4. i节点

最后一个记录各个文件分别包含哪些磁盘块的方法是给每个文件赋予一个称为i节点（index-node）的数据结构，其中列出了文件属性和文件块的磁盘地址。图4-13中是一个简单例子的描述。给定i节点，就有可能找到文件的所有块。相对于在内存中采用表的方式而言，这种机制具有很大的优势，即只有在对应文件打开时，其i节点才在内存中。如果每个i节点占有n个字节，最多k个文件同时打开，那么为了打开文件而保留i节点的数组所占据的全部内存仅仅是kn个字节。只需要提前保留少量的空间。

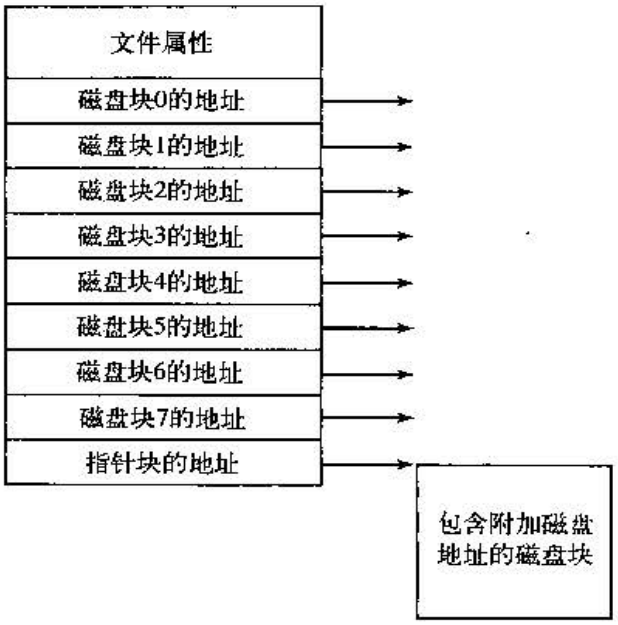

图4-13 i节点的例子

这个数组通常比上一节中叙述的文件分配表（FAT）所占据的空间要小。其原因很简单，保留所有磁盘块的链接表的表大小正比于磁盘自身的大小。如果磁盘有n块，该表需要n个表项。由于磁盘变得更大，该表格也线性随之增加。相反，i节点机制需要在内存中有一个数组，其大小正比于可能要同时打开的最大文件个数。它与磁盘是10GB、100GB还是1000GB无关。

i节点的一个问题是，如果每个i节点只能存储固定数量的磁盘地址，那么当一个文件所含的磁盘块的数目超出了i节点所能容纳的数目怎么办？一个解决方案是最后一个“磁盘地址”不指向数据块，而是指向一个包含磁盘块地址的块的地址，如图4-13所示。更高级的解决方案是：可以有两个或更多个包含磁盘地址的块，或者指向其他存放地址的磁盘块的磁盘块。在后面讨论UNIX时，我们还将涉及i节点。

4.3.3 目录的实现

在读文件前，必须先打开文件。打开文件时，操作系统利用用户给出的路径名找到相应目录项。目录项中提供了查找文件磁盘块所需要的信息。因系统而异，这些信息有可能是整个文件的磁盘地址（对于连续分配方案）、第一个块的编号（对于两种链表分配方案）或者是i节点号。无论怎样，目录系统的主要功能是把ASCII文件名映射成定位文件数据所需的信息。

与此密切相关的问题是在何处存放文件属性。每个文件系统维护诸如文件所有者以及创建时间等文件属性，它们必须存储在某个地方。一种显而易见的方法是把文件属性直接存放在目录项中。很多系统确实是这样实现的。这个办法用图4-14a说明。在这个简单设计中，目录中有一个固定大小的目录项列表，每个文件对应一项，其中包含一个（固定长度）文件名、一个文件属性结构以及用以说明磁盘块位置的一个或多个磁盘地址（至某个最大值）。

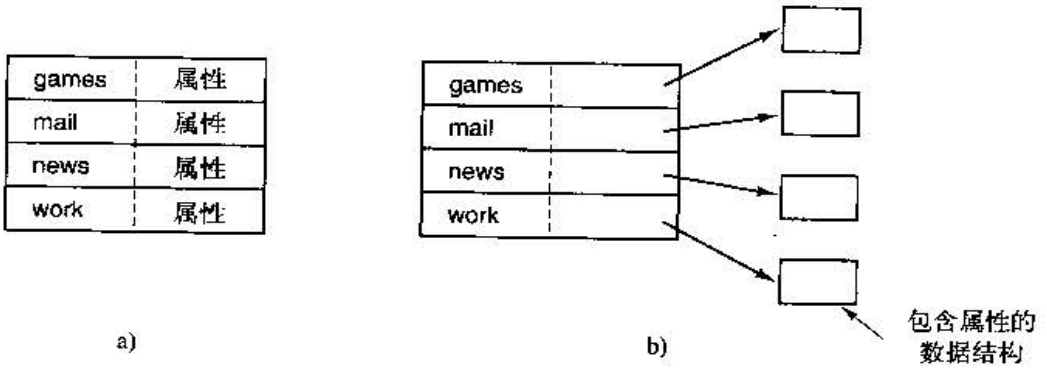

图4-14 a) 简单目录，包含固定大小的目录项，在目录项中有磁盘地址和属性；b) 每个目录项只引用i节点的目录

对于采用i节点的系统，还存在另一种方法，即把文件属性存放在i节点中而不是目录项中。在这种