write、ioctl和close，以及即插即用和电源操作、参数设置、刷新系统缓冲区等。在Win32层，这些API被包装成接口，向特定的设备提供了更高一级的操作。在底层，这些API打开设备，并执行这些基本类型的操作。即使是对一些元数据的操作，如重命名文件，也没有用专门的系统调用来实现。它们只是特殊的ioctl操作。在我们解释了I/O设备栈和I/O管理器使用的I/O请求包（IRP）之后，读者将对上面的陈述更有体会。

I/O系统调用	描　述
NtCreateFile	打开一个新的或已存在的文件或设备
NtReadFile	从一个文件或设备上读取数据
NtWriteFile	把数据写到一个文件或设备
NtQueryDirectoryFile	请求关于一个目录的信息，包括文件
NtQueryVolumeInformationFile	请求关于一个卷的信息
NtSetVolumeInformationFile	修改卷信息
NtNotifyChangeDirectoryFile	当任何在此目录中或其子目录树中的文件被修改时执行完成
NtQueryInformationFile	请求关于一个文件的信息
NtSetInformationFile	修改文件信息
NtLockFile	给文件中一个区域加锁
NtUnlockFile	解除区域锁
NtFsControlFile	对一个文件进行多种操作
NtFlushBuffersFile	把内存文件缓冲刷新到磁盘
NtCancelIoFile	取消文件上未完成的I/O操作
NtDeviceIoControlFile	对一个设备的特殊操作

图11-37　执行I/O的原生NT API 调用

保持了Windows一贯的通用哲学，原生NT I/O系统调用带有很多参数并包括很多变种。图11-37列出了I/O管理器中主要的系统调用接口。NtCreateFile用于打开已经存在的或者新的文件。它为新创建的文件提供了安全描述符和一个对被请求的访问权限的详细描述，并使得新文件的创建者拥有了一些如何分配磁盘块的控制权。NtReadFile和NtWriteFile需要文件句柄、缓冲区和长度等参数。它们也需要一个明确的文件偏移量的参数，并且允许指定一个用于访问文件锁定区域字节的钥匙。正如上面提到的，大部分的参数都和指定哪一个函数来报告（很可能是异步）I/O操作的完成有关。

NtQuerydirectoryFile是一个在执行过程中访问或修改指定类型对象信息的标准模式的一个例子，在这种模式中存在多种不同的查询API。在本例中，指定类型的对象是指与某些目录相关的一些文件对象。一个参数用于指定请求什么类型的信息，比如目录中的文件名列表，或者是经过扩展的目录列表所需要的每个文件的详细信息。由于它实际上是一个I/O操作，因此它支持所有的报告I/O操作已完成的标准方法。NtQueryVolumeInformationFile很像是目录查询操作，但是与目录查询操作不同的是，它有一个参数是打开的卷的文件句柄，不管这个卷上是否有文件系统。与目录不同的是，卷上有一些参数可以修改，因此这里有了单独用于卷的API NtSetVolumeInformationFile。

NtNotifyChangeDirectoryFile是一个有趣的NT范式的例子。线程可以通过I/O操作来确定对象是否发生了改变（对象主要是文件系统的目录，就像在此例中，也可能是注册表键）。因为I/O操作是异步的，所以线程在调用I/O操作后会立即返回并继续执行，并且只有在修改对象之后线程才会得到通知。未处理的请求作为一个外部的I/O操作，使用一个I/O请求包（IRP）被加入到文件系统的队列中等待。如果想从系统移除一个文件系统卷，给执行过未处理I/O操作的线程的通知就会出问题，因为那些I/O操作正在等待。因此，Windows提供了取消未处理I/O操作的功能，其中包括支持文件系统强行卸载有未处理I/O操作的卷的功能。

NtQueryInformationFile是一个用于查询目录中指定文件的信息的系统调用。还有一个与它相对应的系统调用：NtSetInformationFile。这些接口用于访问和修改文件的各种相关信息，如文件名，类似于加密、压缩、稀疏等文件特征，其他文件属性和详细资料，包括查询内部文件ID或给文件分配一个唯一的二进制名称（对象ID）。