所有计算机也是愚蠢的。

现在回到空闲表方法，只需要在内存中保存一个指针块。当文件创建时，所需要的块从指针块中取出。现有的指针块用完时，从磁盘中读入一个新的指针块。类似地，当删除文件时，其磁盘块被释放，并添加到内存的指针块中。当这个块填满时，就把它写入磁盘。

在某些特定情形下，这个方法产生了不必要的磁盘I/O。考虑图4-23a中的情形，内存中的指针块只有两个表项了。如果释放了一个有三个磁盘块的文件，该指针块就溢出了，必须将其写入磁盘，这就产生了图4-23b的情形。如果现在写入含有三个块的文件，满的指针块不得不再次读入，这将回到图4-23a的情形。如果有三个块的文件只是作为临时文件被写入，当它被释放时，就需要另一个磁盘写操作，以便把满的指针块写回磁盘。总之，当指针块几乎为空时，一系列短期的临时文件就会引起大量的磁盘I/O。

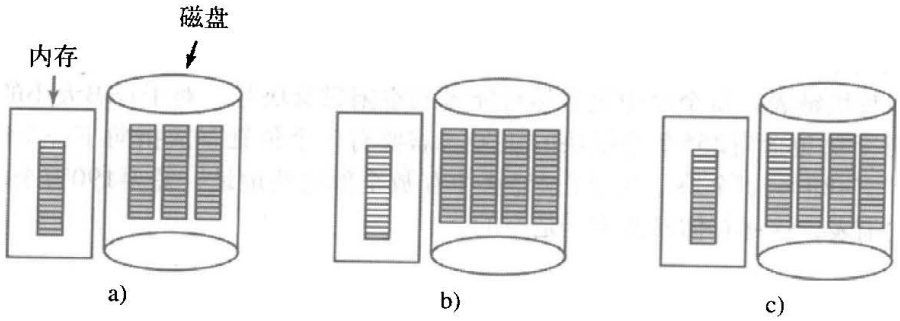

图4-23 a) 在内存中一个被指向空闲磁盘块的指针几乎充满的块，以及磁盘上三个指针块；b) 释放一个有三个块的文件的结果；c) 处理该三个块的文件的替代策略（带阴影的表项代表指向空闲磁盘块的指针）

一个可以避免过多磁盘I/O的替代策略是，拆分满了的指针块。这样，当释放三个块时，不再是从图4-23a变化到图4-23b，而是从图4-23a变化到图4-23c。现在，系统可以处理一系列临时文件，而不需进行任何磁盘I/O。如果内存中指针块满了，就写入磁盘，半满的指针块从磁盘中读入。这里的思想是：保持磁盘上的大多数指针块为满的状态（减少磁盘的使用），但是在内存中保留一个半满的指针块。这样，它可以既处理文件的创建又同时处理文件的删除操作，而不会为空闲表进行磁盘I/O。

对于位图，在内存中只保留一个块是有可能的，只有在该块满了或空了的情形下，才到磁盘上取另一块。这样处理的附加好处是，通过在位图的单一块上进行所有的分配操作，磁盘块会较为紧密地聚集在一起，从而减少了磁盘臂的移动。由于位图是一种固定大小的数据结构，所以如果内核是（部分）分页的，就可以把位图放在虚拟内存内，在需要时将位图的页面调入。

3. 磁盘配额

为了防止人们贪心而占有太多的磁盘空间，多用户操作系统常常提供一种强制性磁盘配额机制。其思想是系统管理员分给每个用户拥有文件和块的最大数量，操作系统确保每个用户不超过分给他们的配额。下面将介绍一种典型的机制。

当用户打开一个文件时，系统找到文件属性和磁盘地址，并把它们送入内存中的打开文件表。其中一个属性告诉文件所有者是谁。任何有关该文件大小的增长都记到所有者的配额上。

第二张表包含了每个用户当前打开文件的配额记录，即使是其他人打开该文件也一样。这张表如图4-24所示，该表的内容是从被打开文件的所有者的磁

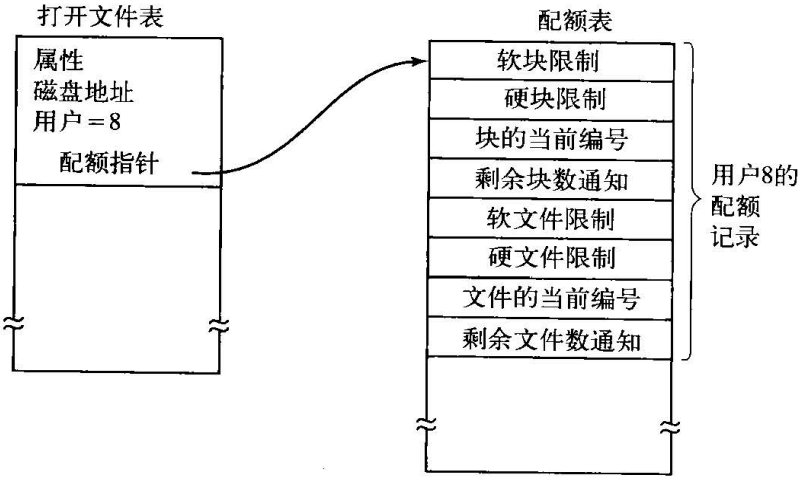

图4-24 在配额表中记录了每个用户的配额