机制的思想是：进程可以通过发起一个系统调用，将一个文件映射到其虚拟地址空间的一部分。在多数实现中，在映射共享的页面时不会实际读入页面的内容，而是在访问页面时才会被每次一页地读入，磁盘文件则被当作后备存储。当进程退出或显式地解除文件映射时，所有被改动的页面会被写回到文件中。

内存映射文件提供了一种I/O的可选模型。可以把一个文件当作一个内存中的大字符数组来访问，而不用通过读写操作来访问这个文件。在一些情况下，程序员发现这个模型更加便利。

如果两个或两个以上的进程同时映射了同一个文件，它们就可以通过共享内存来通信。一个进程在共享内存上完成了写操作，此刻当另一个进程在映射到这个文件的虚拟地址空间上执行读操作时，它就可以立刻看到上一个进程写操作的结果。因此，这个机制提供了一个进程之间的高带宽通道，而且这种应用很普遍（甚至扩展到用来映射无名的临时文件）。很显然，如果内存映射文件可用，共享库就可以使用这个机制。

3.5.8 清除策略

如果发生缺页中断时系统中有大量的空闲页框，此时分页系统工作在最佳状态。如果每个页框都被占用，而且被修改过的话，再换入一个新页面时，旧页面应首先被写回磁盘。为保证有足够的空闲页框，很多分页系统有一个称为*分页守护进程*（paging daemon）的后台进程，它在大多数时候睡眠，但定期被唤醒以检查内存的状态。如果空闲页框过少，分页守护进程通过预定的页面置换算法选择页面换出内存。如果这些页面装入内存后被修改过，则将它们写回磁盘。

在任何情况下，页面中原先的内容都被记录下来。当需要使用一个已被淘汰的页面时，如果该页框还没有被覆盖，将其从空闲页框缓冲池中移出即可恢复该页面。保存一定数目的页框供给比使用所有内存并在需要时搜索一个页框有更好的性能。分页守护进程至少保证了所有的空闲页框是“干净”的，所以空闲页框在被分配时不必再急着写回磁盘。

一种实现清除策略的方法就是使用一个双指针时钟。前指针由分页守护进程控制。当它指向一个脏页面时，就把该页面写回磁盘，前指针向前移动。当它指向一个干净页面时，仅仅指针向前移动。后指针用于页面置换，就像在标准时钟算法中一样。现在，由于分页守护进程的工作，后指针命中干净页面的概率会增加。

3.5.9 虚拟内存接口

到现在为止，所有的讨论都假定虚拟内存对进程和程序员来说是透明的，也就是说，它们都可以在一台只有较少物理内存的计算机上看到很大的虚拟地址空间。对于不少系统而言这样做是对的，但对于一些高级系统而言，程序员可以对内存映射进行控制，并可以通过非常规的方法来增强程序的行为。这一节我们将简短地讨论一下这些问题。

允许程序员对内存映射进行控制的一个原因就是为了允许两个或者多个进程共享同一部分内存。如果程序员可以对内存区域进行命名，那么就有可能实现共享内存。通过让一个进程把一片内存区域的名称通知另一个进程，而使得第二个进程可以把这片区域映射到它的虚拟地址空间中去。通过两个进程（或者更多）共享同一部分页面，高带宽的共享就成为可能——一个进程往共享内存中写内容而另一个从中读出内容。

页面共享也可以用来实现高性能的消息传递系统。一般地，传递消息的时候，数据被从一个地址空间复制到另一个地址空间，开销很大。如果进程可以控制它们的页面映射，就可以这样来发送一条消息：发送进程清除那些包含消息的页面的映射，而接收进程把它们映射进来。这里只需要复制页面的名字，而不需要复制所有数据。

另外一种高级存储管理技术是*分布式共享内存*（Feeley等人，1995；Li，1986；Li 和 Hudak，1989；Zekauskas 等人，1994）。该方法允许网络上的多个进程共享一个页面集合，这些页面可能（而不是必要的）作为单个的线性共享地址空间。当一个进程访问当前还没有映射进来的页面时，就会产生缺页中断。在内核空间或者用户空间中的缺页中断处理程序就会对拥有该页面的机器进行定位，并向它发送一条消息，请求它清除该页面的映射，并通过网络发送出来。当页面到达时，就把它映射进来，并重新开始运行引起缺页中断的指令。在第8章中我们将详细讨论分布式共享内存。