API的都没有提供一个引用机制来处理在用户态的并行多线程之间的句柄使用。从而多线程并发访问句柄会带来竞争条件（race condition）和bug，例如，可能发生一个线程在别的线程使用完特定的句柄之前就把它关闭了。或者多次关闭一个句柄。或者关闭另一个线程仍然在使用的句柄，然后重新打开它指向不同的对象。

也许Windows的API应该被设计为每个类型对象带有一个关闭API，而不是单一的通用NTClose操作。这将至少会减少由于用户态线程关闭了错误的处理而发生错误的频率。另一个解决办法可能是在句柄表中的指针之外再添加一个序列域。

为了帮助程序开发人员在他们的程序中寻找这些类似的问题，Windows有一个应用程序验证，软件开发商能够从Microsoft下载。我们将在11.7节介绍类似的驱动程序的验证器，应用程序验证器通过大量的规则检查来帮助程序员寻找可能通过普通测试无法发现的错误。它也可以为句柄释放列表启用先进先出顺序，以便句柄不会被立即重用（即关闭句柄表通常采用效果较好的LIFO排序）。防止句柄被立即重用的情况发生，在这些转化的情况下操作可能错误地使用一个已经关闭的句柄，这是很容易检测到的。

该设备对象是执行体中一个最重要的和贯穿内核态的对象。该类型是由I/O管理器指定的，I/O管理器和设备驱动是设备对象的主要使用者。设备对象和驱动程序是密切相关的，每个设备对象通常有一个链接指向一个特定的驱动程序对象，它描述了如何访问设备驱动程序所对应的I/O处理例程。

设备对象代表硬件设备、接口和总线，以及逻辑磁盘分区、磁盘卷甚至文件系统、扩展内核，例如防病毒过滤器。许多设备驱动程序都有给定的名称，这样就可以访问它们，而无需打开设备的实例的句柄，如在UNIX中。我们将利用设备对象以说明Parse程序是如何被使用的，如图11-22所示。

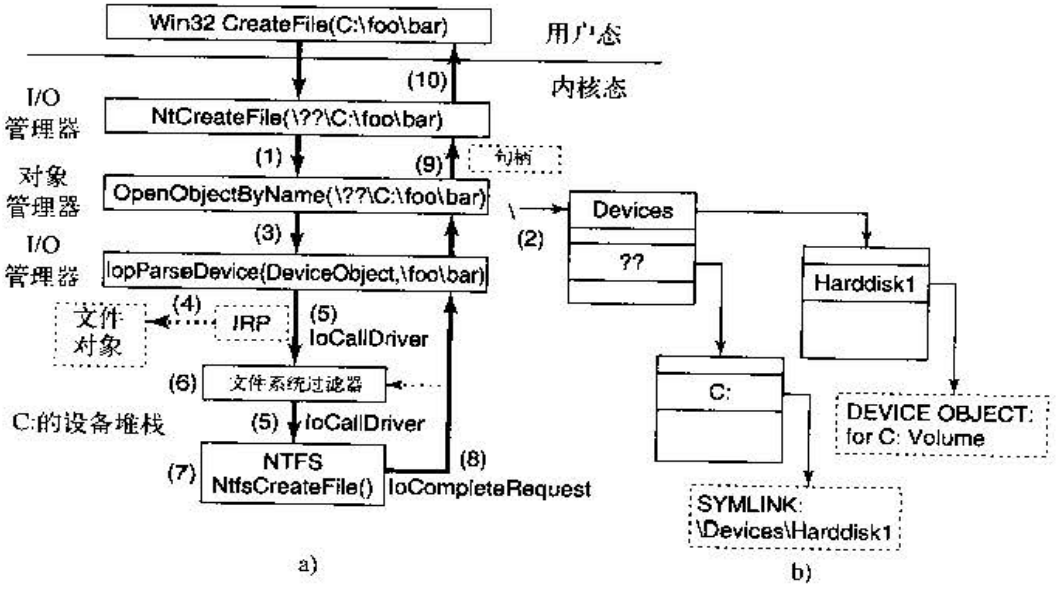

图11-22 I/O和对象管理器创建/打开文件并返回文件句柄的步骤

1）当一个执行组件，如实现了本地系统调用NTCreateFile的I/O管理器，在对象管理器中称之为ObOpenObjectByName，它发送一个NT名字空间的Unicode路径名，例如\??\C:\foo\bar。

2）对象管理器通过目录和符号链接表搜索并最终认定 \??\C: 指的是设备对象（I/O管理器定义的一个类型）。该设备对象在由对象管理器管理的NT名字空间中一个叶节点。

3）然后对象管理器为该对象类型调用Parse程序，这恰好是由I/O管理器实现的IopParseDevice。它不仅传递一个指针给它发现的设备对象（C:），而且还把剩下的字符串\foo\bar也发送过去。

4）I/O管理器将创建一个IRP（I/O请求包），分配一个文件对象，发送请求到由对象管理器确定的设备对象发现的I/O设备堆栈。

5）IRP是在I/O堆栈中逐级传递，直到它到达一个代表文件系统C：实例的设备对象。在每一个阶段，控制是通过一个与这一等级设备对象相连的切入点传递到驱动对象内部。切入点用在这种情况下，是为了支持CREATE操作，因为要求是创建或打开一个名为\foo\bar 的文件。