一些Win32调用使用路径名，然而相关的NT内核调用使用句柄。所以这些封装流程包括打开文件，调用NT内核，最后关闭句柄。封装流程同时包括把Win32 API从ANSI编码变成Unicode编码。在图11-10的Win32函数里使用字符串作参数的实际上是两套API，例如参数CreateProcessW和CreateProcessA。当这些参数要传递到下一个API时，这些字符串必须翻译成Unicode编码，因为NT内核调用只认识Unicode。

Win32 调用	本地NT API调用
CreateProcess	NtCreateProcess
CreateThread	NtCreateThread
SuspendThread	NtSuspendThread
CreateSemaphore	NtCreateSemaphore
ReadFile	NtReadFile
DeleteFile	NtSetInformationFile
CreateFileMapping	NtCreateSection
VirtualAlloc	NtAllocateVirtualMemory
MapViewOfFile	NtMapViewOfSection
DumplicateHandle	NtDuplicateObject
CloseHandle	NtClose

图11-10 Win32 API调用以及它们所包含的本地NT API调用示例

因为已经存在的Win32接口很少随着操作系统的改变而改变，所以从理论上说能在前一个版本系统上运行的程序也能正常地在新版本的系统上运行。可在实际情况中，依然经常存在新系统的兼容性问题。Windows太复杂了以至于有些表面上不合逻辑的改动会导致应用程序运行失败。应用程序本身也有问题，例如，它们也经常做细致的操作系统版本检查或者本身就有潜在的问题只不过是在新系统上暴露出来了。然而，微软依旧尽力在每个版本上测试不同的兼容性问题，并且力图提供特定的解决办法。

Windows支持两种特殊环境，一种叫WOW。WOW32通过映射16位字符串到32位，来在32位x86系统用16位Windows 3.x应用程序。同样，WOW64允许32位的程序在x64架构的系统上运行。

Windows API体系不同于UNIX体系。对于后者来说，操作系统函数很简单，只有很少的参数以及很少的方法来执行同样的操作，从而可以有很多途径来完成同样的操作。Win32提供了非常广泛的接口和参数，常常能通过三四种方法来做同样的事情，同时把低级别和高级别的函数混和到一起，例如CreateFile和CopyFile。

这意味着Win32提供了一组非常多的接口，但是这也增加了复杂度，原因是在同一个API中糟糕的系统分层以及高低级别函数的混和。为了学习操作系统，我们仅仅关注那些低级别的函数封装了相关的NT内核的API的Win32 API。

Win32 有创建和管理进程和线程的调用。Win32也有许多进程内部通信的调用，例如创建、销毁、互斥、信号、通信接口和其他IPC实体。

虽然大量的内存管理系统对程序员来说是看不见的，但是一个重要的特征是可见的：即一个进程把文件映射到虚拟内存的一块区域上。这样允许线程可以使用指针来读写部分文件，而不必执行在硬盘和内存之间具体的读写数据操作。通过内存映射，内存系统可以根据需求来执行I/O操作（要求分页）。

Windows 处理内存映射文件使用三种完全不同的手段。第一种，它提供允许进程管理它们自己虚拟空间的接口，包括预留地址范围为以后用。第二种，Win32支持一种称作文件映射的抽象，这用来代替可定位的实体，如文件（文件的映射在NT的层次中称作section）。通常，文件映射是使用文件句柄来关联文件。但有时候也用来指向分页系统中的私有页面。

第三种方法是把文件映射的视图映射到一个进程的地址空间。Win32仅仅允许为当前进程创建一个视图，但是NT潜在的手段更加通用，允许为任意你有权限句柄的进程创建视图。和UNIX中的mmap相比，要区分开创建文件映射和把文件映射到地址空间的操作。

在Windows中，文件映射的内核态实体被句柄所取代。就像许多句柄一样，文件映射能够被复制到其他进程中去。这些进程中的任意一个能够根据需求映射文件到自己的地址空间中。这对共享进程间的私有内存是非常有用的，而且不必再创建文件来实现。在NT层，文件的映射（sections）也和NT名字空间保持一致，能够通过文件名来访问。

对许多程序来说，一个重要的领域是文件I/O操作。在Win32基本视图中，一个文件仅仅是一组有顺序的字节流。Win32提供超过60种调用来创建和删除文件和目录、打开关闭文件、读写文件、提取设置文件属性、锁定字节流范围以及更多基础操作的功能，这些功能基于文件系统的组织以及文件的各自访