始化的驱动程序对象，该对象就代表你的驱动程序。WDM 驱动程序的 DriverEntry 例程应完成对这个对象的初始化并返回。非 WDM 驱动程序需要做大量额外的工作，它们必须探测自己的硬件，为硬件创建设备对象(用于代表硬件)，配置并初始化硬件使其正常工作。而对于 WDM 驱动程序，颇麻烦的硬件探测和配置工作由 PnP 管理器自动完成;

DriverEntry 的第二个参数是设备服务键的键名。这个串不是长期存在的(函数返回后可能消失)，如果以后想使用该串就必须先把它复制到安全的地方。

对于 WDM 驱动程序的 DriverEntry 例程，其主要工作是把各种函数指针填入驱动程序对象。这些指针为操作系统指明了驱动程序容器中各种子例程的位置。它们包括下面这些指针成员(驱动程序对象中)：

DriverUnload 指向驱动程序的清除例程。I/O 管理器会在卸载驱动程序前调用该例程。通常，WDM 驱动程序的 DriverEntry 例程一般不分配任何资源，所以 DriverUnload 例程也没有什么清除工作要做。DriverExtension->AddDevice 指向驱动程序的 AddDevice 函数。PnP 管理器将为每个硬件实例调用一次 AddDevice 例程。DriverStartIo 如果驱动程序使用标准的 IRP 排队方式，应该设置该成员，使其指向驱动程序的 StartIo 例程。下面是一个近乎完整的 DriverEntry 例程：

```
extern "C"
NTSTATUS    DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING
RegistryPath)
{
   DriverObject->DriverUnload = DriverUnload;
   DriverObject->DriverExtension->AddDevice = AddDevice;
```

```
DriverObject->DriverStartIo = StartIo;
DriverObject->MajorFunction[IRP_MJ_PNP] = DispatchPnp;
DriverObject->MajorFunction[IRP_MJ_POWER] = DispatchPower;
DriverObject->MajorFunction[IRP_MJ_SYSTEM_CONTROL] = DispatchWmi;
...
servkey.Buffer = (PWSTR) ExAllocatePool(PagedPool, RegistryPath->Length +
sizeof(WCHAR));
  if (!servkey.Buffer)
    return STATUS_INSUFFICIENT_RESOURCES;
  servkey.MaximumLength = RegistryPath->Length + sizeof(WCHAR);
  RtlCopyUnicodeString(&servkey, RegistryPath);
  return STATUS_SUCCESS;
}
```

前三条语句为驱动程序的其它入口点设置了函数指针。在这里，我用了能表达其功能的名字命名了这些函数：DriverUnload、AddDevice、StartIo。每个 WDM 驱动程序必须能处理 PNP、POWER、SYSTEM_CONTROL 这三种请求；应该在这里为这些请求指定派遣函数。在早期的 Windows 2000 DDK 中。IRP_MJ_SYSTEM_CONTROL 曾被称作 IRP_MJ_WMI，所以我把系统控制派遣函数命名为 DispatchWmi。在省略号处，你可以插入设置其它 MajorFunction 指针的代码。

如果驱动程序需要访问设备的服务键，可以在这里备份 RegistryPath 串。例如，如果驱动程序要作为 WMI 生产者，则需要备份这个串。这里我假设已经在某处声明了一个类型为 UNICODE_STRING 的全局变量 servkey。返回 STATUS_SUCCESS 指出函数成功。如果函数失败，应该返回 NTSTATUS.H 中的一个错误代码，或者返回用户定义的错误代码。STATUS_SUCCESS 的值为 0。

4.2.4 USB 驱动程序的初始化

I/O 管理器提供了一个服务函数：IoRegisterDriverReinitialization。它可以为非 WDM 驱动程序解决一个奇特的问题。非 WDM 驱动程序需要在 DriverEntry 例程中枚举它的硬件，并在其硬件的所有可能实例被识别前装入内存并初始化。