第四章 USB 驱动程序开发

一个完整的 USB 系统包括主机系统和 USB 设备。所有的传输事务都是由主机发起的。一个主机系统又可以分为以下几个层次结构，如图 4-1 所示：

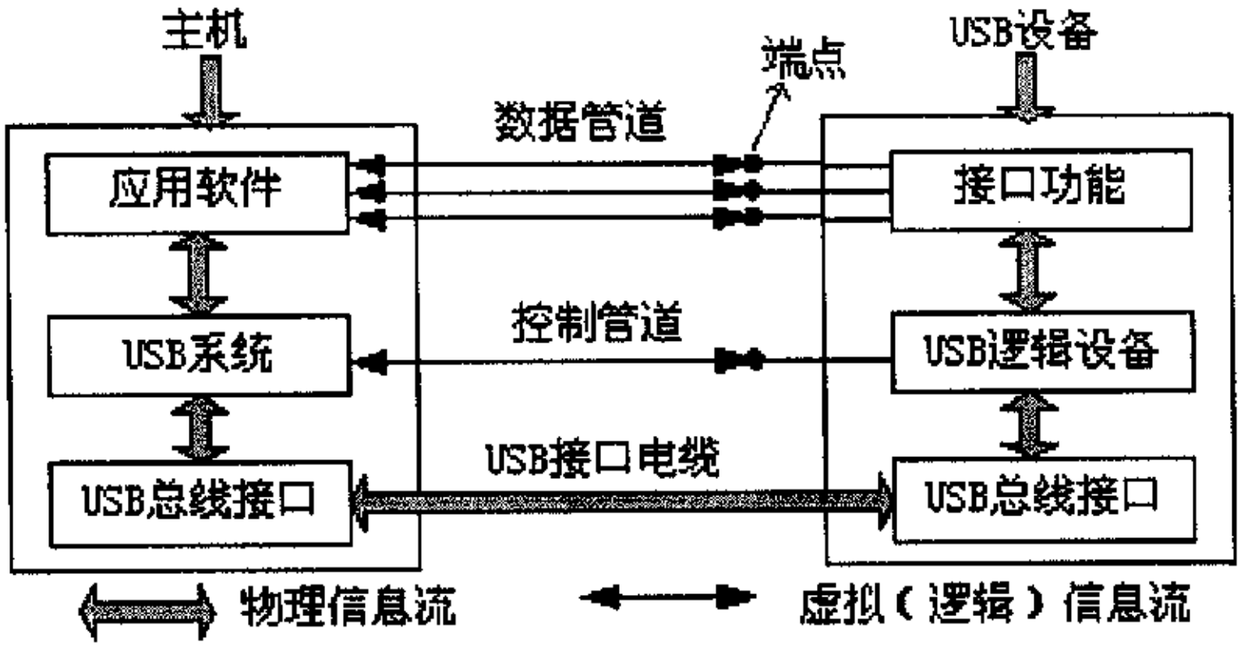

图 4-1 USB 系统主机系统

USB 总线接口包括 USB 主控制器和根集线器，其中 USB 主控制器负责处理主机与设备之间电气和协议层的互连，根集线器提供 USB 设备连接点。USB 系统使用 USB 主控制器来管理主机和 USB 设备之间的数据传输，另外它也负责管理 USB 资源,如带宽等。应用软件不能直接访问 USB 设备硬件，而通过 USB 系统和 USB 总线接口与 USB 设备进行交互。

USB 设备包含一些向主机软件提供一系列 USB 设备的特征和能力的信息的设备描述符，用来配置设备和定位 USB 设备驱动程序。这些信息确保了主机以正确的方式访问设备。通常，一个设备有一个或多个配置(Configuration)来控制其行为。配置是接口(Interface)的集合，接口指出软件应该如何访问硬件。接口又是端点(endpoint)的集合，每一个与 USB 交换数据的硬件就为端点，它是作为通信管道的一个终点。图 1 显示了一个多层次结构的通信模型，它表明了端点和管道所扮演的角色。

编写 USB 驱动程序必须采用 WDM 驱动程序。对于 USB 设备来说，其驱动程序可分为 USB 底层驱动程序和 USB 功能驱动程序。USB 底层驱动程序由操作系统提供，它位于 USB 功能驱动程序的下面，负责与实际的硬件打交道，实现繁琐的底层通信。USB 功能驱动程序由设备开发者编写，位于 USB 底层驱动程序的上面，不与实际的硬件打交道，而是通过向 USB 底层驱动程序发送包含

URB(USB Request Block，USB 请求块)的 IRP(I/O Request Packet，I/O 请求包)，来实现对 USB 设备信息的发送和接收。采用这种分层驱动程序的设计方法有两个优点：(1)多个 USB 设备可以通过 USB 底层驱动程序来协调它们的工作；(2)编写分层驱动程序较之编写单一驱动程序相对简单，且可以节省内存和资源，不易出错。若应用程序想对设备进行 I/O 操作，它需调用 Windows API 函数 ，I/O 管理器将此请求构造成一个合适的 I/O 请求包 IRP 并把它传递给 USB 功能驱动程序。USB 功能驱动程序接收到这个 IRP 以后，根据 IRP 中包含的具体操作代码，构造相应 USB 请求块并把此 URB 放到一个新的 IRP 中，然后把此 IRP 传递 USB 底层驱动程序，USB 底层驱动程序根据 IRP 中所含的 URB 执行响应的操作(如从 USB 设备读取数据)，并把操作结果返还给 USB 功能驱动程序。USB 功能驱动程序接收到此 IRP 后，将操作结果通过 IRP 返还给 I/O 管理器，最后 I/O 管理器将此 IRP 操作结果返还给应用程序，至此应用程序对 USB 设备的一次 I/O 操作完成。USB 功能驱动程序除负责处理应用程序的 I/O 请求外，还要处理 PnP 请求(如设备启动请求 IRP_MN_START_DEVICE，设备删除 IRP_MN_REMOVE_DEVICE 等)。通过对这些 PnP 请求的处理，USB 功能驱动程序可支持设备的热插拔和即插即用功能。

4.1 USB 启动的过程

为了叙述清晰，本章在介绍 USB 驱动程序设计之前，先介绍 USB 的启动过程。 USB 系统主要由主控制器(Host Controller)、USB Hub 和 USB 外设(Peripherals Node)组成系统拓扑结构，如图 4-2 所示。

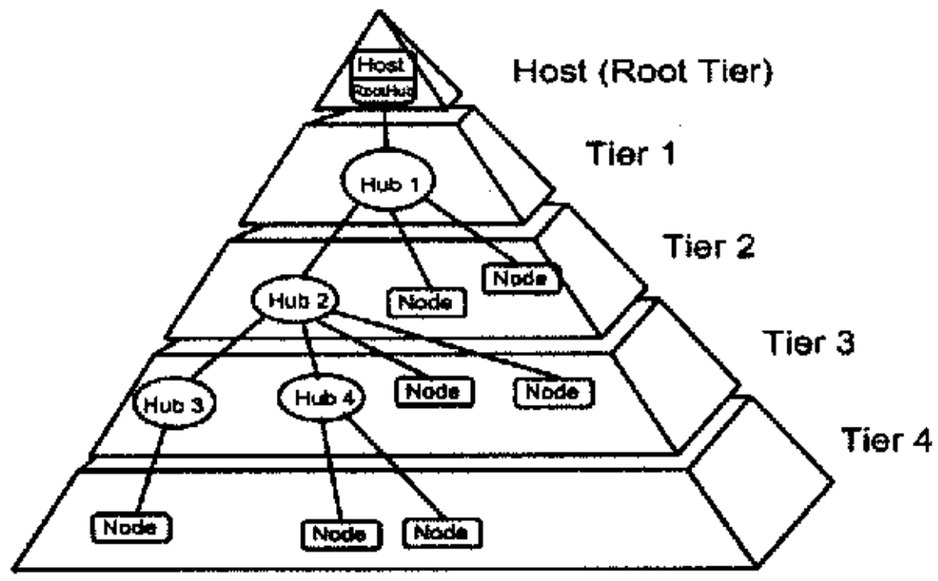

图 4-2 USB 系统拓扑结构

在应用程序可以与一个 USB 设备通信之前，主机需要知道设备支持哪些传输类型和终端，主机也必须分配一个地址给设备，主机通过一个被称为枚举的信