当系统启动时，该交换分区为空，并在内存中以单独的项给出它的起始和大小。在最简单的情况下，当第一个进程启动时，留出与这个进程一样大的交换区块，剩余的为总空间减去这个交换分区。当新进程启动后，它们同样被分配与其核心映像同等大小的交换分区。进程结束后，会释放其磁盘上的交换区。交换分区以空闲块列表的形式组织。更好的算法在第10章里讨论。

与每个进程对应的是其交换区的磁盘地址，即进程映像所保存的地方。这一信息是记录在进程表里的。写回一个页面时，计算写回地址的过程很简单：将虚拟地址空间中页面的偏移量加到交换区的开始地址。但在进程启动前必须初始化交换区，一种方法是将整个进程映像复制到交换区，以便随时可将所需内容装入，另一种方法是将整个进程装入内存，并在需要时换出。

但这种简单模式有一个问题：进程在启动后可能增大，尽管程序正文通常是固定的，但数据有时会增长，堆栈也总是在随时增长。这样，最好为正文、数据和堆栈分别保留交换区，并且允许这些交换区在磁盘上多于一个块。

另一个极端的情况是事先什么也不分配，在页面换出时为其分配磁盘空间，并在换入时回收磁盘空间，这样内存中的进程不必固定于任何交换空间。其缺点是内存中每个页面都要记录相应的磁盘地址。换言之，每个进程都必须有一张表，记录每一个页面在磁盘上的位置。这两个方案如图3-29所示。

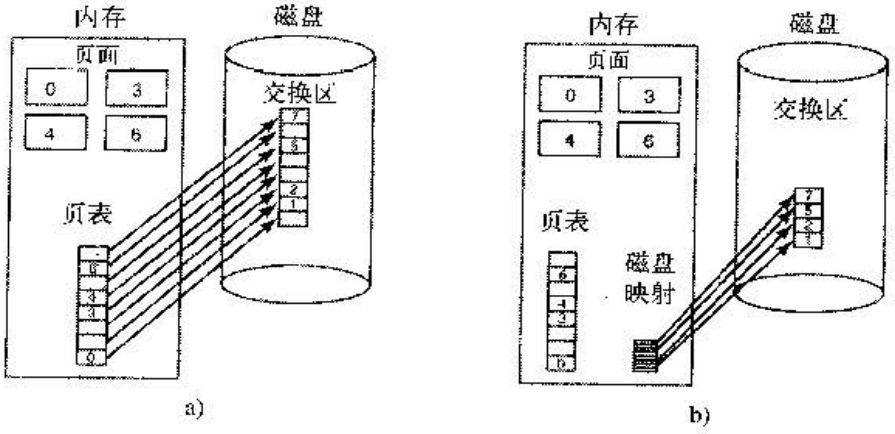

图3-29 a) 对静态交换区分页；b) 动态备份页面

在图3-29a中，有一个带有8个页面的页表。页面0、3、4和6在内存中。页面1、2、5和7在磁盘上。磁盘上的交换区与进程虚拟地址空间（8页面）一样大，每个页面有固定的位置，当它从内存中被淘汰时，便写到相应位置。该地址的计算需要知道进程的分页区域的起始位置，因为页面是按照它们的虚拟页号的顺序连续存储的。内存中的页面通常在磁盘上有镜像副本，但是如果页面装入后被修改过，那么这个副本就可能是过期的了。内存中的深色页面表示不在内存，磁盘上的深色页面（原则上）被内存中的副本所替代，但如果有一个内存页面要被换回磁盘并且该页面在装入内存后没有被修改过，那么将使用磁盘中（深色）的副本。

在图3-29b中，页面在磁盘上没有固定地址。当页面换出时，要及时选择一个空磁盘页面并据此来更新磁盘映射（每个虚拟页面都有一个磁盘地址空间）。内存中的页面在磁盘上没有副本。它们在磁盘映射表中的表项包含一个非法的磁盘地址或者一个表示它们未被使用的标记位。

不能保证总能够实现固定的交换分区。例如，没有磁盘分区可用时。在这种情况下，可以利用正常文件系统中的一个或多个较大的、事前定位的文件。Windows就使用这个方法。然而，可以利用优化方法减少所需的磁盘空间量。既然每个进程的程序正文来自文件系统中某个（可执行的）文件，这个可执行文件就可用作交换区。而更好的方法是，由于程序正文通常是只读的，当内存资源紧张、程序页不得不移出内存时，尽管丢弃它们，在需要的时候再从可执行文件读入即可。共享库也可以用这个方式工作。

3.6.6 策略和机制的分离

控制系统复杂度的一种重要方法就是把策略从机制中分离出来。通过使大多数存储管理器作为用户级进程运行，就可以把该原则应用到存储管理中。在Mach（Young 等人，1987）中首先应用了这种分