这些系统调用本质上是特定于文件的ioctl的一种形式。这组操作可以用来重命名或删除一个文件。但是请注意，它们处理的并不是文件名，所以要重命名或删除一个文件之前必须先打开这个文件。它们也可以被用来重新命名NTFS上的交换数据流（见11.8节）。

存在独立的API（NtLockFile和NtUnlockFile）用来设置和删除文件中字节域的锁。通过使用共享模式，NtCreateFile允许访问被限制的整个文件。另一种是这些锁API，它们用来强制访问文件中受限制的字节域。读操作和写操作必须提供一个与提供给NtLockFile的钥匙相符合的密钥，以便操作被锁定的区域。

UNIX中也有类似的功能，但在UNIX中应用程序可以自由决定是否认同这个区域锁。NtFsControlFile和前面提到的查询和设置操作很相像，但它是一个旨在处理特定文件的操作，其他的API并不适合处理这种文件。例如，有些操作只针对特定的文件系统。

最后，还有一些其他的系统调用，比如NtFlushBuffersFile。像UNIX的sync系统调用一样，它强制把文件系统数据写回到磁盘。NtCancelIoFile用于取消对一个特定文件的外部I/O请求，NtDeviceIoControlFile实现了对设备的ioctl操作。它的操作清单实际上比ioctl更长。有一些系统调用用于按文件名删除文件，并查询特定文件的属性——但这些操作只是由上面列出的其他I/O管理器操作包装而成的。在这里，我们虽然列出，但并不是真的要把它们实现成独立的系统调用。还有一些用于处理I/O完成端口的系统调用，Windows的队列功能帮助多线程服务器提高使用异步I/O操作的效率，主要通过按需准备线程并降低在专用线程上服务I/O所需要的上下文切换数目来实现。

11.7.3 I/O实现

Windows I/O系统由即插即用服务、电源管理器、I/O管理器和设备驱动模型组成。即插即用服务检测硬件配置上的改变并且为每个设备创建或拆卸设备栈，也会引起设备驱动程序的装载和卸载。功耗管理器会调节I/O设备的功耗状态，以在设备不用的时候降低系统功耗。I/O管理器为管理I/O内核对象以及如IoCallDrivers和IoCompleteRequest等基于IRP的操作提供支持。但是，支持Windows I/O所需要的大部分工作都由设备驱动程序本身实现。

1. 设备驱动程序

为了确保设备驱动程序能和Windows Vista的其余部分协同工作，微软公司定义了设备驱动程序需要符合的WDM（Windows驱动程序模型）。WDM被设计成能在Windows 98系统上运行，也能在从Windows 2000开始的基于NT的系统上运行。WDM允许开发人员编写与两类系统都兼容的驱动程序。微软公司还提供了一个用于帮助驱动程序开发人员编写符合模型的驱动程序的开发工具箱（Windows驱动程序开发工具箱）。大部分Windows驱动程序的开发过程都是先复制一份合适的简单的驱动程序，然后修改它。

微软公司也提供一个*驱动程序验证器*，用以验证驱动程序的多个行为以确保驱动程序符合Windows驱动程序模型的结构要求和I/O请求的协议要求、内存管理等。操作系统中带有此验证器，管理员可能通过运行verifier.exe来控制驱动程序验证器，验证器允许管理员配置要验证哪些驱动程序以及在怎样的范围（多少资源）内验证这些驱动程序。

即使有所有的驱动程序开发和验证支持，在Windows中写一个简单的驱动程序仍然是非常困难的事情，因此微软建立了一个叫做WDF（Windows驱动程序基础）的包装系统，它运行在WDM顶层，简化了很多更普通的需求，主要和驱动程序与电源管理和即插即用操作之间的正确交互有关。

为了进一步简化编写驱动程序，也为了提高了系统的健壮性，WDF包含UMDF（用户模式驱动程序架构），使用UMDF编写的驱动程序作为在进程中执行的服务。还有KMDF（内核模式驱动程序架构），使用KMDF编写的驱动程序作为在内核中执行的服务，但是也使得WDM中的很多细节变得不可预料。由于底层是WDM，并且WDM提供了驱动程序模型，因此，本节将主要关注WDM。

在Windows中，设备是由设备对象描述的。设备对象也用于描述硬件（例如总线），软件抽象（例如文件系统、网络协议），还可以描述内核扩展（例如病毒过滤器驱动程序）。上面提到的这些设备对象都是由Windows中的设备栈来组织的，见前面的图11-16。

I/O操作从I/O管理器调用可执行API IoCallDriver程序开始，IoCallDriver带有指向顶层设备对象和