行之前被重新装入内存。尽管没有从图11-31中体现出来，Windows Vista还包含一些原生API函数来允许一个进程访问其他进程的虚拟内存。前提是该进程被给予了控制权，即它拥有一个相应的句柄。

Win32 API 函数	描　述
VirtualAlloc	保留或提交一个区域
VirtualFree	释放或解除提交一个区域
VirtualProtect	改变在一个区域上的读/写/执行保护
VirtualQuery	查询一个区域的状态
VirtualLock	使一个区域常驻内存（即不允许被替换到外存）
VirtualUnlock	使一个区域以正常的方式参与页面替换策略
CreateFileMapping	创建一个文件映射对象并且可以选择是否赋予该对象一个名字
MapViewOfFile	映射一个文件（或一个文件的一个部分）到地址空间中
UnmapViewOfFile	从地址空间中删除一个被映射的文件
OpenFileMapping	打开一个之前创建的文件映射对象

图11-31　Windows中用来管理虚拟内存的主要的Win32 API函数

列出的最后四个API函数是用来管理内存映射文件的。为了映射一个文件，首先必须通过调用CreateFileMapping来创建一个文件映射对象（见图11-23）。这个函数返回一个文件映射对象（即一个内存区对象）的句柄，并且可以选择是否为该操作添加一个名字到Win32地址空间中，从而其他的进程也能够使用它。接下来的两个函数从一个进程的虚拟地址空间中映射或取消映射内存区对象之上的视图。最后一个API能被一个进程用来映射其他进程通过调用CreateFileMapping创建并共享出来的映射，这样的映射通常是为了映射匿名内存而建立的。通过这样的方式，两个或多个进程能够共享它们地址空间中的区域。这一技术允许它们写内容到相互的虚拟内存的受限的区域中。

11.5.3　存储管理的实现

运行在x86处理器上的Windows Vista操作系统为每个进程都单独提供了一个4GB大小的按需分页（demand-paged）的线性地址空间，不支持任何形式的分段。从理论上说，页面的大小可以是不超过64KB的2的任何次幂。但是在Pentium处理器上，页面正常情况下固定地设置成4KB大小。另外，操作系统可以使用4MB的页来改进处理器存储管理单元中的快表（Translation Lookaside Buffer，TLB）的效率。内核以及大型应用程序使用了4MB大小的页面以后，可以显著地提高性能。这是因为快表的命中率提高了，并且访问页表以寻找在快表中没有找到的表项的次数减少了。

调度器选择单个线程来运行而不太关心进程，存储管理器则不同，它完全是在处理进程而不太关心线程。毕竟，是进程而非线程拥有地址空间，而地址空间正是存储管理器所关心的。当虚拟地址空间中的一片区域被分配之后，就像图11-32中进程A被分配了4片区域那样，存储管理器会为它创建一个虚拟地址描述符（Virtual Address Descriptor，VAD）。VAD列出了被映射地址的范围，用来表示作为后备存储的文件以及文件被映射区域起始位置的节区以及权限。当访问第一个页面的时候，创建一个页目录并且把它的物理地址插入进程对象中。一个地址空间被一个VAD的列表所完全定义。VAD被组织成平衡树的形式，从而保证一个特定地址的描述符能够被快速地找到。这个方案支持稀疏的地址空间。被映射的区域之间未使用的地址空间不会使用任何内存中或磁盘上的资源，从这个意义上说，它们是“免费”的。

1. 页面失效处理

当在Windows Vista上启动一个进程的时候，很多映射了程序的EXE和DLL映像文件的页面可能已经在内存中，这是因为它们可能被其他进程共享。映像中的可写页面被标记成写时复制（copy-on-write），使得它们能一直被共享，直到内容要被修改的那一刻。如果操作系统从一次过去的执行中认出了这个EXE，它可能已经通过使用微软称之为超级预读取（SuperFetch）的技术记录了页面引用的模式。超级预读取技术尝试预先读入很多需要的页面到内存中，尽管进程尚未在这些页面上发生页面失效。这一技术通过重叠从磁盘上读入页面和执行映像中的初始化代码，减小了启动应用程序所需的延时。同时，它改进了磁盘的吞吐量，因为使用了超级预读取技术以后，磁盘驱动器能够更轻易地组织对磁盘的读请求