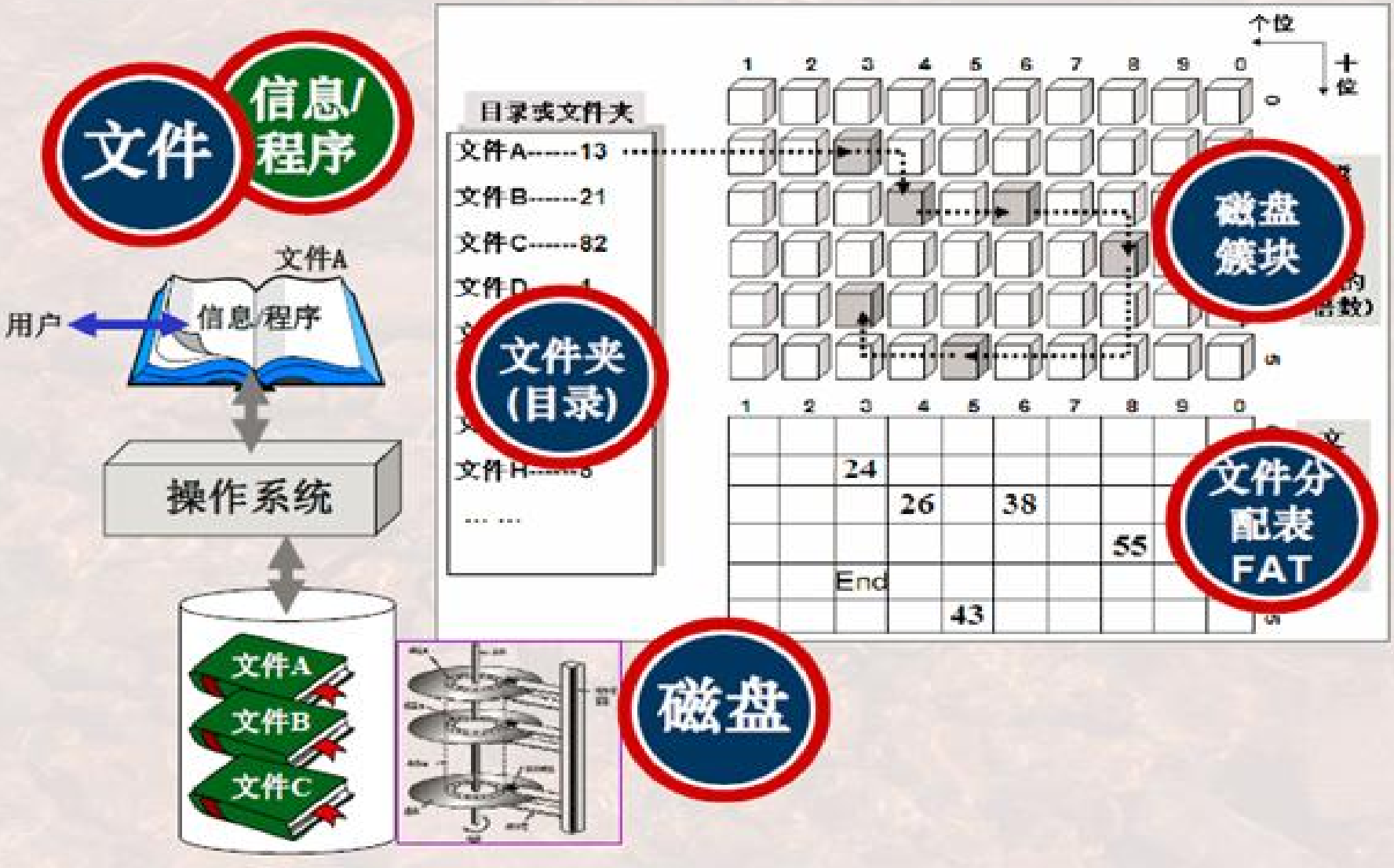
第八章 磁盘存储器的管理

回顾：OS是怎么管理磁盘的？



8.1 外存的组织形式（文件物理结构）

8.1.1 连续分配

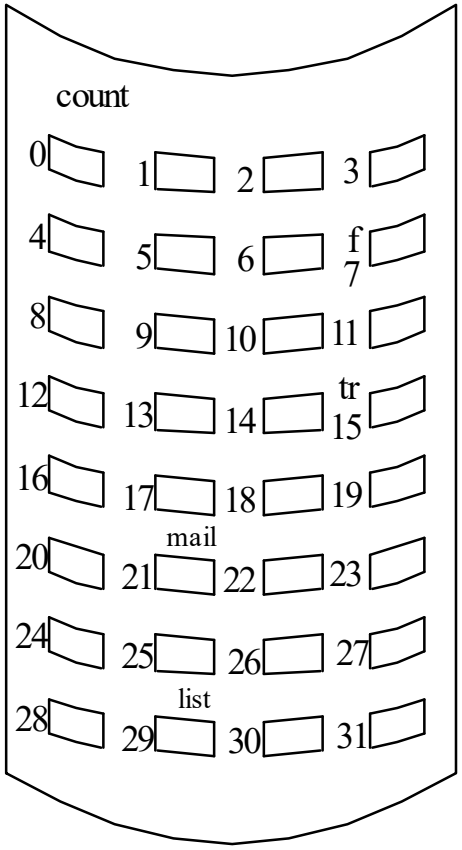
连续分配（磁带，磁盘都可采用）

- 每个文件分配一组相邻盘块。
- 特点：简单

  （1）顺序访问容易且速度快，因磁头移动距离小。

  （2）要求连续空间，一段时间后需整理磁盘以消除外部碎片。

  （3）文件不易动态增长和删除。
- 文件对应目录项（属性）中包含：
  - 始址、总块数。



目录

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

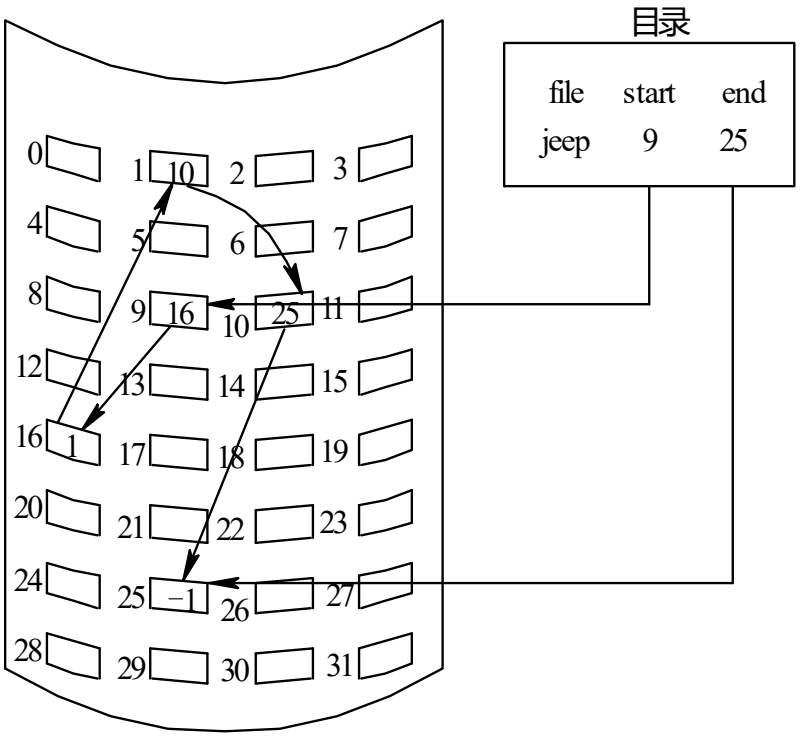
连续结构支持直接访问

8.1.2 链接分配

- 文件离散地分配于各盘块中，以提高外存利用率，文件长度可变，易于增删，只能顺序存取。

1. 隐式链接

- 文件目录表中有start块号，每块中有下一块号。
- 特点：只适合于顺序访问，对随机访问效率低，可靠性差。



隐式链接不支持直接访问

连续分配方式与隐式链接分配方式比较：

某文件由100条记录组成，记录从1开始编号，现欲插入一条新的记录，作为其第10条记录。

（1）若文件系统采用连续分配方式，每个磁盘块存放1条记录，最少需要访问多少次磁盘块？

（2）若文件系统采用隐式链接分配方式，每个磁盘块存放一条记录和一个链接指针，则完成上述插入操作最少需要访问多少次磁盘块？

思考：两种分配方式的区别？

连续分配方式与隐式链接分配方式比较：

解答：

（1）连续分配方式，插入第10条记录，要把文件前9条记录前移，每条记录读出和存回各为访盘1次，需9*2=18次，写第10条记录为1次，共19次。

（2）隐式链接分配是离散存放方式，插入新的记录无需移动其它记录，插入第10条记录，需要访问前9条记录，找到第9条记录的指针，需访盘9次；然后把第9条记录的链接指针赋给新的第10条记录，把第10条记录存到磁盘需访盘1次；接着将第9条记录的链接指针存回第10条记录的磁盘块需访盘1次；合计共需访盘11次。

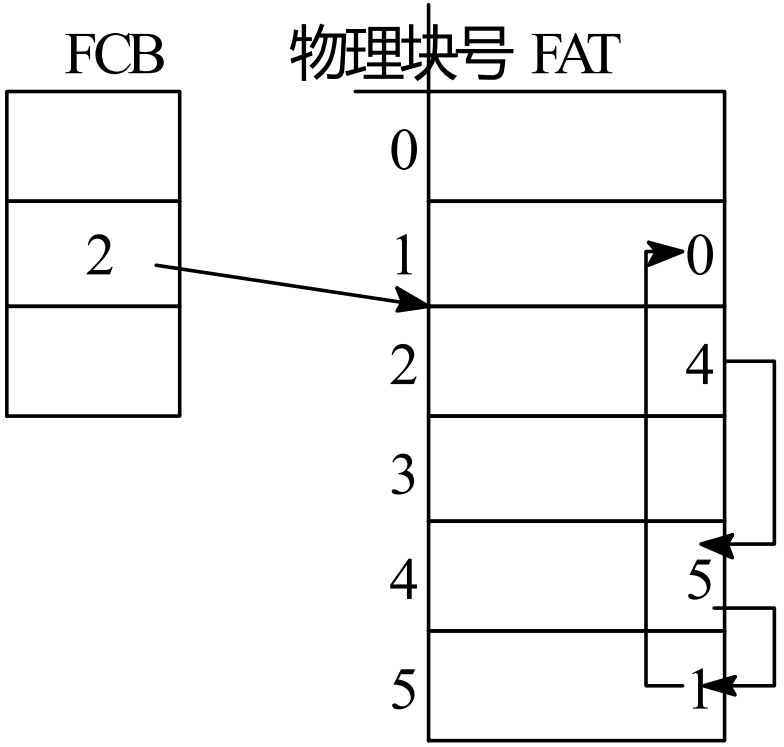
8.1.2 链接分配

2. 显式链接：把用于链接的指针显式存放在内存的一张表中，查找在内存中进行。

①FAT是整个磁盘仅设置一张，表的序号是物理盘块号，从0至N-1，每个表项存放链接指针，即下一盘块号。

②文件目录中相应目录项中的物理地址即第一盘块号。

③查找记录过程在内存中进行，大大提高速度。



显式链接可支持直接访问

计算FAT表的开销

例子：假定磁盘块的大小为1K，对于540M的磁盘，其文件分配表FAT需要占用多少存储空间？当硬盘容量为1.2G时，FAT需要占用多少空间？

解：硬盘大小540M，磁盘块大小1K，所以硬盘共有盘块：

540M/1K=540K(个)

又 512K<540K<1024K(即2^{19}<540K<2^{20})

故540K个盘块号要用20位二进制表示，即FAT表的每个表目为2.5字节。FAT要占用的存储空间总数为：

2.5*540K=1350K

当硬盘大小为1.2G,硬盘共有盘块：1.2G/1K=1.2M(个)

又 1M<1.2M<2M (即2^{20}<1.2M<2^{21})

故1.2M个盘块号要用21位二进制表示。为了方便文件分配表的存取，每个表目用24位表示，即3个字节。FAT要占用的存储空间总数为：

3*1.2M=3.6M

8.1.3 FAT技术

★概念

微软公司早、中期推出的操作系统一直都是采用的FAT技术最早使用的是12位的FAT12，后来为16位的FAT16、32位的FAT32。Windows NT、Windows 2000和Windows XP操作系统为新技术文件系统NTFS。

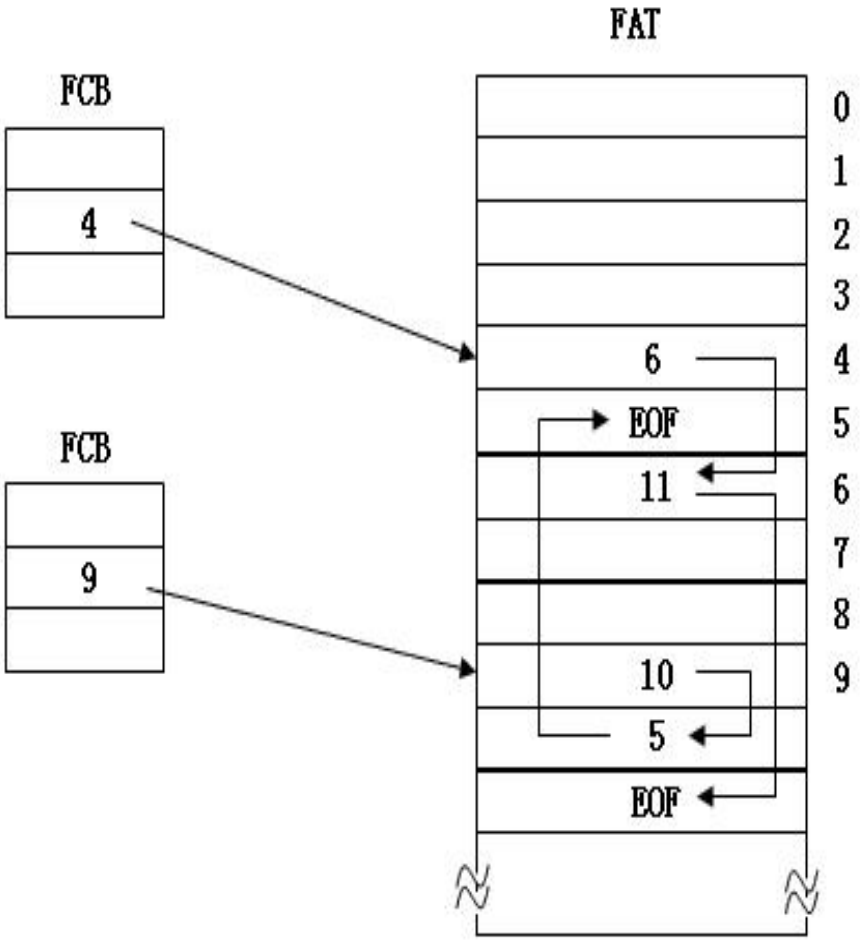
★卷

- 也称为分区，在FAT中，支持将一个物理磁盘分成四个逻辑磁盘，每个逻辑磁盘就是一个卷。
- 每个卷都专门划出一个单独区域来存放自己的目录和FAT表，以及自己的逻辑驱动器字母，如：“C：”、“D：”等。

1. FAT12

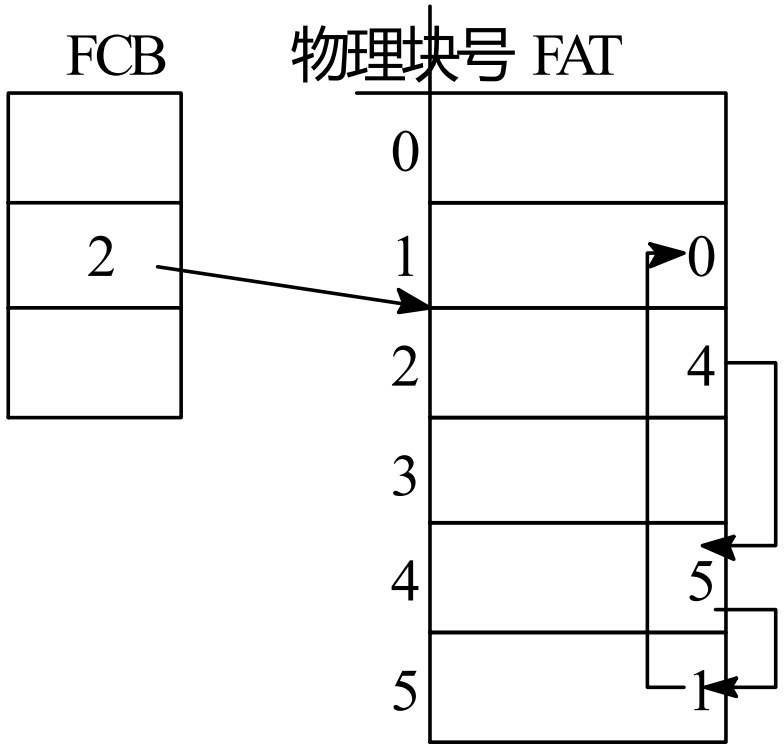
早期的FAT12文件系统

FAT12是以盘块为基本分配单位，在每个分区中都配有两张相同的文件分配表FAT1和FAT2。FAT的每个表项中存放一下一个盘块号，而将文件的第一个盘块号放在自己的FCB中。



问题--：最大磁盘容量的限制

- 每个FAT表项为12位，在FAT表中最多允许有2^{12}个表项；
- 如果采用以盘块作为基本分配单位，每个盘块的大小一般是512字节，则每个磁盘分区的容量为2MB（ 2^{12} ×512B）；
- 一个物理磁盘能支持4个逻辑磁盘分区，所以相应的磁盘最大容量仅为8MB。



计算FAT表的开销？

解决方法：

- 引入一个新的分配单位——簇
  - 簇是一组相邻的扇区，盘块分配时以簇作为分配的基本单位。
  - 簇的大小一般是2n（n为整数）个盘块；
  - **优点**：FAT表的存取开销减少，可管理的磁盘容量增大。
  - **缺点**：相应的**簇内碎片**也将随之成倍地增加。

如何管理一个更大的磁盘？

- *块数—FAT表的表项数*
- *块的大小—簇的大小*

2. FAT16

- 将FAT表增至16位，最大表项数将增至65536(2^{16})，此时便能将一个磁盘分区分为65536个簇。
- FAT16的每个簇中可以有的盘块数为4、8、…，直到64，由此得出FAT16可以管理的最大分区空间为 $2^{16}\times64\times512=2048MB=2GB$。
- 当磁盘容量迅速增加时，簇的容量还必须增大，**簇内碎片**所造成的浪费也增大。

例如，当要求磁盘分区的大小为8GB时，则每个簇的大小达到128KB，这意味着内部零头最大可达到128KB。一般而言，对于1GB～4GB的硬盘来说，大约会浪费10%～20%空间。

★解决方法 微软推出了FAT32。

3. FAT32

- FAT32是FAT系列文件系统的最后一个产品。每一簇在FAT表中的表项占据4字节，允许管理比FAT16更多的簇，允许采用较小的簇。
- 事实上，FAT32只用了28位的磁盘地址，当每个簇为4KB，FAT32可以管理的单个最大磁盘空间为4KB×2^{28}=1TB。
- 理论上，当每个簇为32KB，最大的磁盘空间为32KB×2^{28}=8TB，但实际上，FAT32系统限制为2TB。

★三种FAT类型的最大分区以及所对应的块的大小如图所示：

块大小	FAT12	FAT16	FAT32
0.5KB	2■B		
1KB	4■B		
2KB	8■	128■	
4KB	16■	256■	1TB
8KB		512■	2TB
16KB		1024■	2TB
32KB		2048■	2TB

8.1.4 NTFS的文件组织方式

NTFS新特征

NTFS（New Technology File System）是一个专门为Windows NT开发的、全新的文件系统，并适用于Windows 2000/XP/2003。

- ①使用了64位磁盘地址；
- ②在NTFS中可以很好地支持长文件名，单个文件名限制在255个字符以内，全路径名为32767个字符；
- ③具有系统容错功能，即在系统出现故障或差错时，仍能保证系统正常运行；
- ④能保证系统中的数据一致性，这是一个非常有用的功能；
- ⑤还提供了文件加密、文件压缩等功能。

- 从12位-16位-32位-64位，FAT表的存储开销越来越大；
- 访问一个文件需要将FAT表调入内存，进行检索，但是检索涉及到的范围很小；
- ***如何改进？***

文件类型	表项数	表项大小	FAT 表的大小	磁盘大小
FAT12	2^{12}	1.5B	6KB	16MB
FAT16	2^{16}	2B	2^{17}B=128KB	2048MB
FAT32	2^{32}	4B	2^{34}B=16GB	2TB

8.1.5 索引分配

1. 单级索引

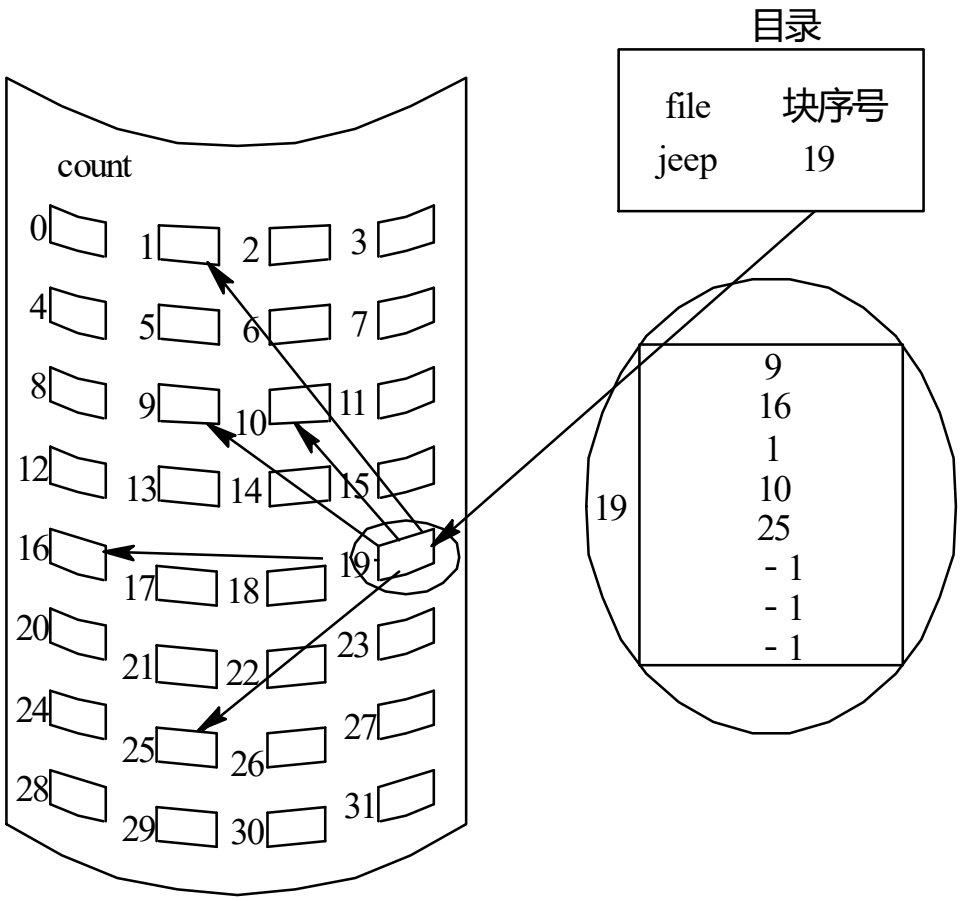
- 链接分配问题：
  - 不能高效直接存取；
  - FAT需占较大的内存。
- 概念：为每个文件分配一个索引块

特点：

（1）文件较大时有利。文件较小时浪费外存空间（还需为小文件建索引块）

（2）当文件较大时，索引块太多，查找速度减慢.

解决：当索引太大时，则需建立多级索引



索引结构可支持直接访问

8.1.5 索引分配

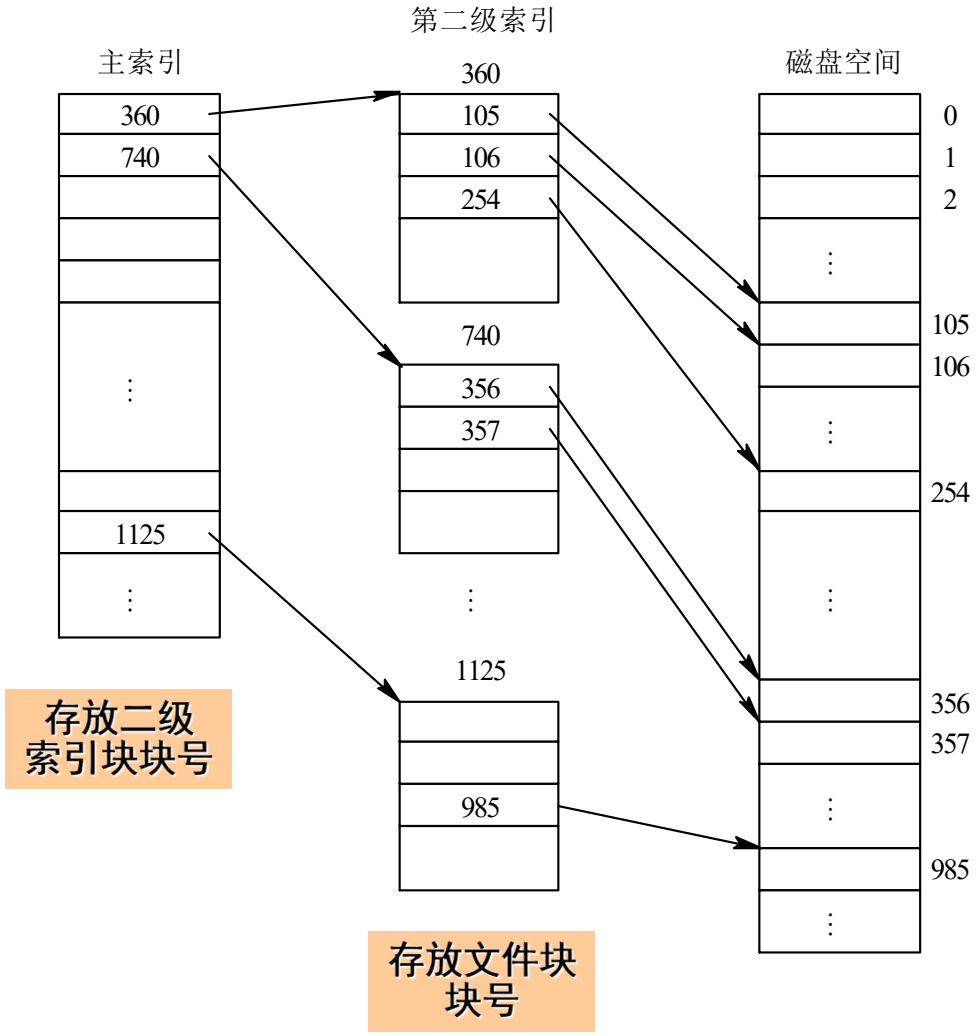
2. 多级索引

两级：

- 设一个盘块大小为4KB，每个盘块号占4B，则一个盘块可存1K个块号。
- 两级索引存放的文件的盘块号总数为：
  1K×1K=1M(个)

故文件的最大长度为

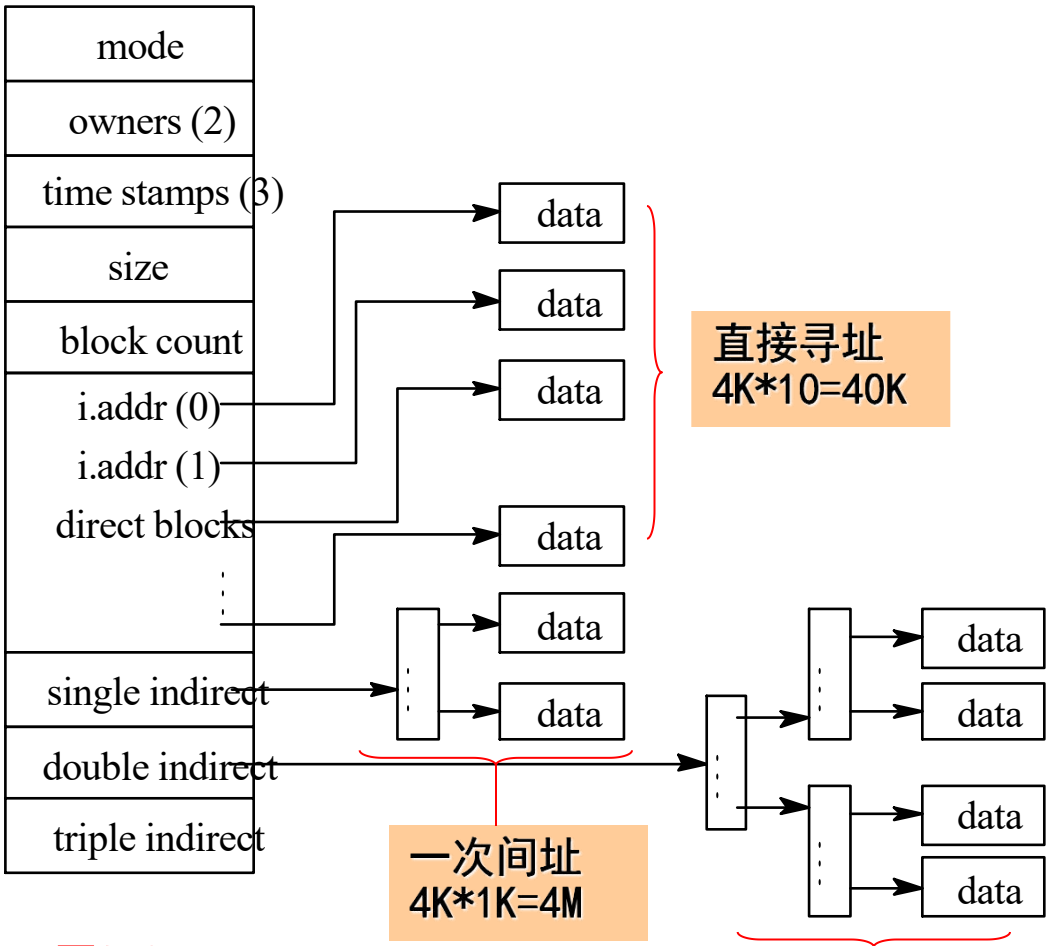
1M×4K=4GB



8.1.3 索引分配

3.混合分配方式（UNIX系统）

- 一、二、多级索引合用
- 设每个块大小为4k，一索引项占4字节，则
  1. 直接地址：小文件（<40k)则立即读出。
  2. 一次间址：4M
  3. 多次寻址:4G->4T。



- 可以把盘块号直接放入FCB或inode里面；
- UNIX System V的inode中设13个地址项；

讨论一下两个问题：

1. 对于给定文件，决定采用哪种策略的原则：

•当对文件的访问通常是顺序的，而且文件比较小时，采用连续分配。

•当文件较大且通常是顺序访问的，采用链接结构。

•当文件较大且通常是随机访问的，采用索引分配。

2. 评价三类分配方法速度方面的特性：

•连续分配最快

•链接分配较慢，因为磁头可能不得不在存取文件之间移动。

•索引分配最慢，除非任何时刻，整个索引都放在内存中，否则，必须花一定时间去存取文件索引的下一块。

练习：考虑一个存于盘上的文件系统，其中的文件由大小为512B的块组成。假定每一个文件有一个文件目录项，该目录项包含该文件的文件名，文件的长度以及第一块（或第一索引块）和最后一块的位置。而且，该目录项位于内存。对于索引分配策略，该目录项指明第一索引块，该索引块又依次指向511个文件块且有一指向下一索引块的指针。对连续、隐式链接和索引分配策略中的一种。

（1）说明在这个系统是如何实现逻辑地址到物理地址的映射？

（2）如果当前位于逻辑块10且希望访问逻辑块4，问必须从盘上读多少文件块？

答：令m是起始块号，逻辑地址除以512，得到商x（逻辑块号）和余数y（块内地址）。

•连续分配

(1)将x加上m以获得相应物理块的位移

(2) 1

•链接分配

(1)查链表直至找到所需块号

```
m1=m;
for i:=1 to x
begin
  取位于m1的物理块;
  用下一块地址替代m1的值;
end
```

(2) 4

•索引分配

(1)将第一索引块读入内存，用逻辑块号（x）除以511，设除得的商和余数分别为r和s。

```
for i:=1 to r
  begin
    从当前索引块的地址512获得块地址q;
    将地址q处的索引块送内存;
  end
```

利用s作为该索引块的位移以获得相应物理块的地址。

(2) 1

8.2 文件存储空间的管理

8.2.1 空闲盘块表法和空闲链表法

1. 空闲盘块表法

- 分配：首次/循环首次/最佳/最坏
- 回收：判断是否合并。
- 由于连续分配比较快，因此对兑换空间及小文件的管理适用。

序号	第一空闲盘块号	空闲盘块数
1	2	4
2	9	3
3	15	5
4	—	—

图　空闲盘块表

8.2.1 空闲表法和空闲链表法

2. 空闲链表法

(1) 空闲盘块链：将所有空闲盘块通过指针连接在一起。
优缺点：实现简单，但是工作效率低。在为一个文件分配盘块时，可能要重复操作多次。
(2) 空闲盘区链 ：每个块中包含有尽可能多的空闲块号。

例：对于1k大小的块，考察20M磁盘需要多少块的空闲链表？
分析：对于1K大小的块，20M的磁盘包含的块数：
20M/1K=20K（个） 又 $2^{14}<20K<2^{15}$
故需要15位表示块号，即2个字节。
又一块为1KB，即可包含512个16位的磁盘块号
故20M的磁盘需要20K/512=40个块的空闲块链表。
说明：实际上，每个块中只包含511个空闲块号和下一空闲块结点的块号，通常情况下，我们采用空闲块存放空闲块链表。

8.2.2 位示图法

1. **位示图**：建立一张位示图，以反映整个存储空间的分配情况。用一个二进制位标示块，n个块的磁盘需要n位位图，在位图中，空闲块用1表示，分配块用0表示（或者反之）。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

图 6-21 位示图

8.2.2 位示图法

例子：对于1k大小的块，考察20M的磁盘需要多少块的位图？

分析：对于1K大小的块，20M的磁盘包含的块数：20M/1K=20K（个）

又每块用一位表示

故20M的磁盘需要20K位表示

又1块是1KB即8Kb

故20M的磁盘需要20Kb/8Kb=2.5,即3个块

结论：①位图所需空间比空闲链表少；

②一般来说位图方案较好，但当空闲块很少时，且只有一个块内存空间可用时，则选择链表方案更好。

有一计算机系统利如图所示的位示图来管理空闲盘块。盘块的大小为1KB，现要为某文件分配两个盘块，试说明盘块的具体分配过程。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 分配两个盘块的过程如下：
- (1) 顺序扫描位示图，从中找到第一个值为0的二进制位，得到其行号i=3，列号j=3。
- (2) 将所找到的二进制位转换成与之对应的盘块号。

  盘块号计算公式为：b=（3-1）*16+3=35；
- (3) 修改位示图，令map[3，3]=1，并将该盘块分配出去。
- 类似地，可使用相同的方法找到第二个值为0的二进制位，得到行号i=4，列号j=7，其对应的盘块号为55，令map[i，j]=1，并将该盘块分配出去。

8.2.2 位示图法

2. 盘块的分配

(1)顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位(“0”表示空闲时)。

(2)将所找到的一个或一组二进制位，转换成与之相应的盘块号。位于位示图的第i行、第j列，相应的盘块号应按下式计算：

$$b=n(i-1)+j$$

式中， n代表每行的位数。

(3)修改位示图，

令map [i, j] =1

3. 盘块的回收

(1)将回收盘块的盘块号转换成位示图中的行号和列号。 转换公式为：

$$i=(b-1)\ DIV\ n+1$$

$$j=(b-1)\ MOD\ n+1$$

(2)修改位示图。

令map [i, j] =0

8.2.3 成组链接法

基本思想：

①空闲盘块号栈存放当前可用的一组空闲盘块号及栈中尚有的空闲盘块号数；

②文件区中所有空闲盘块分成若干组；

③每一组含有的盘块总数和所有的盘块号记入前一组最后一个盘块中；

④第一组盘块总数和所有盘块号记入空闲盘块号栈；

⑤最末一组只有N-1个盘块。

8.2.3 成组链接法

1. 空闲盘块的组织

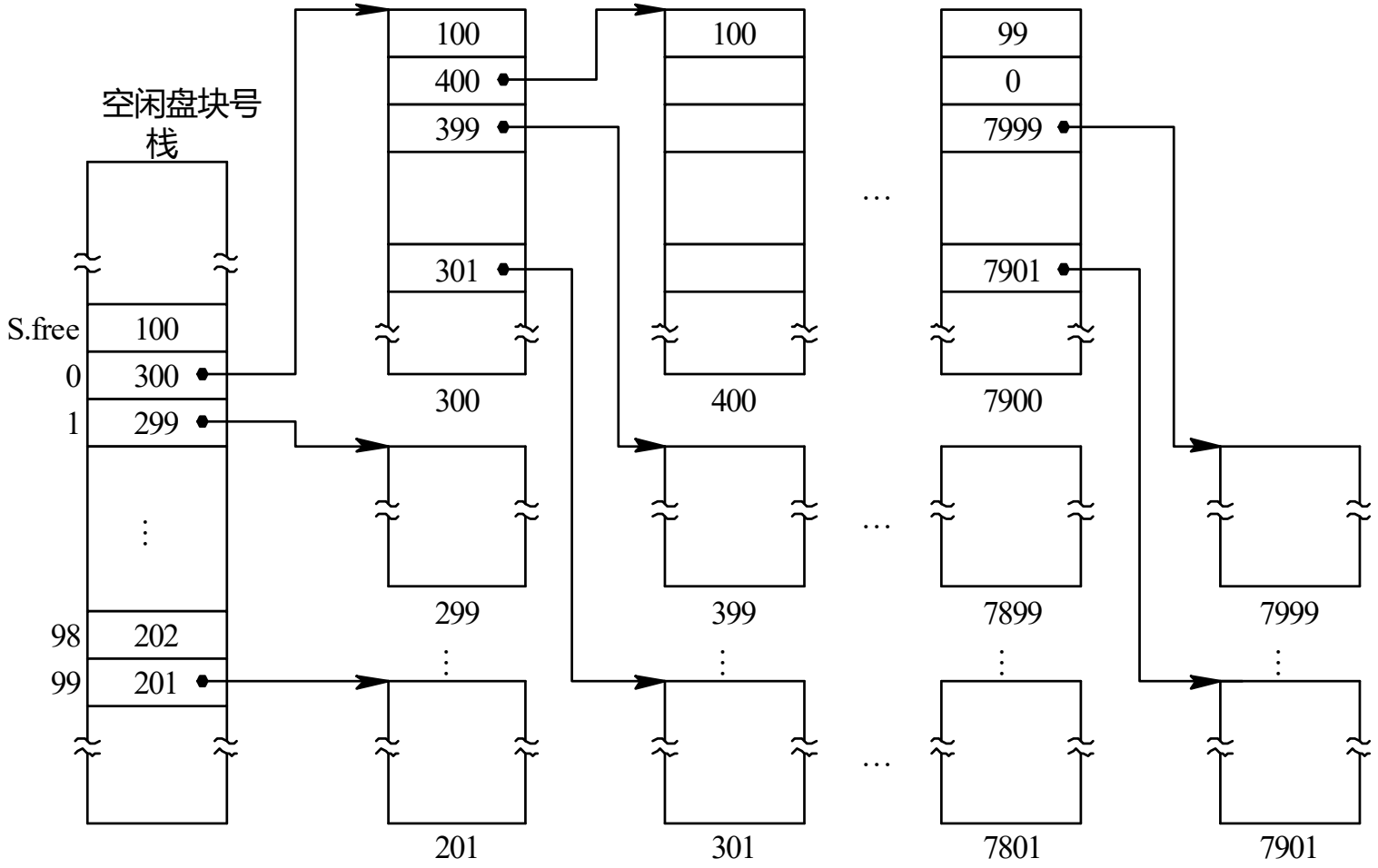


图 6-22 空闲盘块的成组链接法

8.2.3 成组链接法

2. 空闲盘块的分配与回收

分配：

- 首先检查空闲盘块号栈是否上锁，如未上锁，便从栈顶取出一空闲盘块号，将与之对应的盘块分配给用户，然后将栈顶指针下移一格。

- 若该盘块号已是栈底，即S.free(0)，即最后一个可分配的盘块号。须调用磁盘读过程，将栈底盘块号所对应盘块的内容读入栈中，作为新的盘块号栈的内容，然后，把原栈底对应的盘块分配出去(其中的有用数据已读入栈中)。

8.2.3 成组链接法

2. 空闲盘块的分配与回收

回收：

- 将回收盘块的盘块号记入空闲盘块号栈的顶部，并执行空闲盘块数加1操作。

- 当栈中空闲盘块号数目已达100时，表示栈已满，便将现有栈中的100个盘块号，记入新回收的盘块中，再将其盘块号作为新栈底。

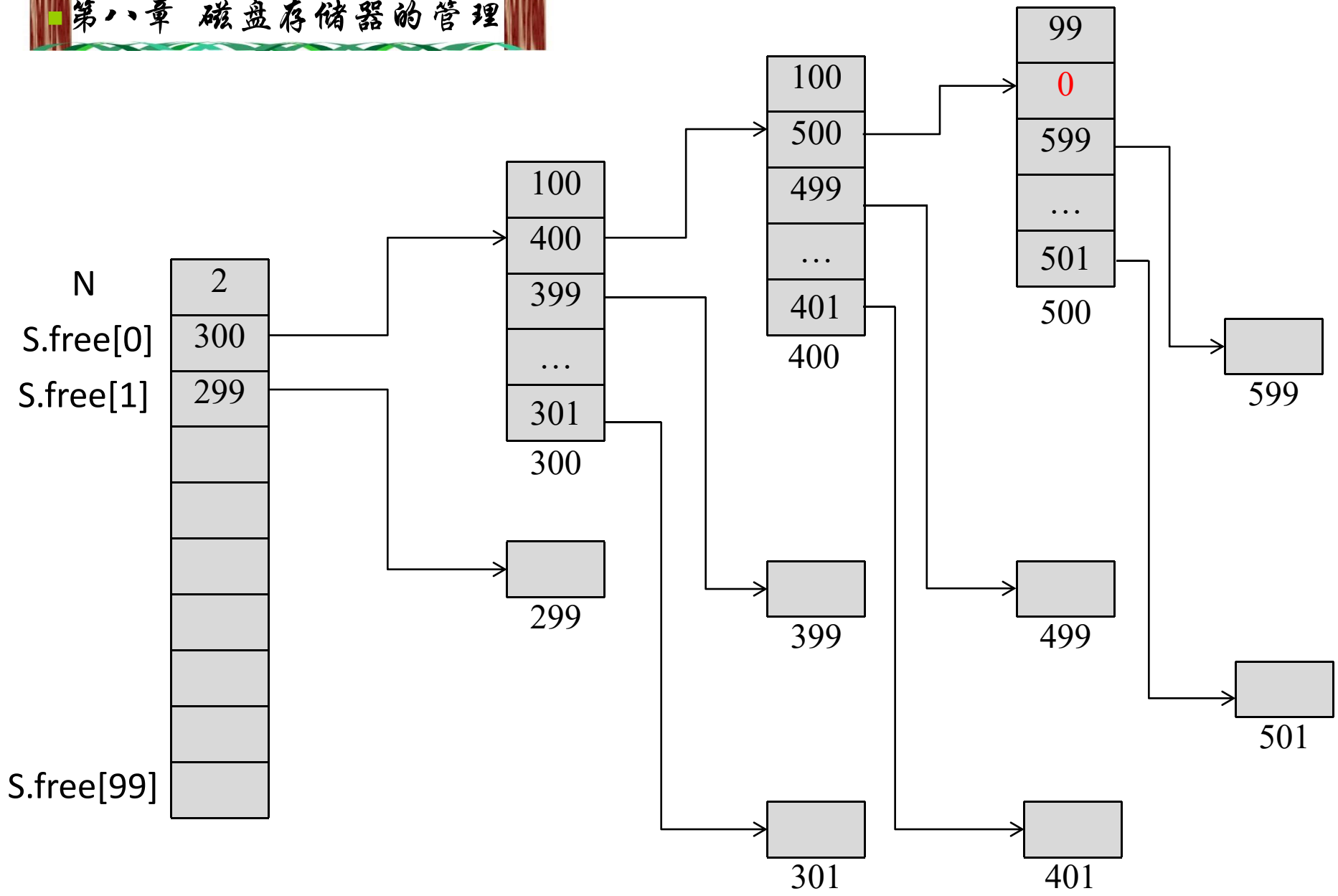
成组链接法经典例题

某个系统采用成组链接法来管理磁盘的空闲空间，目前磁盘的状态如图所示：

（1） 该磁盘中目前还有多少个空闲盘块？

（2） 请简述磁盘块的分配过程。

（3） 在为某个文件分配3个盘块后，系统要删除另一文件，并回收它所占的5个盘块，它们的盘块号依次为700、711、703、788、701，请画出回收后的盘块链接情况。


N
2
S.free[0]
300
S.free[1]
299
S.free[99]
100
400
399
…
301
300
299
100
500
499
…
401
400
399
301
99
0
599
…
501
500
599
501
499
401

解：

（1） 磁盘中目前还有301个空闲盘块。

首先看空闲盘块号栈，此时N=2，表示有两个空闲盘块299、300，而盘块300号上面又写着有100个空闲盘块：301-400，它还有下一个链接的盘块400；在盘块400中，记录有100个空闲盘块401-500；然后又链接到500号盘块，在500号盘块中，虽然N=99，但是第一个是0，它表示空闲盘块链的结尾。因此，总共的空闲盘块有：299、300、301-400、401-500、501-599；即301个空闲盘块。

解：

（2）分配三个空闲盘块，分配的过程是这样的：

- 首先看空闲盘块号栈，发现N=2，那么到达栈顶即S.free[2-1]=299，即把299号盘块分配出去了；

- 然后分配第二个盘块，这时N=1，如果再分配就会变成空栈了，因为S.free[N-1]=S.free[0]!=0,所以需要将300号盘块的内容拷贝到空闲盘块号栈，并分配300号盘块。

- 接着同理分配301号盘块。

解：

（3）回收700、711、703、788、701号盘块：

- 回收的过程也是从栈顶开始的，首先看N=99，然后回收700，会将700放在S.free[N]的位置，然后将N加1变成100；

- 然后回收711号盘块，因为此时空闲栈的N=100，已经满了，如果再回收，需要将空闲盘块栈的内容移动到711号盘块上，然后将空闲盘块栈的S.free[0]设置为711，N设置为1；

- 最后回收703/788/701。

8.3 提高磁盘I/O速度的途径

8.3.1 磁盘高速缓存(Disk Cache)

1. 磁盘高速缓存的形式

- 利用内存中的存储空间，来暂存从磁盘中读出的一系列盘块中的信息。
- 高速缓存逻辑上属于磁盘， 物理上是内存。
- 高速缓存在内存中可分成两种形式：
  - 第一种是在内存中开辟一个单独的存储空间来作为磁盘高速缓存，其大小是固定的，不会受应用程序多少的影响；
  - 第二种是把所有未利用的内存空间变为一个缓冲池，供请求分页系统和磁盘I/O时(作为磁盘高速缓存)共享。此时高速缓存的大小，显然不再是固定的。

2. 数据交付方式

系统可以采取两种方式， 将数据交付给请求进程：

(1) 数据交付。这是直接将高速缓存中的数据， 传送到请求者进程的内存工作区中。

(2) 指针交付。只将指向高速缓存中某区域的指针， 交付给请求者进程。

后一种方式由于所传送的数据量少，因而节省了数据从磁盘高速缓存存储空间到进程的内存工作区的时间。

3. 置换算法

由于请求分页系统与磁盘高速缓存的工作情况不同，因而在置换算法中所应考虑的问题也有所差异。高速缓存的**置换算法**除了考虑到最近最久未使用这一原则外， 还考虑了以下几点：

- （1） 访问频率。
- （2） 可预见性。
- （3） 数据的一致性。

4. 周期性地写回磁盘

- UNIX系统

专门增设了一个修改(update)程序，使之在后台运行，该程序周期性地调用一个系统调用SYNC。该调用的主要功能是强制性地将所有在高速缓存中已修改的盘块数据写回磁盘。一般是把两次调用SYNC的时间间隔定为30s。这样，因系统故障所造成的工作损失不会超过30s的劳动量。

- MS-DOS

只要高速缓存中的某盘块数据被修改，便立即将它写回磁盘，并将这种高速缓存称为“写穿透、高速缓存”(write-through cache)。 MS-DOS所采用的写回方式，几乎不会造成数据的丢失， 但须频繁地启动磁盘。

8.3.2 提高磁盘I/O速度的其它方法

1. 提前读(Read-Ahead)
2. 延迟写
3. 优化物理块的分布
4. 虚拟盘

8.3.3 廉价磁盘冗余阵列

1. 并行交叉存取

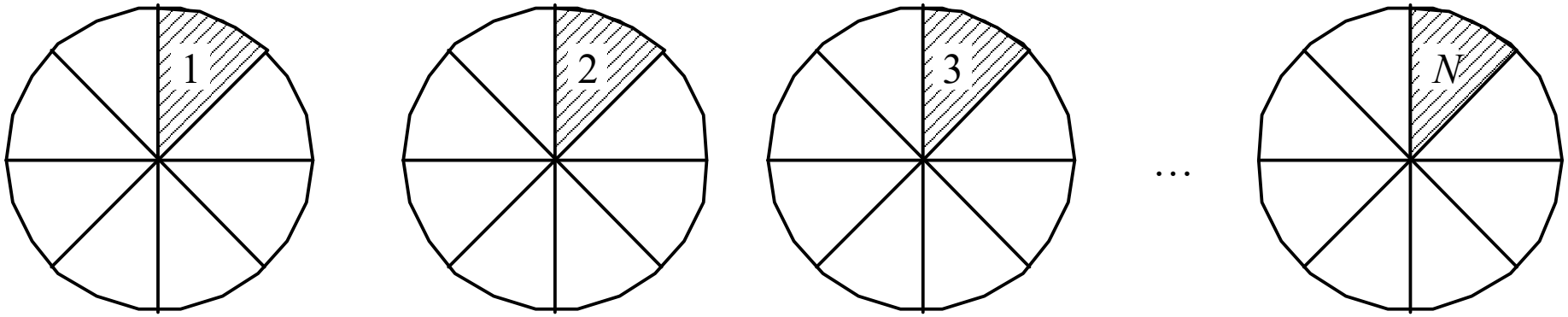


图8-12 磁盘并行交叉存取方式

2. RAID的分级

(1) RAID 0级。

(2) RAID 1级。

(3) RAID 3级。

(4) RAID 5级。

(5) RAID 6级和RAID 7级。

3. RAID的优点

(1) 可靠性高。

(2) 磁盘I/O速度高。

(3) 性能/价格比高。

8.4 提高磁盘可靠性的技术

1. 第一级容错技术SFT-Ⅰ

1) 双份目录和双份文件分配表

在磁盘上存放的文件目录和文件分配表FAT，是文件管理所用的重要数据结构。如果这些表格被破坏，等效于文件的丢失。因此在不同的磁盘上或在磁盘的不同区域中，建立(双份)目录表和FAT。

2) 热修复重定向和写后读校验：针对坏块管理。

- **热修复重定向**(Hot-Redirection)：在硬盘上为该坏块表分配一个扇区，当控制器第一次初始化时，他读坏块表并找一个空闲块代替坏块（即在热修复重定位区找），并在坏块中记录映射，此后，全部对该坏块的请求都采用该空闲块。
- **写后读校验**(Read after write Verification)方式：写入磁盘后，即读出进行比较，一致则写入成功。

2. 第二级容错技术SFT-II

(1) 磁盘镜像(Disk Mirroring)

针对磁盘驱动器故障

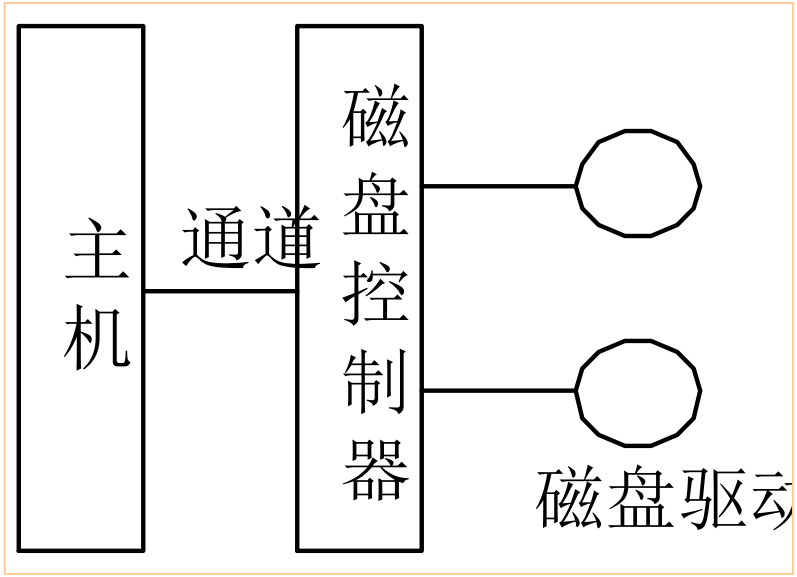


图 8-13 磁盘镜像示意

(2) 磁盘双工(Disk Duplexing)

针对磁盘控制器或者通道故障

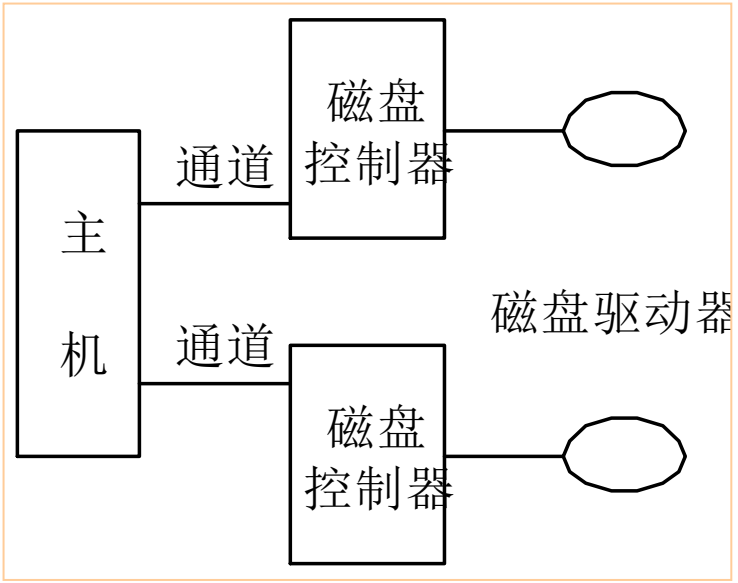


图 8-14 磁盘双工示意

8.5 数据一致性控制

8.5.1 事务

1. 事务的定义

事务是用于访问和修改各种数据项的一个程序单位。事务也可以被看作是一系列相关读和写操作。被访问的数据可以分散地存放在同一文件的不同记录中，也可放在多个文件中。只有对分布在不同位置的同一数据所进行的读和写(含修改)操作全部完成时，才能再以托付操作(Commit Operation)来终止事务。只要有一个读、写或修改操作失败，便须执行夭折操作(Abort Operation)。读或写操作的失败可能是由于逻辑错误，也可能是系统故障所导致的。

2. 事务记录(Transaction Record)

·事务名：用于标识该事务的惟一名字；

·数据项名：它是被修改数据项的惟一名字；

·旧值：修改前数据项的值；

·新值：修改后数据项将具有的值。

8.5.1 事务

3. 恢复算法

恢复算法可利用以下两个过程：

（1） undo〈T_i〉。该过程把所有被事务T_i修改过的数据，恢复为修改前的值。

（2） redo〈T_i〉。该过程能把所有被事务T_i修改过的数据，设置为新值。

如果系统发生故障， 系统应对以前所发生的事务进行清理。

8.5.2 检查点

1. 检查点(Check Points)的作用

引入检查点的主要目的，是使对事务记录表中事务记录的清理工作经常化， 即每隔一定时间便做一次下述工作： 首先是将驻留在易失性存储器(内存)中的当前事务记录表中的所有记录，输出到稳定存储器中；其次是将驻留在易失性存储器中的所有已修改数据，输出到稳定存储器中；然后是将事务记录表中的〈检查点〉记录，输出到稳定存储器中； 最后是每当出现一个〈检查点〉记录时，系统便执行上小节所介绍的恢复操作，利用redo和undo过程实现恢复功能。

8.5.2 检查点

2. 新的恢复算法

恢复例程首先查找事务记录表，确定在最近检查点以前开始执行的最后的事务T_i。在找到这样的事务后， 再返回去搜索事务记录表，便可找到第一个检查点记录，恢复例程便从该检查点开始，返回搜索各个事务的记录，并利用redo和undo过程对它们进行处理。

如果把所有在事务T_i以后开始执行的事务表示为事务集T， 则新的恢复操作要求是：对所有在T中的事务T_K，如果在事务记录表中出现了〈T_K托付〉记录，则执行redo〈T_K〉操作； 反之，如果在事务记录表中并未出现〈T_K托付〉记录，则执行undo〈T_K〉操作。

8.5.3 并发控制

1. 利用互斥锁实现“顺序性”

2. 利用互斥锁和共享锁实现顺序性

8.5.4 重复数据的数据一致性问题

1. 重复文件的一致性

文件名	i 结点
文件 1	17
文件 2	22
文件 3	12
文件 4	84

(a)

文件名	i 结点		
文件 1	17	19	40
文件 2	22	72	91
文件 3	12	30	29
文件 4	84	15	66

(b)

图 6-28 UNIX类型的目录

8.5.4 重复数据的数据一致性问题

2. 盘块号一致性的检查

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	0 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	1 0	0 0	0 1	1 0	0 0	1 1

(a) 正常情况

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	0 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	0 0	1 0	0 0	0 1	1 0	0 0	1 1

(b) 丢失了盘块

图 6-29 检查盘块号一致性情况

8.5.4 重复数据的数据一致性问题

盘块号 / 计数器组	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	2 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	0 0	0 0	0 1	1 0	0 0	1 0

(c) 空闲盘块号重复出现

盘块号 / 计数器组	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	1 0	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	0 2	0 0	0 1	1 0	0 0	1 1

(d) 数据盘块号重复出现

图 6-29 检查盘块号一致性情况

8.5.4 重复数据的数据一致性问题

3. 链接数一致性检查

为每个共享文件建立一个表项，其中含有该索引结点号的计数值。 在进行检查时，从根目录开始查找，每当在目录中遇到该索引结点号时， 便在该计数器表中相应文件的表项上加1。当把所有目录都检查完后，便可将该计数器表中每个表项中的索引结点号计数值与该文件索引结点中的链接计数count值加以比较，如果两者一致，表示是正确的；否则，便是发生了链接数据不一致的错误。

8.5.4 重复数据的数据一致性问题

如果索引结点中的链接计数count值大于计数器表中相应索引结点号的计数值，则即使在所有共享此文件的用户都不再使用此文件时，其count值仍不为0，因而该文件不会被删除。这种错误的后果是使一些已无用户需要的文件仍驻留在磁盘上，浪费了存储空间。解决的方法是用计数器表中的正确的计数值去为count重新赋值。

反之，如果出现count值小于计数器表中索引结点号计数值的情况时，就有潜在的危险。假如有两个用户共享一个文件，但是count值仍为1， 这样， 只要其中有一个用户不再需要此文件时， count值就会减为0，从而使系统将此文件删除， 并释放其索引结点及文件所占用的盘块，导致另一共享此文件的用户所对应的目录项，指向了一个空索引结点，最终是使该用户再无法访问此文件。如果该索引结点很快又被分配给其它文件，则又会带来潜在的危险。 解决的方法是将count值置为正确值。

小结：文件系统的层次模型

- 第一层用户接口，根据用户对文件的存取要求，把不同的系统调用换成内部调用格式；
- 第二层符号文件系统，将上层的用户文件名换成系统内部名；
- 第三层基本文件系统，根据文件内部名找到文件的说明信息：存取控制表、文件结构及物理地址等。
- 第四层存取控制验证，根据存取控制信息和用户访问要求验证文件访问的合法性；
- 第五层逻辑文件系统，根据文件逻辑结构找到所要操作的数据的相对块号；
- 第六层物理文件系统，根据文件物理结构将相对块号转换物理地址；
- 第七层文件存储设备分配策略和设备策略模块，前者实现对空闲块的管理；后者将物理块号转换成相应物理设备所要求的地址格式，准备I/O操作；
- 第八层I/O调度及控制，执行具体的读写操作。

小结：文件系统的层次模型