开发的 Windows 操作系统中都将存在，比如 Windows Me 和 Windows XP。微软在通过 WDM 模型的引入，希望减轻设备驱动程序的开发难度和周期，逐渐规范设备驱动程序的开发，应该说，WDM 将成为以后设备驱动程序的主流。

3.2.1 WDM 驱动程序模型

WDM 模型主要采用分层的方法，模仿面向对象的技术，按照微软一贯的思路，先进行逻辑上的“分层”，然后将标准的实现和低层细节“封装”起来，形成“基类”，客户程序通过“继承”的方式来扩展“基类”的功能，完成所需要的实现。在微软的技术文献中，称 Windows NT 和 Windows 2000 为“基于对象”（object-based）的系统，和操作系统一样，WDM 驱动程序模型也是“基于对象”的系统。在系统中既使用对象又使用类和继承等机制，而且对象之间仅能通过传递消息实现彼此的通信，这样的方法才称为“面向对象的方法”。如果仅使用对象，则这种方法可以称为“基于对象”的方法。“基于对象”的方法虽然不能得到“面向对象”的所有优点，但依然可以使系统的设计、分析和理解更加的清楚。

在 WDM 模型中，每个硬件设备至少有两个驱动程序：一个功能驱动程序（function driver）和一个总线驱动程序（bus driver）。一个设备还可能有过滤驱动程序（filter driver），用来变更标准设备驱动程序的行为。这些服务于同一个设备的驱动程序组成了一个链表，称为设备栈。

WDM 模型使用了如图 3-1 的层次结构。图中左边是一个设备对象堆栈。设备对象是系统为帮助软件管理硬件而创建的数据结构。一个物理硬件可以有多个这样的数据结构。处于堆栈最底层的设备对象称为物理设备对象(physical device object)，或简称为 PDO。在设备对象堆栈的中间某处有一个对象称为功能设备对象(functional device object)，或简称 FDO。在 FDO 的上面和下面还会有一些过滤器设备对象(filter device object)。位于 FDO 上面的过滤器设备对象称为上层过滤器，位于 FDO 下面(但仍在 PDO 之上)的过滤器设备对象称为下层过滤器。

这样的数据结构。处于堆栈最底层的设备对象称为物理设备对象(physical device object)，或简称为 PDO。在设备对象堆栈的中间某处有一个对象称为功能设备对象(functional device object)，或简称 FDO。在 FDO 的上面和下面还会有一些过滤器设备对象(filter device object)。位于 FDO 上面的过滤器设备对象称为上层过滤器，位于 FDO 下面(但仍在 PDO 之上)的过滤器设备对象称为下层过滤器。

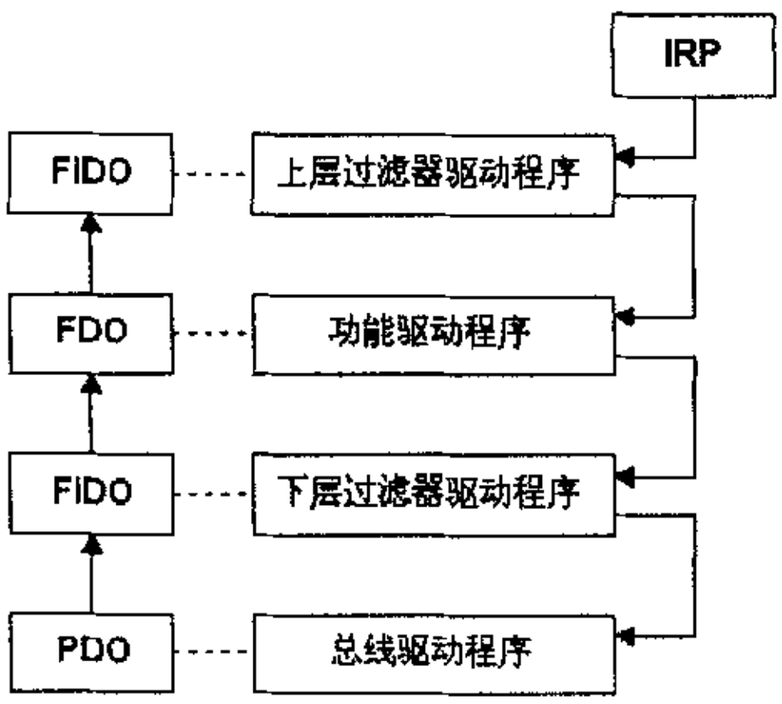

图 3-1WDM 中设备对象和驱动

总线驱动程序

总线驱动程序为实际的 I / O 总线服务，比如 IEEE 1394。在 WDM 的定义中，一个总线是这样的设备，它用来连接其他的物理的、逻辑的、虚拟的设备。总线包括传统的总线 SCSI 和 PCI，也包括并口、串口、以及 i8042 端口。微软已经为 Windows 操作系统提供了总线驱动程序。总线驱动程序已经包含在操作系统里了，用户不必安装。一个总线驱动程序负责以下的工作：

- 枚举总线上的设备；
- 向操作系统报告总线上的动态事件；
- 响应即插即用和电源管理的 I / O 请求；
- 提供总线的多路存取（对于一些总线）；
- 管理总线上的设备；
- 功能驱动程序
- 功能驱动程序是物理设备的主要驱动程序，它实现设备的具体功能，一般由设备的生产商来编写。功能驱动程序的主要功能是：
- 提供对设备的操作接口；
- 操作对设备的读写；
- 管理设备的电源策略；

过滤驱动程序

过滤驱动程序是一个可选项，当一个用户需要改变或新添一些功能到一个设备、一类设备或一种总线时，就可以编写一个过滤驱动程序。在设备栈里，过滤驱动程序安装在一个或几个设备驱动程序的上面或下面。过滤驱动程序拦截对具

体设备、类设备、总线的请求，做相应的处理，以改变设备的行为或添加新的功能。但过滤驱动程序只处理那些它所关心的I/O请求，对于其他的请求可以交给其他的驱动程序来处理，这样可以非常灵活改变设备的行为，至少用户会这样看。比如：

一个USB键盘的上层过滤驱动程序可以强制执行附加的安全检查。

一个鼠标的低层过滤驱动程序，通过对鼠标移动的数据做非线性的转换，可以得到一个有加速效果的鼠标轨迹。

功能驱动程序的组成

功能驱动程序由类驱动程序和微型驱动程序（Minidriver）组成。类驱动程序实现了某一类设备的常用操作，由微软提供，驱动程序的开发者可以只编写非常小的微型驱动程序，去处理具体设备特殊的操作，而对于其他大量的常规操作，可以调用该类的类驱动程序，这也是WDM驱动程序的优点之一。

微软提供的类驱动程序处理常用的系统任务，比如，即插即用功能和电源管理。类驱动程序保证了操作系统在处理类似的任务时的一致性，从而提高了系统的稳定性。

设备生产商提供微型驱动程序，以实现自己设备的特殊功能，同时调用合适的类驱动程序完成其他的通用工作。将大量的标准操作的代码通过各种类驱动程序来实现，并集成在操作系统中，这样的方式可以有效的减少具体设备的微型驱动程序的大小，也就减小了程序出错的可能。

如果某一类设备存在着工业标准，微软就会提供一个该类设备的WDM类驱动程序。这个类驱动程序实现了该类设备所有必须的任务，但不实现任何具体设备所特有的东西。比如，微软提供的HID（人工输入设备）类驱动程序的实现，是根据USB HID 类规范v.11的规定，但并不实现任何一种具体设备的特殊功能，比如，USB键盘、鼠标、游戏控制等等。

微软支持的**WDM**总线和类驱动程序包括：

高级配置和电源接口（ACPI）总线驱动程序与PC机的ACPI BIOS打交道，枚举系统中的设备并控制它们的功率使用。外设组件互联（PCI）总线驱动程序枚举和配置PCI总线上的设备。PnP ISA总线驱动程序对可以使用即插即用配置的工业标准体系结构（ISA）设备做类似的工作。

流类驱动程序支持数据流的高带宽传输，以得到更快的数据处理速度。这个类驱动程序经常与音频端口类驱动程序结合使用，以支持实时的视频和音频。该类驱动也负责多任务时序，直接内存存取(DMA)，内存优化，即插即用和 I/O 缓冲区管理。

IEEE 1394 总线驱动程序枚举和控制 IEEE 1394 高速总线。

USB 总线驱动程序枚举和控制低速的 USB 总线。USB 客户驱动程序使用各种 IOCTL 通过 USB 类驱动程序访问它们的设备。

SCSI 和 CDROM / DVD 驱动程序用于访问硬盘、软盘、光驱和 DVD。

人工输入设备（HID）类驱动程序管理多种总线（如 USB）间的数据与指令语法翻译。大多数时候，本类驱动控制由用户交互接口传来的数据，如键盘，鼠标和游戏杆等。

静态图像体系结构（STI）根本不是一个驱动程序，而是使用微型驱动程序获得扫描仪和静态相机的一种手段。STI 目前支持 SCSI 设备、串行设备、并行设备和 USB 设备。STI 是基于 COM（组件对象模型）。

分层的驱动程序结构

如上所述 WDM 使用了分层的驱动程序结构，而且 WDM 是基于对象的。为了便于对硬件的管理，WDM 里对每一个单一的硬件引入了一些数据结构，在图 3-1 的左手部分描述了一个 DEVICE_OBJECT 数据结构栈。在数据结构栈中最低层的是物理设备对象（Physical Device Object），用于描述我们的设备与物理总线的关系，简称为 PDO。在 PDO 的上面，有功能设备对象（Function device Object），用来描述设备的逻辑功能，简称为 FDO。在数据结构栈的其他位置， FDO 的上面或下面，有许多的过滤设备对象（Filter Device Objects），简称为 FiDO。在数据结构栈中的每一个对象都属于一个特定的驱动程序，如图 3-1 中间的虚线所指示的，PDO 属于总线驱动程序，FDO 属于功能驱动程序，FiDO 属于过滤驱动程序。

操作系统中的即插即用管理器（PnP Manager）根据设备驱动程序的指令来建立这个数据对象堆栈。前面我们已经知道，总线驱动程序的作用之一是枚举总线上的设备，当总线驱动程序检测到一个设备时，PnP 管理器就马上建立一个 PDO。当建立好 PDO 之后，PnP 管理器通过查找注册表来找到其他的过滤驱动

程序和功能驱动程序。设备的安装程序负责建立这些注册表里的表项，驱动程序的安装，是根据 INF 文件中的指令进行的。注册表中的表项指明了各种驱动程序在数据对象堆栈中的位置，于是 PnP 管理器开始装载最低层的过滤驱动程序，并调用该驱动程序的 AddDevice 函数。该函数在数据对象堆栈中建立一个 FiDO，同时也将前面建立的 PDO 和这个 FiDO 联系在一起。PnP 管理器反复的实现该过程，装载其他位置靠上的低层过滤驱动程序、功能驱动程序、上层过滤驱动程序，直到该堆栈完成。

应用程序对设备的存取通过提交 IO 请求包（IRP）来进行。在操作系统中，对设备的存取过程是这样的：操作系统中的 I / O 管理器接受 I / O 请求（通常是由用户态的应用程序发出的），建立相应的 IRP 来描述它，将 IRP 发送给合适的驱动程序，然后跟踪执行过程，当操作完成后，将返回的状态通知请求的发起者。操作系统中的 I / O 管理器、即插即用管理器、电源管理器都使用 IRP 来与内核模式驱动程序、WDM 驱动程序进行通信，并且，各驱动程序之间的通信也是依靠 IRP。

在 WDM 驱动程序中，IRP 首先从最上层进入，如图 3-1 里右手边的箭头，然后，依次往下传送。在每一层，驱动程序自行决定对 IRP 的处理。有时，一个驱动程序除了把继续 IRP 向下传递外，并不做任何事情。有时，一个驱动程序会完全接管 IRP，不再把它向下传递了。当然，一个驱动程序也可以处理 IRP 后，再把它继续向下传递。这取决于驱动程序的功能和 IRP 的含义。

微软在 WDM 模型中使用分层的驱动程序结构，通过分层的方法，在处理对设备的 I / O 请求时，利用添加合适的驱动程序层的方法，从而非常灵活的改变设备的行为，以实现不同设备的功能。

3.2.2 WDM 驱动程序模块组成

一个 WDM 设备驱动程序的模块组成（如图 3-2 所示）主要包括：初始化自己、创建和删除设备、处理 Win32 打开和关闭文件句柄的请求、处理 Win32 系统的 I/O 请求、对设备的串行化访问、访问硬件、调用其它驱动程序、取消 I/O 请求、超时 I/O 请求、处理一个 PnP 设备被加入或删除的情况、处理电源管理请求、调用 Windows 管理诊断 WMI 向系统管理员报告。其中，“初始化”模块必不可少，所有驱动程序都需要通过分发例程处理用户的 I/O 请求。而 WDM 风格