当然不是只有Windows有内核态对象。UNIX系统也同样支持内核态对象，例如文件、网络数据包、管道、设备、进程、共享内存的IPC设备、消息端口、信号和I/O设备。在UNIX中有各种各样的方式命名和访问对象，例如文件描述符、进程ID、System V IPC对象的整形ID和设备节点。每一类的UNIX对象的实现是特定于其类别的。文件和socket使用不同的设施facility，并且是System V IPC机制、程序、装置之外的。

Windows中的内核对象使用一个的基于NT名字空间中关于对象的句柄和命名统一设备指代内核对象，而且使用一个统一的集中式对象管理器。句柄是进程特定的，但正如上文所述，可以在被另一个进程使用。对象管理器在创建对象时可以给对象命名，可以通过名字打开对象的句柄。

对象管理器在NT名字空间中使用统一的字符编码标准（宽位字符）命名。不同于UNIX，NT一般不区分大小写（它保留大小写但不区分）。NT名字空间是一个分层树形结构的目录，象征联系和对象。

对象管理器提供统一的管理同步、安全和对象生命期的设备。对于对象管理器提供给用户的一般设备是否能为任何特定对象的用户所获得，这是由执行体部件来决定的，它们都提供了操纵每一个对象类型的内部API。

这不仅是应用程序使用对象管理器中的对象。操作系统本身也创建和使用对象——而且非常多。大多数这些对象的创建是为了让系统的某个部分存储相当一段长时间的信息或者将一些数据结构传递给其他的部件，但这都受益于对象管理器对命名和生存周期的支持。例如，当一个设备被发现，一个或多个设备创建代表该设备对象，并在理论上说明该设备如何连接到系统的其他部分。为了控制设备而加载设备的驱动程序，创建驱动程序对象用来保存属性和提供驱动程序所实现的函数的指针，这些函数是实现对I/O请求的处理。操作系统中在以后使用其对象时会涉及这个驱动。驱动也可以直接通过名字来访问，而不是间接的通过它所控制的设备来访问的(例如，从用户态来设置控制它的操作的参数)。

不像UNIX把名字空间的根放在了文件系统中，NT的名字空间则是保留在了内核的虚拟内存中。这意味着NT在每次系统启动时，都得重新创建最上层的名字空间。内核虚拟内存的使用，使得NT可以把信息存储在名字空间里，而不用首先启动文件系统。这也使得NT更加容易地为系统添加新类型的内核态的对象，原因是文件系统自身的格式不需要为每种新类型的目标文件进行改变。

一个命名的目标文件可以标记为永久性的，这意味着这个文件会一直存在，即使在没有进程的句柄指向该对象条件下，除非它被删除或者系统重新启动。这些对象甚至可以通过提供parse例程来扩展NT的名字空间，这种例程方式类似于允许对象具有UNIX中挂载点的功能。文件系统和注册表使用这个工具在NT的名字空间上挂载卷和储集。访问到一个卷的设备对象即访问了原始卷（raw volume），但是设备对象也可以表明一个卷可以加载到NT名字空间中去。卷上的文件可以通过把卷相关文件名加在卷所对应的设备对象的名称后面来访问。

永久性名字也用来描述同步的对象或者共享内存，因此它们可以被进程共享，避免了当进程频繁启动和停止时来不断重建。设备文件和经常使用的驱动程序会被给予永久性名字，并且给予特殊索引节点持久属性，这些索引节点保存在UNIX的 /dev目录下。

我们将在下一节中描叙纯NT API的更多特征，讨论Win32 API在NT 系统调用的封装性。

11.2.2 Win32应用编程接口

Win32函数调用统称为Win32 API接口。这些接口已经被公布并且详细地写在了文档上。这些接口在调用的时候采用库文件链接流程：通过封装来完成原始NT系统调用，有些时候也会在用户态下工作。虽然原始API没有公布，但是这些API的功能可以通过公布的Win32 API来调用实现。随着新的Windows版本的更新，更多的API函数相应增加，但是原先存在的API调用确很少改变，即使Windows进行了升级。

图11-10表示出各种级别的Win32 API调用以及它们封装的原始API调用。最有趣的部分是关于图上令人乏味的映射。许多低级别的Win32函数有相对应的原始NT函数，这一点都不奇怪，因为Win32就是为原始NT API设计的。在许多例子中，Win32函数层必须利用Win32的参数传递给NT内核函数。例如，规范路径名并且映射到NT内核路径，包括特殊的MS-DOS设备（如LPT:）。当创建进程和线程时，使用的Win32 API函数必须通知Win32子系统进程csrss.exe，告知它有新的进程和线程需要它来监督，就像我们在11.4节里描述的那样。