当I/O管理器分配一个IRP时，为了分派一个足够大的IRP，它必须知道这个设备栈的深度。在建立设备栈的时候，I/O管理器会在每一个设备对象的域中记录栈的深度。注意，在任何栈中都没有正式地定义下一个设备对象是什么。这个信息被保存在栈中当前驱动程序的私有数据结构中。事实上这个栈实际上并不一定是一个真正的栈。在每一层栈中，驱动程序都可以自由地分配新的IRP，或者继续使用原来的IRP，或者发送一个I/O操作给另一个设备栈，或者甚至转换到一个系统工作线程中继续执行。

IRP包含标志位、索引到驱动程序分派表的操作码、指向内核与用户缓冲区的指针和一个MDL（内存描述符列表）列表。MDL用于描述由缓冲区描述的物理内存框，也就是用于DMA操作。有一些域用于取消和完成操作。当I/O操作已经完成后，在处理IRP时用于排列这个IRP到设备中的域会被重用。目的是给用于在原始线程的上下文中调用I/O管理器的完成例程的APC控制对象提供内存。还有一个连接域用于连接所有的外部IRP到初始化它们的线程。

3. 设备栈（Device Stack）

Windows Vista中的驱动程序可以自己完成所有的任务，如图11-40所示的打印机驱动程序。另一方面，驱动程序也可以堆叠起来，即一个请求可以在一组驱动程序之间传递，每个驱动程序完成一部分工作。图11-40也给出了两个堆叠的驱动程序。

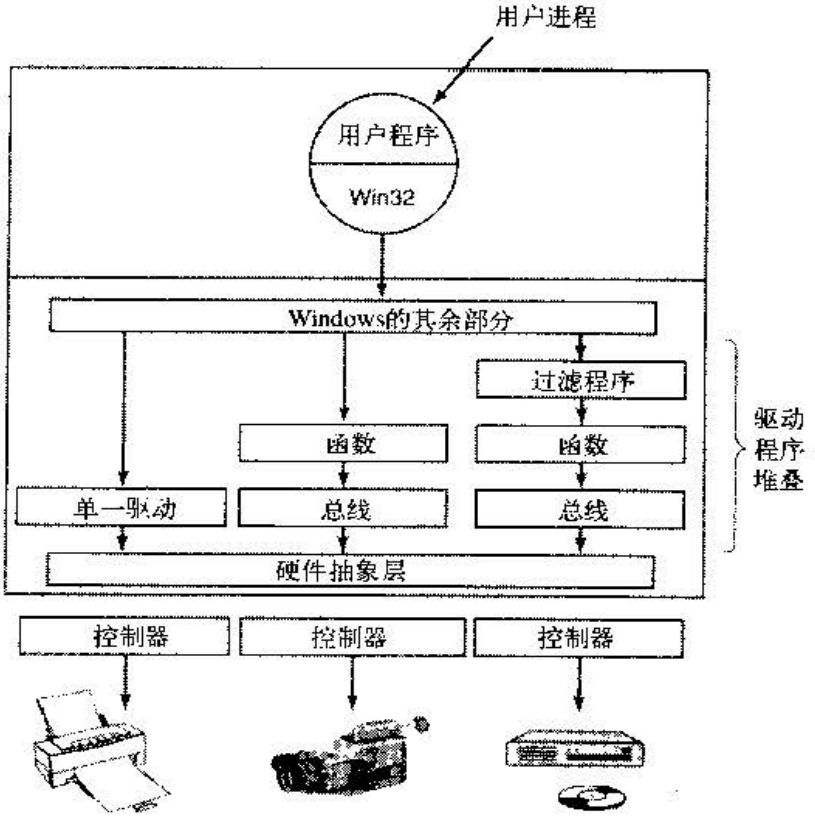

图11-40 Windows允许驱动程序堆叠起来操作设备。这种堆叠是通过设备对象（Device Object）来表示的

堆叠驱动程序的一个常见用途是将总线管理与控制设备的功能性工作分离。因为要考虑多种模式和总线事务，PCI总线上的总线管理相当复杂。通过将这部分工作与特定于设备的部分分离，驱动程序开发人员就可以从学习如何控制总线中解脱出来了。他们只要在驱动栈中使用标准总线驱动程序就可以了。类似地，USB和SCSI驱动程序都有一个特定于设备的部分和一个通用部分。Windows为其中的通用部分提供了公共的驱动程序。

堆叠设备驱动程序的另一个用途是将过滤器驱动程序（filter driver）插入到驱动栈中。我们已经讨论过文件系统过滤器驱动程序的使用了，该驱动程序插入到文件系统之上。过滤器驱动程序也用于管理物理硬件。在IRP沿着设备栈（Device Stack）向下传递的过程中，以及在完成操作（completion operation）中IRP沿着设备栈中各个设备驱动程序指定的完成例程（completion routine）向上传递的过