第七章 文件系统

7.1 文件和文件系统
7.2 文件结构
7.3 文件存储空间管理
7.4 目录管理
7.5 文件的共享
7.6 文件的保护
7.7 磁盘容错技术
7.8 文件系统性能的改善
7.9 数据一致性控制

7.1 文件和文件系统

7.1.1 引言

7.1.2 文件

7.1.3 文件系统

7.1.1 引言

在早期计算机系统中，人们直接用物理地址存放信息。存放信息时，要求用户指出并记住信息存放在哪个设备的哪些磁道、哪些扇区上。

在多用户的环境中这几乎是不可能的，更是不能忍受的。

实际上对用户来说，关心的不是信息的具体存放位置，而是存取方法的方便、可靠。不是信息的物理结构而是信息的逻辑结构。

因此，引入文件和文件系统的概念，它是操作系统的重要组成部分。

7.1.2 文件

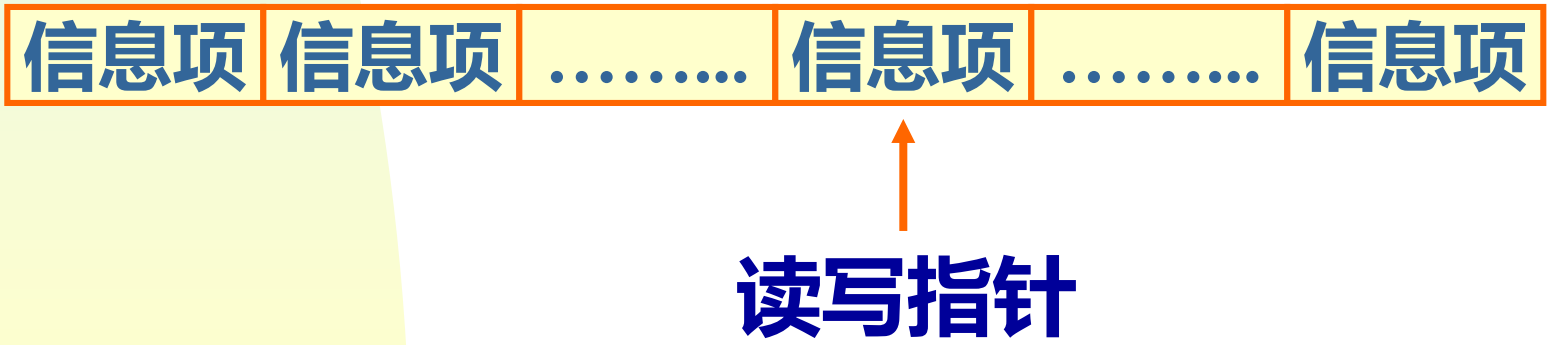
1、文件的定义

文件是计算机系统中信息存放的一种组织形式，目前尚无严格的定义，下面给出两种有代表性的解释：

文件是赋名的信息（数据）项的集合。

文件是赋名有关联的信息单位（记录）的集合。

编号：0　　1　　……　　i　　……　　n-1



这两种解释定义了两种文件形式：

（1）流式文件

文件是由字节组成，这是一种无结构的文件，或称流式文件。

（2）记录式文件

文件是由记录组成。而记录则是由一组相关信息项组成。

例如：学生登记表可视为一个记录，它包括学生姓名、出生年月、性别、籍贯等信息项。

2、文件的分类（P223）

（1）以文件的用途分类

系统文件：操作系统的执行程序和数据组成的文件，不对用户开放，仅供系统使用。

库文件：系统为用户提供的各种标准函数，标准过程和实用程序等。

用户只能使用这些文件，而无权对其进行修改。

用户文件: 由用户的信息组成的文件，如源程序文件，数据文件等。可使用和修改。

（2）从按文件的操作保护分类

只读文件：只允许进行读操作。

读写文件：允许进行读写操作。

只执行文件：只能执行，不能读写。

（3）按文件的性质分类

普通文件：指一般的用户文件和系统文件。

目录文件：指由文件目录项组成的文件。

特别文件：有的系统把设备作为文件统一管理和使用，并为区别起见，把设备称为特别文件。

UNIX操作系统把文件分成普通文件、目录文件和特别文件。

3、文件属性

文件属性：文件类型、文件长度、文件物理位置、文件创建时间、存取保护等。

文件的属性一般存放在文件的目录项中。

例如MS-DOS系统中，文件属性占目录项的一个字节，在这个字节中，01表示文件仅读，02表示隐含文件等。

7.1.3 文件系统

操作系统中负责管理文件的机构称为文件系统。也有的文献上叫信息系统。

文件系统负责文件的创立、撤消、读写、修改、复制和存取控制等，并管理存放文件的各种资源。

7.2 文件结构

7.2.1 概述

7.2.2 文件的逻辑结构

7.2.3 文件的物理结构

7.2.1 概述

研究文件结构有两种观点：

用户的观点（文件的**逻辑结构**）：主要研究用户思维中的抽象文件，为用户提供一种逻辑结构清晰、使用简便的逻辑文件。

用户将按这种形式去存取、检索和加工文件。

例如用户可将文件看作字节的集合。或者用户将文件看作记录的集合。

实现的观点（文件的**物理结构**）：主要研究驻留在存储介质上的文件的结构。

文件的物理结构：文件的各个字节在存储介质上是如何摆放的。

7.2.2 文件的逻辑结构

1、文件的逻辑结构

流式文件：基本信息单位是字节或字，其长度是所含字节的数量。

这种文件的优点是节省存储空间。

在这种文件中无需额外的说明和控制信息。

记录式文件：记录式文件是一种结构文件。由若干个记录组成，文件中的记录可按顺序编号为记录1，记录2，……，记录n。

定长记录文件：记录的长度相等

变长记录文件：记录长度不相等

相对流式文件而言，记录式文件的使用不很方便，尤其是变长记录文件。另外在文件中还要有说明记录长度的信息，这就浪费了一部分存储空间。

因此许多现代计算机操作系统如UNIX操作系统等都取消了记录式文件。

按文件组织方式分类

顺序文件

记录按某种顺序（例如：关键字、时间顺序）排列。记录包括定长和变长

顺序文件记录寻址：变长记录需要按顺序查找或建立索引

按文件组织方式分类

索引文件

一个文件对应一个索引表，每条记录占一项，内容包括：关键词、逻辑起始地址、长度等

索引表是顺序文件

按文件组织方式分类

索引顺序文件

索引表中每一项对应一组记录。组间索引查找，组内顺序查找

一级索引举例：10000条记录，顺序5000，索引100*100：50+50=100

二级索引举例：1000 000条记录，顺序50万，索引100*100*100:50+50+50=150

2、文件的存取方法

顺序存取

文件存取最简单的方法是顺序存取，即严格按文件信息单位排列的顺序依次存取。

当打开文件时，文件的存取指针指向第一个信息单位，存取完成后指针加1

随机存取

也称**直接存取**，每次存取操作时必须先确定存取的位置。

流式文件或定长记录的文件：容易确定

不定长的记录式文件：从第一个记录开始顺序查询

解决方法：建立索引

7.2.3 文件的物理结构

文件的物理结构是指文件在物理存储介质上的结构。

1、连续结构

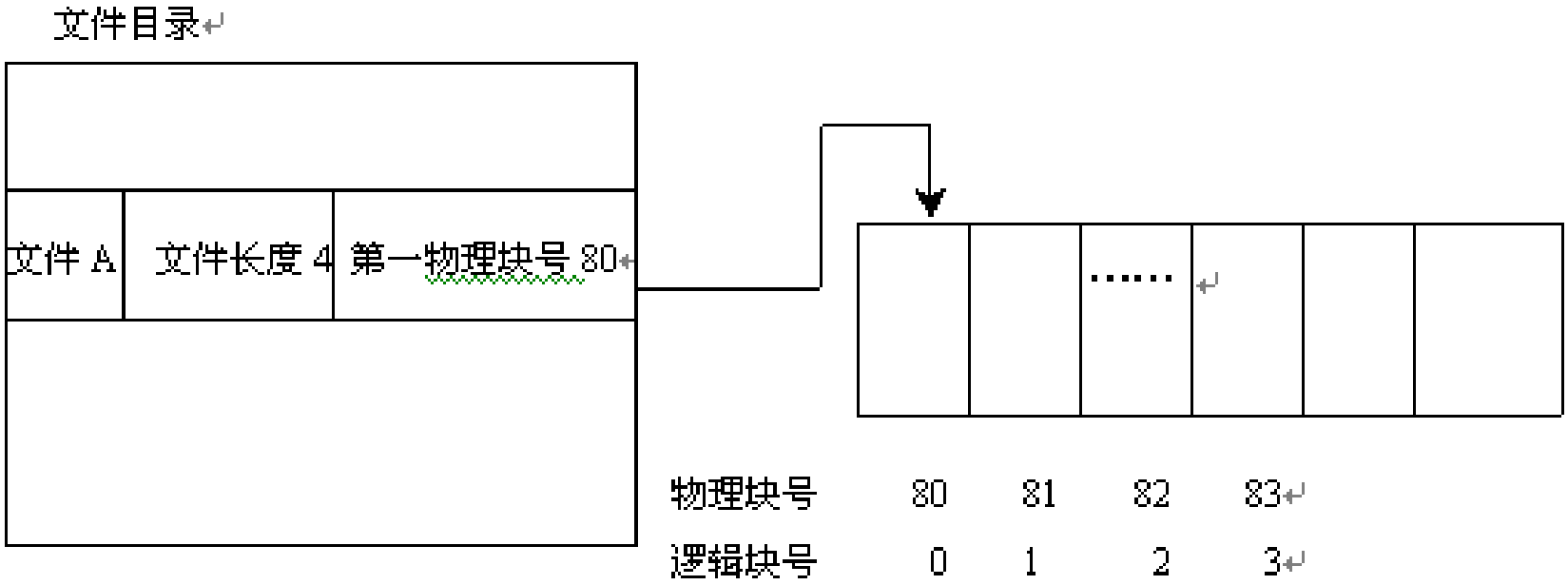
2、链接结构

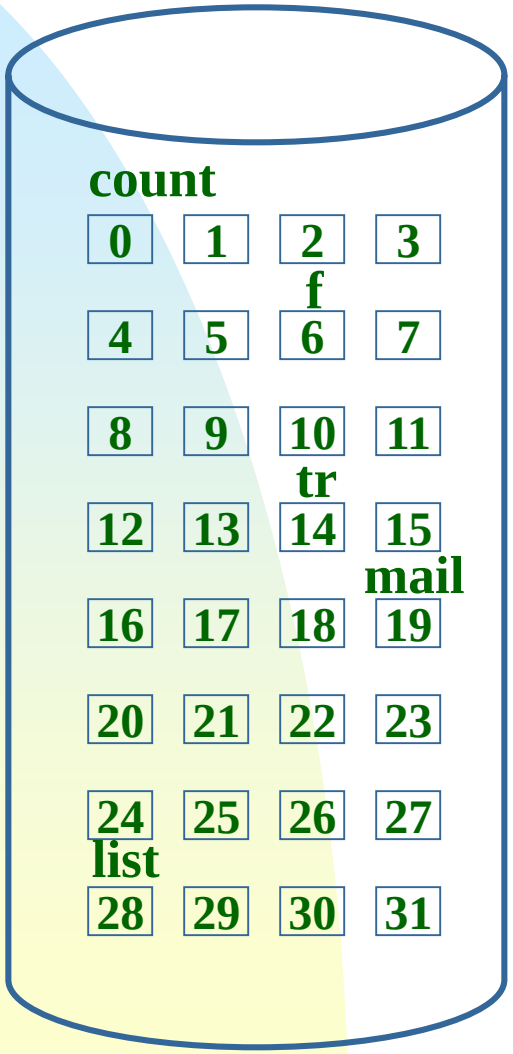
3、索引结构

1、连续结构

文件的全部信息存放在外存的**一片连续编号**的物理块中

要求：要求给出文件的最大长度，以便分配空间


文件目录
文件 A
文件长度 4
第一物理块号 80
.....
物理块号 80 81 82 83
逻辑块号 0 1 2 3



文件目录

文件名	始址	块数
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

优点

简单

支持顺序存取和随机存取

顺序存取速度快

所需的磁盘寻道次数和寻道时间最少

缺点

文件不易动态增长

- 预留空间:浪费
- 重新分配和移动

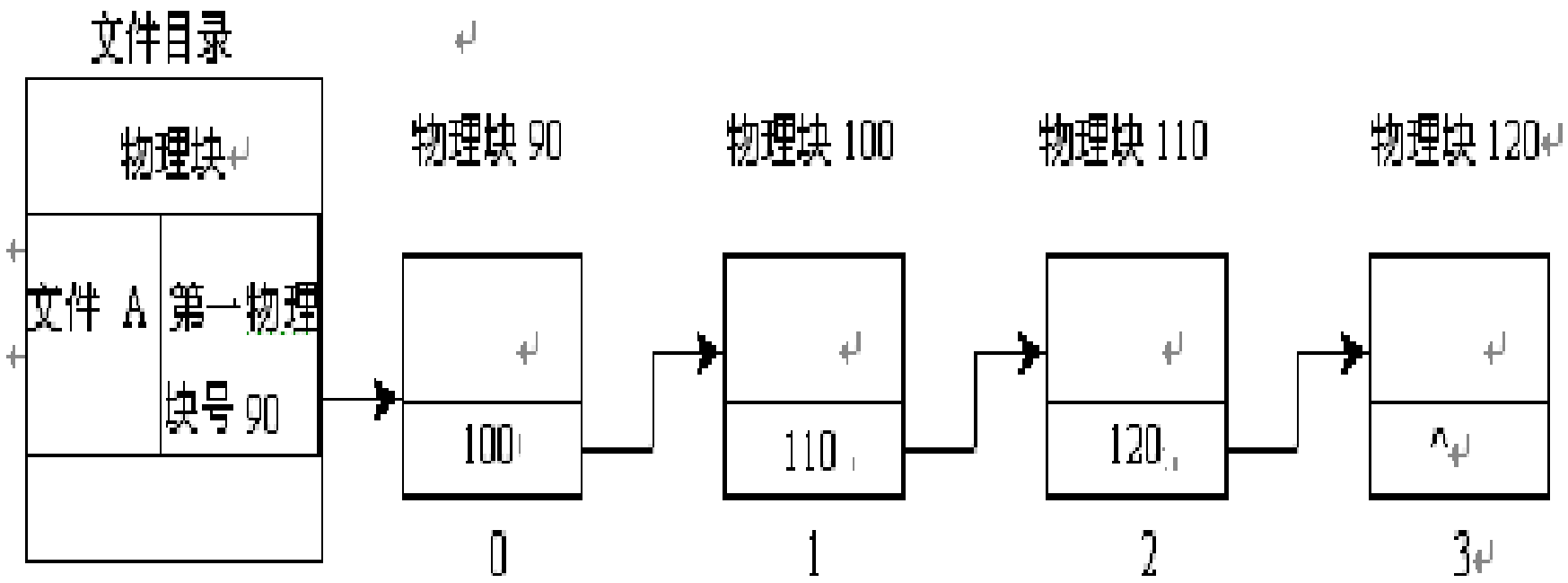
不利于文件插入和删除

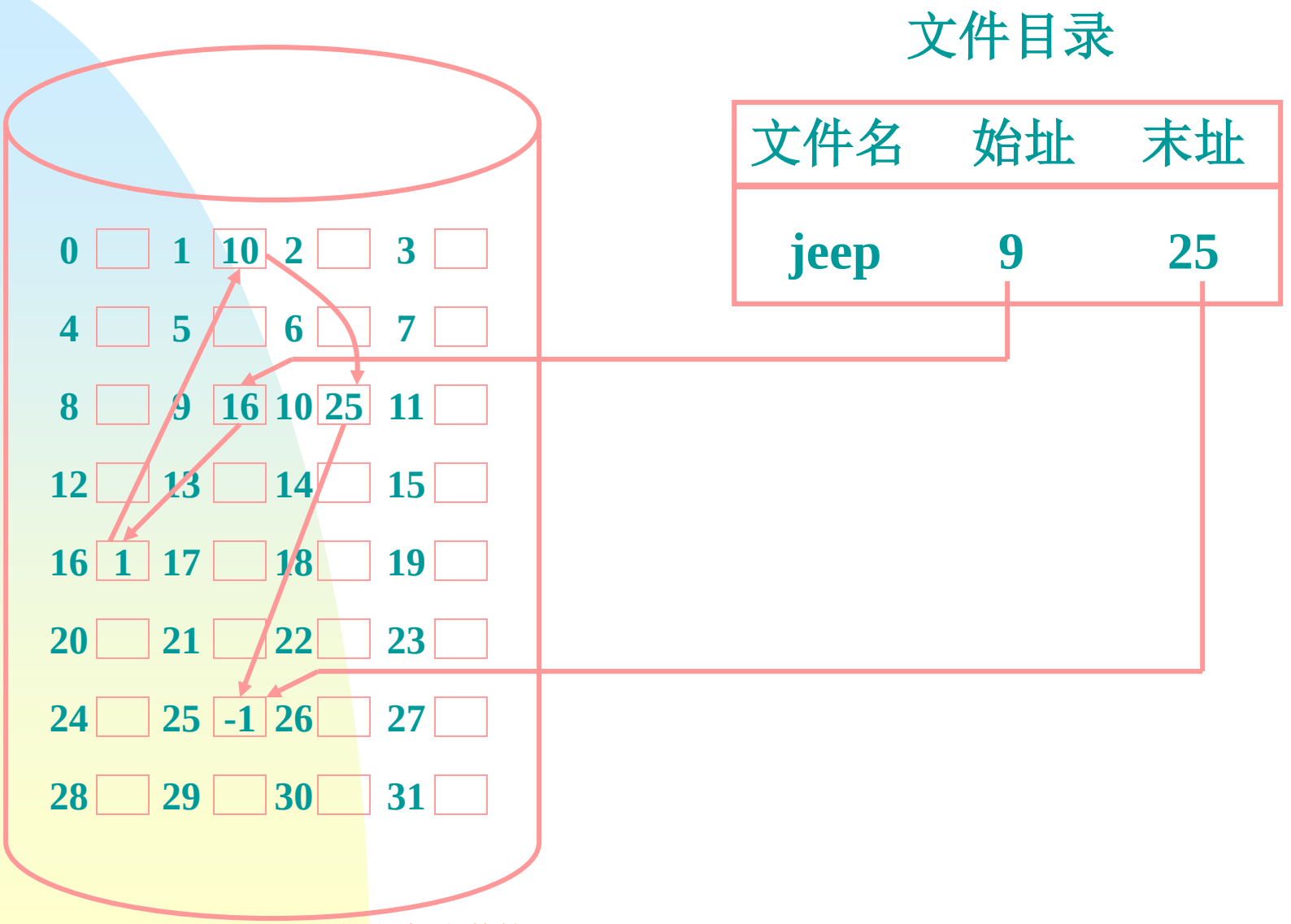
外部碎片问题

- 存储压缩技术

2、链接结构

非连续的结构，存放文件信息的**每一物理块中有一个指针**，指向下一个物理块


文件目录
物理块
文件 A
第一物理块号 90
物理块 90
物理块 100
物理块 110
物理块 120
100
110
120
^
0
1
2
3


文件目录
文件名 始址 末址
jeep 9 25
0 1 10 2 3
4 5 6 7
8 9 16 10 25 11
12 13 14 15
16 1 17 18 19
20 21 22 23
24 25 -1 26 27
28 29 30 31

链接结构的文件适用于顺序存取。因为要获得某一块的块号，必须读取上一物理块，因此要随机地存取信息就较为困难。

优缺点

优点：

提高了磁盘空间利用率,不存在外部碎片问题

有利于文件插入和删除

有利于文件动态扩充

缺点：

存取速度慢，不适于随机存取

链接指针占用一定的空间

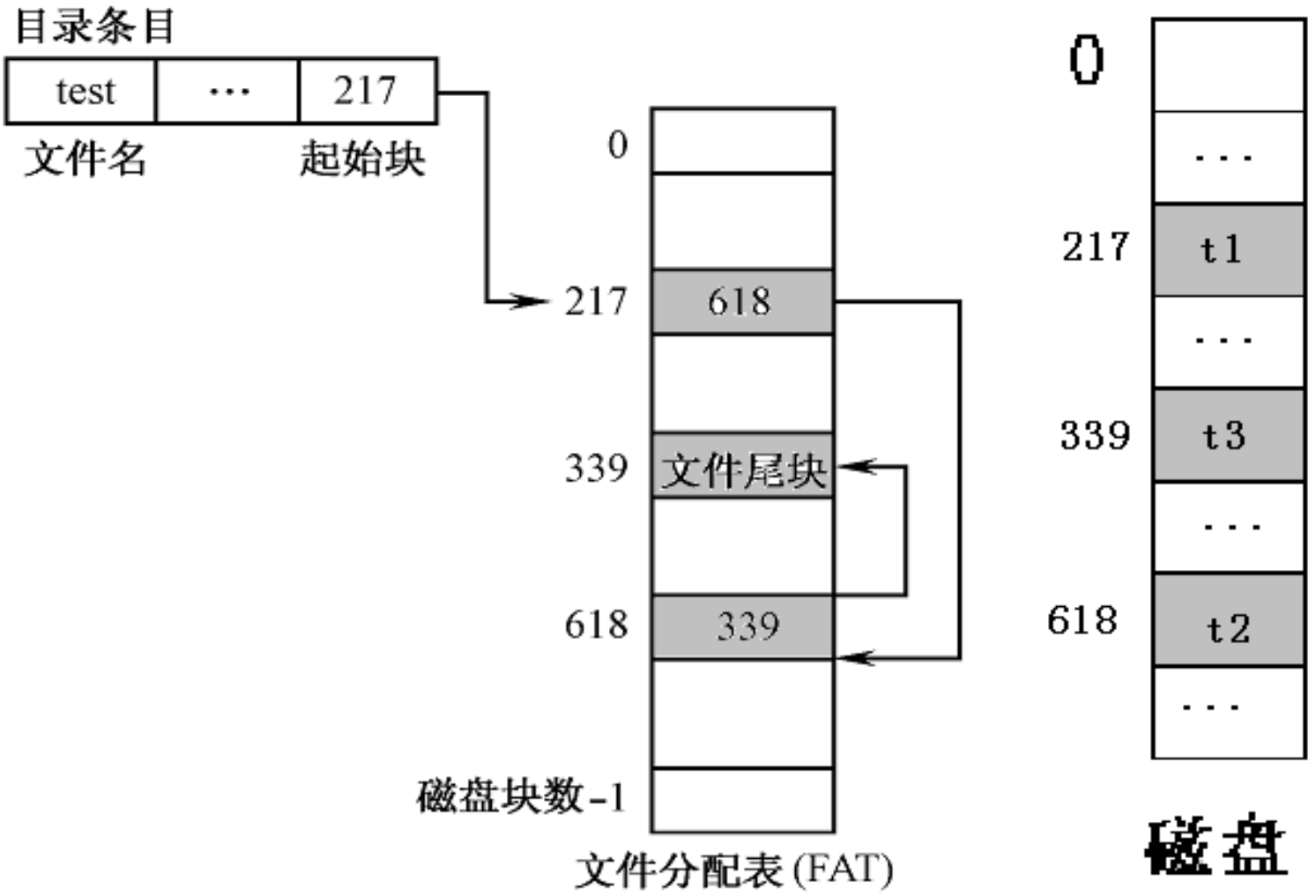
可靠性问题，如指针出错

链接结构的变形

文件分配表(FAT)

将盘块中的链接字按盘块号的顺序集中起来，构成盘文件映射表/文件分配表 。利用FAT可方便地进行随机存取。

图示


目录条目
test
…
217
文件名
起始块
0
217
618
339
文件尾块
618
339
磁盘块数-1
文件分配表 (FAT)
0
217
t1
339
t3
618
t2
磁盘

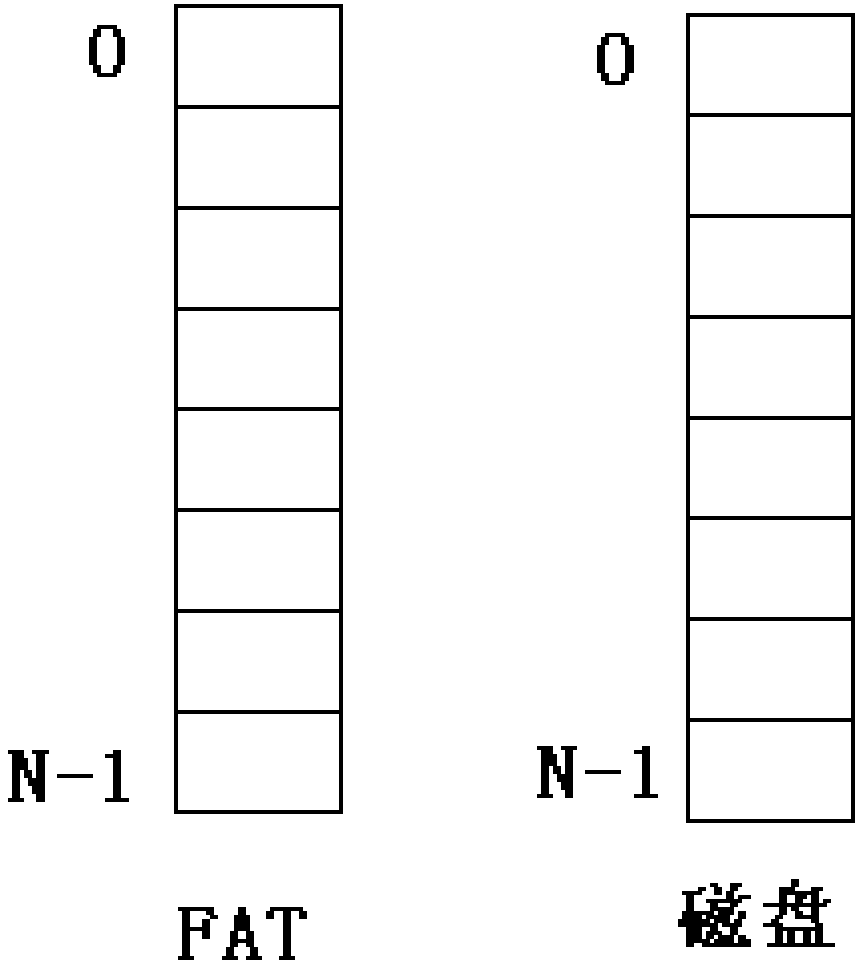
FAT也要占用一定的存储空间，若盘的容量较大，也可能占用较多的存储空间。在进行文件访问时，可能在内存中装不下整个FAT，这样就会造成若要读某块文件信息时，还要读盘块映射表的操作，影响使用效率。

FAT的实例

在MS-DOS和Windows系统中，文件的物理结构使用的是FAT结构。

将磁盘空间划分为块，每块大小为扇区的整数倍。在FAT文件系统中块称为簇

一个磁盘分区能分为多少簇则FAT就有多少表项


0
N-1
FAT
0
N-1
磁盘

思考

什么叫FAT16、FAT32？

在FAT16中一簇最大64个扇区，为什么FAT16能管理的磁盘分区为2G？

FAT32同FAT16相比有什么优点？

思考

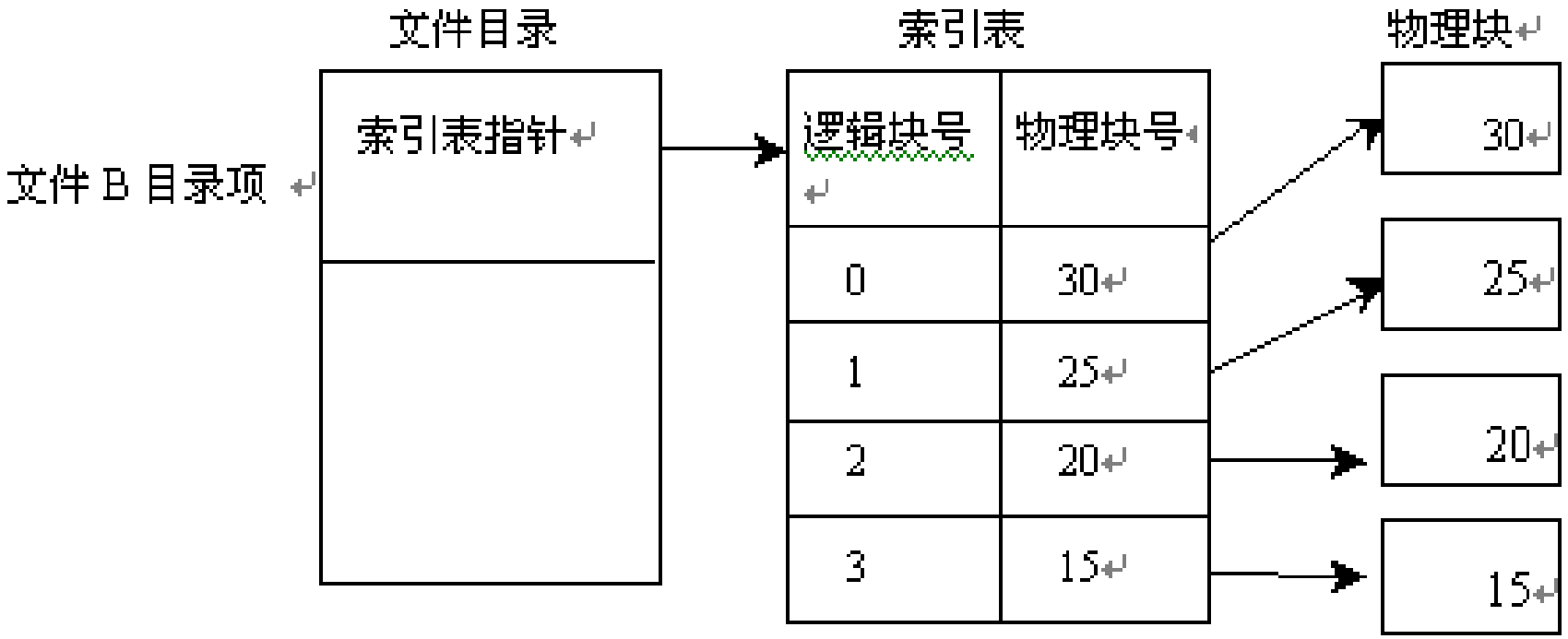
对于FAT16文件系统，若一个磁盘分区的大小为512M，问一个簇最少要为多少个扇区？

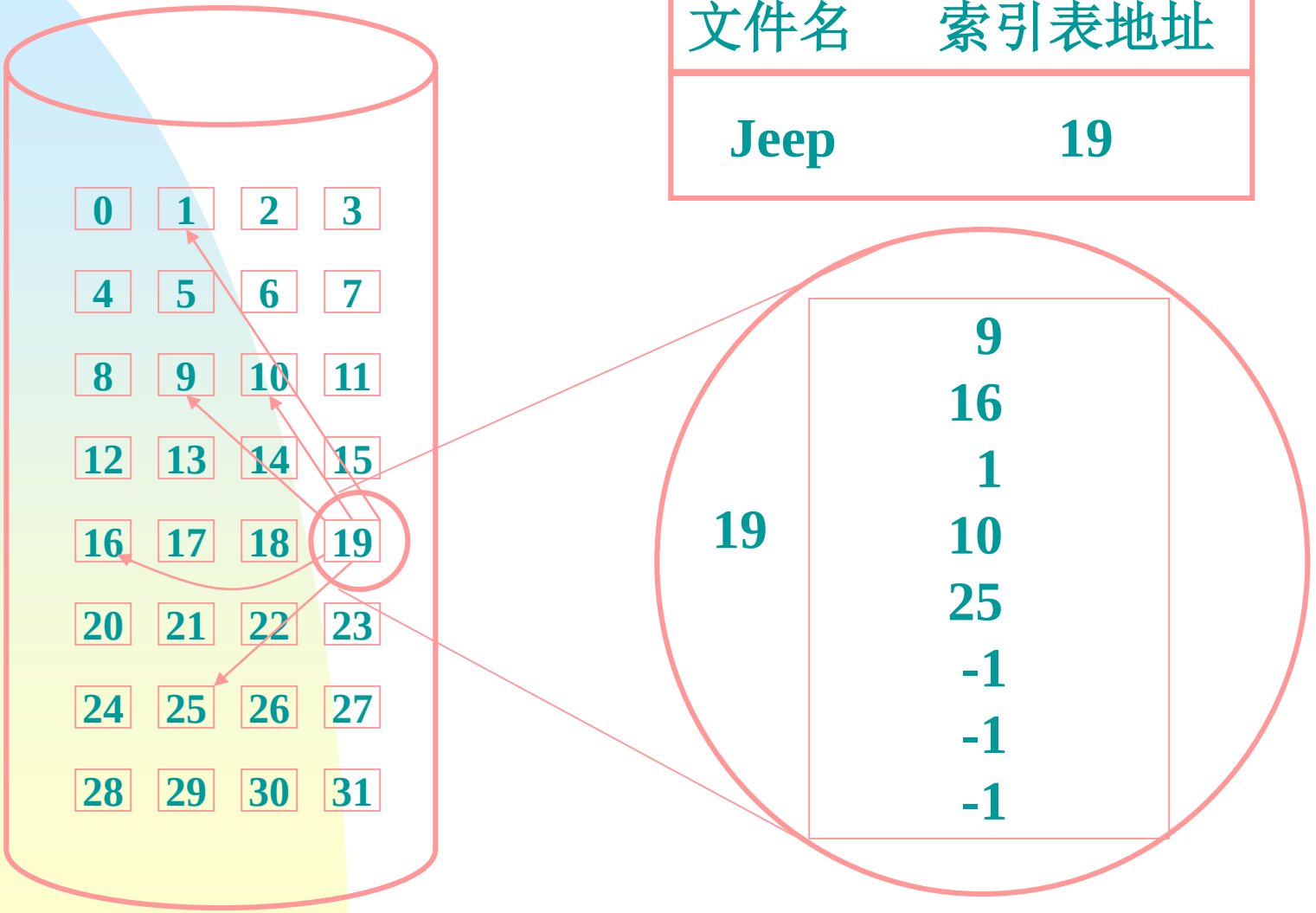
簇是大点好，还是小点好？

3、索引结构

一个文件的信息存放在若干不连续物理块中，系统为每个文件建立一个专用数据结构--索引表，并将这些块的块号存放在索引表中。

一个索引表就是磁盘块地址数组,其中第i个条目指向文件的第i块


文件目录
索引表
物理块
文件 B 目录项
索引表指针
逻辑块号
物理块号
0
30
1
25
2
20
3
15
30
25
20
15


文件目录
文件名
索引表地址
Jeep
19
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
19
9
16
1
10
25
-1
-1
-1

索引组织方式

（1）单级索引

（2）多级索引

（3）增量式索引：FCB中包含直接寻址、一次间址、二次间址、三次间址

优点

保持了链接结构的优点，又解决了其缺点：
即能顺序存取，又能随机存取
满足了文件动态增长、插入删除的要求
能充分利用外存空间

缺点

索引表本身带来了系统开销

如：内外存空间，存取时间

7.3 外存空间管理

外存空间管理主要就是空闲块的管理，有以下方法：

7.3.1 空闲表法

7.3.2 空闲链表法

7.3.3 位图法

7.3.4 成组链接法

7.3.1 空闲表法

与内存管理中的动态分区分配方式相同。

起始空闲盘块号	盘块数
2	4
9	3
15	5
。 。 。	。 。 。

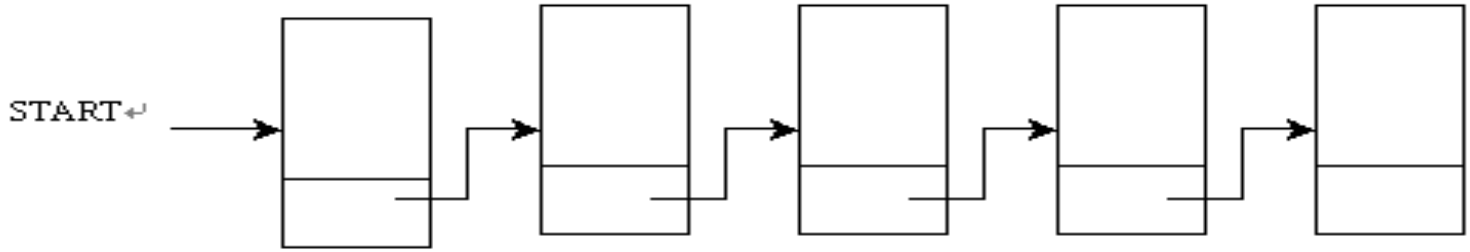
空闲盘块的分配与内存的动态分配类似，同样可以用首次、最佳、最坏适应法。盘块的回收也同内存的回收方式类似。

7.3.2 空闲链表法

空闲块链是另一种空闲块的组织方法，它用一种链结构把所有空闲块链接在一起。

分配：当系统建立文件需分配空闲块时，从链中摘取所需的空闲块，然后调整链首指针。

回收：反之，当回收空闲块时;把释放的空闲块逐个插入链首 。



这种方法只需在系统中保留一个链首指针，令其指向第一个空闲块。

这种方法的优点是简单，但工作效率较低，因为每次在链上增加和移出空闲块时，需要做I/O操作。

例如把一空闲块插入链时，要把链首指针(原指向第一个空闲块)写该空闲块中，然后让链首指针指向该空闲块。从链中摘取空闲块时也要读取下一个空闲块的指针。

7.3.3 位图法

系统为磁盘建立一张位图，在位图中每个物理块占1位，按物理块的顺序排列。"1"表示对应的物理块已占用，"0"表示空闲。

1	1	1	1	0	1	1	0	0	0	0	1	0	0	0	1
1	1	1	1	0	1	1	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	0	1	0	1	0	1	1	0	0	0	0
1	1	1	1	0	1	1	0	0	0	0	1	0	0	0	1
1	1	1	1	0	1	1	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	0	1	0	1	0	1	1	0	0	0	0
0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1

分配时首先在位图中找到为″0″的位，然后转换成对应的物理块号，分配给申请者，并把相应的位置为″1″。

回收时先将释放的物理块号转换成相应的位，并把这一位置为″0″。

位图的大小依据物理磁盘的容量而定。如360KB的软盘，每个物理块为512字节，位图只占用90个字节

7.3.4 成组链接法

UNIX采用此法

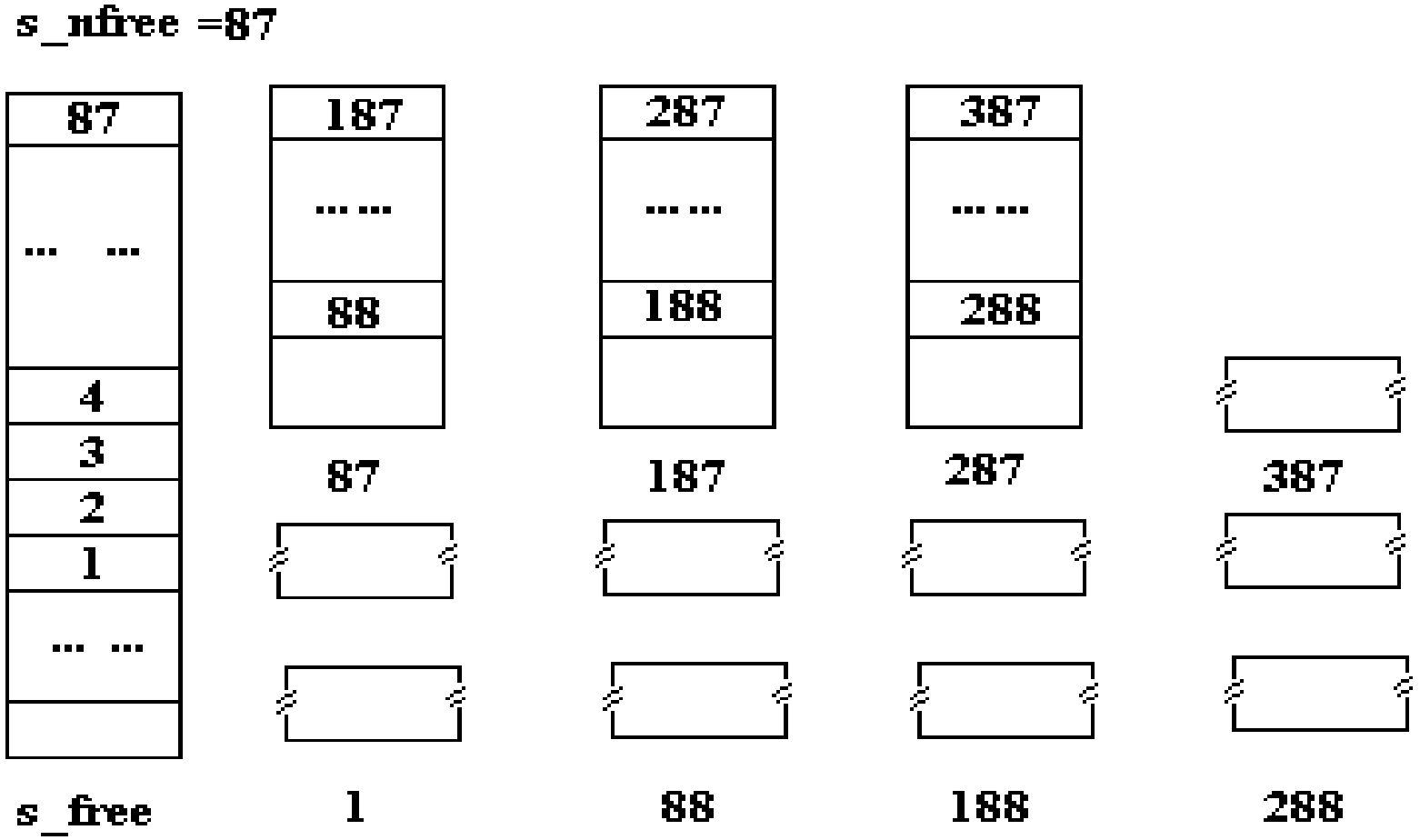
原理

在UNIX中中有一个整型数组s_free[100] 和一个整型变量s_nfree。

将所有的空闲盘块分组，每100个空闲盘块为一组。最后一组的块号填入s_free[ ]、块数赋于s_nfree。其余各组的块号则分别存放在它的下一组的第一个盘块中。

图解

假设共有387个空闲块



分配

分配空闲盘块时，总是分配s_free[s_nfree]所指的盘块，并且s_nfree减1 。当发现是直接管理的最后一个盘块时，即s_nfree=1时，就将该盘块中的索引表写入到s_nfree和s_free[]中，使得下一组变为直接管理。如此类推直到最后一组。

释放

释放空闲盘块时，将其块号登记在s_free[]表中第一个未被占用的项。例如，若s_nfree的原先值为87，则将释放块号登记在s_free[88]中，然后s_nfree加1。若在登记之前发现s_free已满，则将s_free[1]至s_free[100]的内容复写到要释放的盘块中。这样原来直接管理的100空闲盘块变为由释放块间接管理。然后将此该释放块的块号填入s_free[1]，把s_nfree置为1。

7.4 目录管理

7.4.1 基本概念

文件控制块（FCB）：文件控制块是操作系统为管理文件而设置的数据结构，存放了为管理文件所需的所有有关信息

文件控制块是文件存在的标志

FCB就是目录表中的一个目录项

文件控制块的内容：

文件名，文件号，用户名，文件地址，文件长度，文件类型，文件属性，共享计数，文件的建立日期，保存期限，最后修改日期，最后访问日期，口令，文件逻辑结构，文件物理结构等。

文件目录：把所有的FCB组织在一起，就构成了文件目录，即文件控制块的有序集合

目录项：构成文件目录的项目（目录项就是FCB）

目录文件：为了实现对文件目录的管理，通常将文件目录以文件的形式保存在外存，这个文件就叫目录文件

7.4.2 目录结构

1、一级目录结构

为所有文件建立一个目录文件（组成一线性表）

文件名	文件的物理位置	日期	时间	其他信息
C				
bsc				
Wps				
……				

优缺点

优点：简单，易实现

缺点：

限制了用户对文件的命名(**重名问题**)

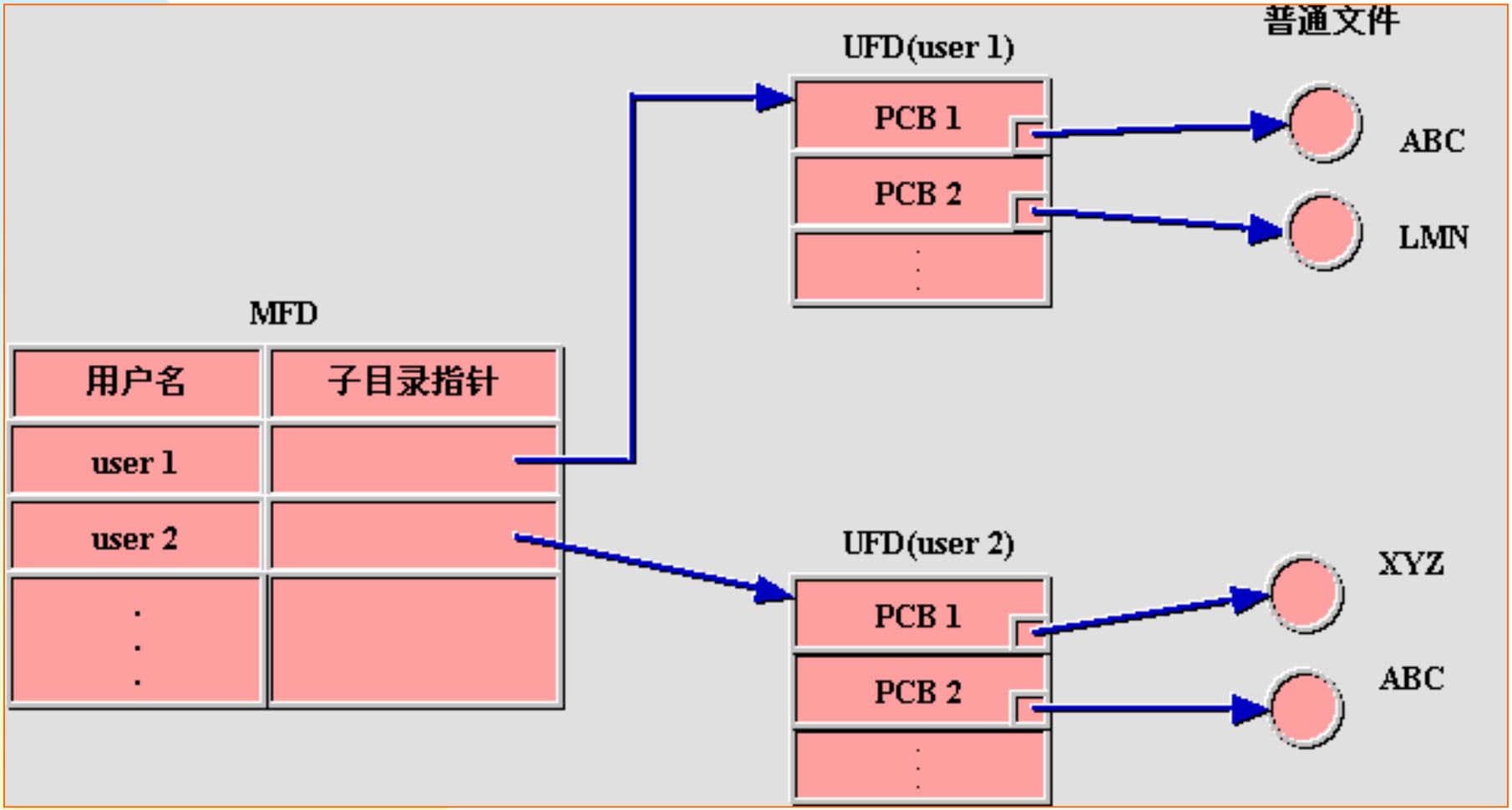
文件平均检索时间长

限制了对文件的共享

2、二级目录结构

二级文件目录结构把目录分成主目录和用户文件目录两级。

主目录由用户名和用户文件目录首地址组成，用户文件目录中登记相应的用户文件的目录项。


普通文件
UFD(user 1)
PCB 1
PCB 2
ABC
LMN
MFD
用户名
子目录指针
user 1
user 2
UFD(user 2)
PCB 1
PCB 2
XYZ
ABC

在二级目录结构中，区别不同的文件除文件名外还有文件的用户名，因此不同的用户可以使用相同的文件名。

例如用户A中使用文件名LISH，用户B也可使用文件名LISH，因为标识这两个文件时还要加上用户名，A：LISH和B：LISH，不致于造成混淆。

优缺点

优点：二级目录结构较为简单，也比较好地解决了重名的问题。

缺点：缺乏灵活性，特别是不能反映现实世界中多层次的关系。

为此人们提出了多级目录结构，其中MULTICS及UNIX系统均采用了多级目录结构，它们是当前文件系统的典型而完美的代表。

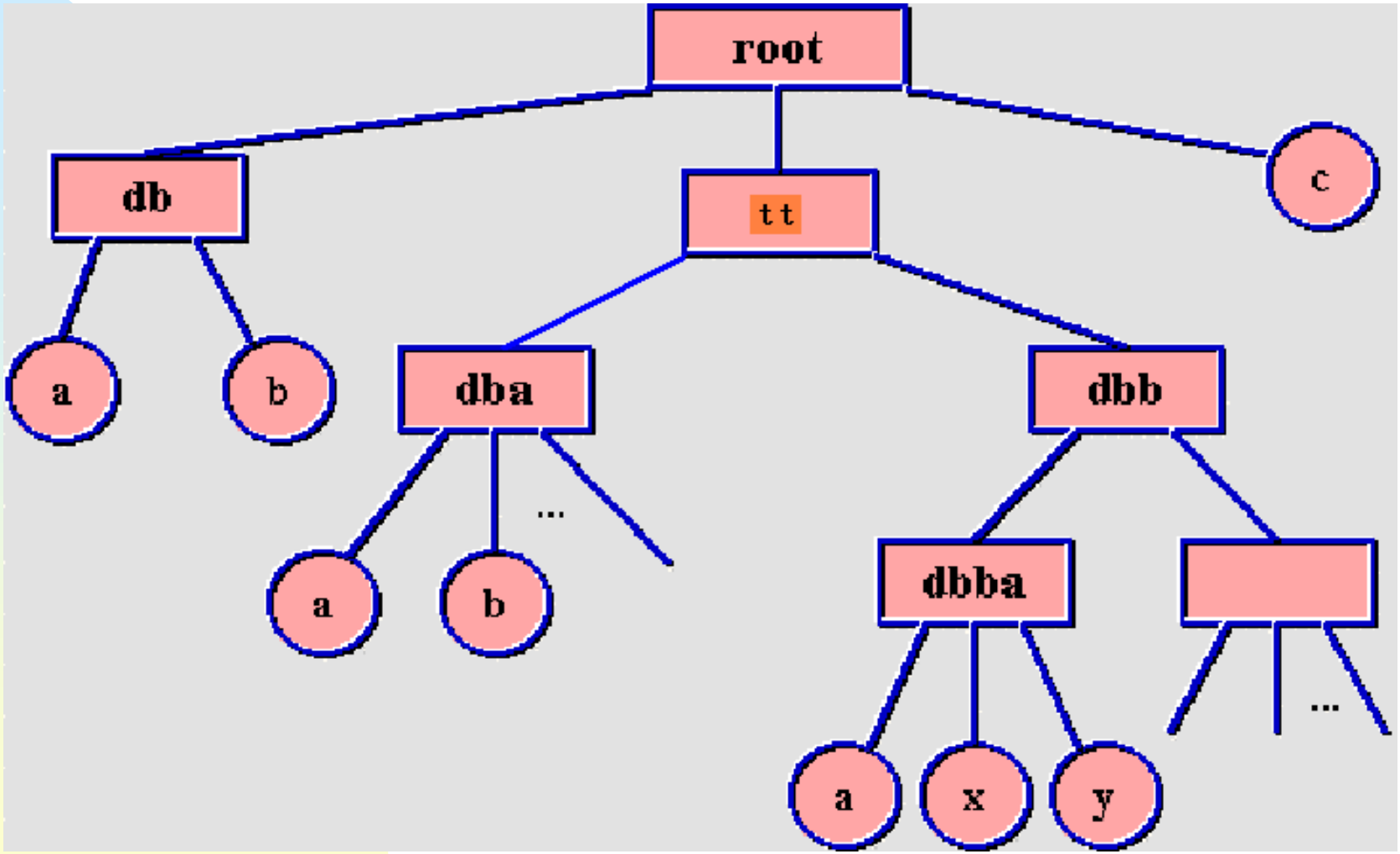
3、多级目录结构

多级目录结构由根目录和各级目录组成，为管理上的方便，除根目录外，其它各级目录均以文件的形式组成目录文件。

根目录中的每个目录项可以对应一个目录文件，也可以对应一个数据文件，同样目录文件中的每个目录项可以对应一个目录文件。也可以对应一个数据文件。如此类推，就形成多级目录结构。

也称树形目录结构

在这种结构中把根目录称为根结点，把各级目录文件称中间结点，用方框表示。数据文件称为叶结点，用圆圈表示。


root
db
tt
c
a
b
dba
dbb
...
a
b
dbba
...
a
x
y

路径名

在多级目录结构中一个文件的唯一标识不再是文件名，而是从根结点开始，经过一个或多个中间结点，到达某个叶结点的一条路径。称这条路径为文件的路径名，它是文件的唯一标识。

路径名由根目录和所经过的目录名和文件名以及分隔符组成，通常使用分隔符 /。例如 /d1/f1，　/d2/d5/f3，　/f7

工作目录

在多级目录结构中，文件路径名一般较长，而用户总是局部地使用文件，为了方便起见，可把经常使用的文件所在的目录指定为工作目录(或称当前目录)。

查询时，若路径名以/开头；则从根目录开始查找，否则从当前目录开始查找。

目录查询技术

在文件目录中查找目录项

（1）线性检索法

按顺序查找

（2）Hash方法

将文件名利用Hash函数得到一个索引值，然后在索引表中查找

4、文件目录改进

为加快目录检索可采用目录项分解法：把FCB分成两部分：

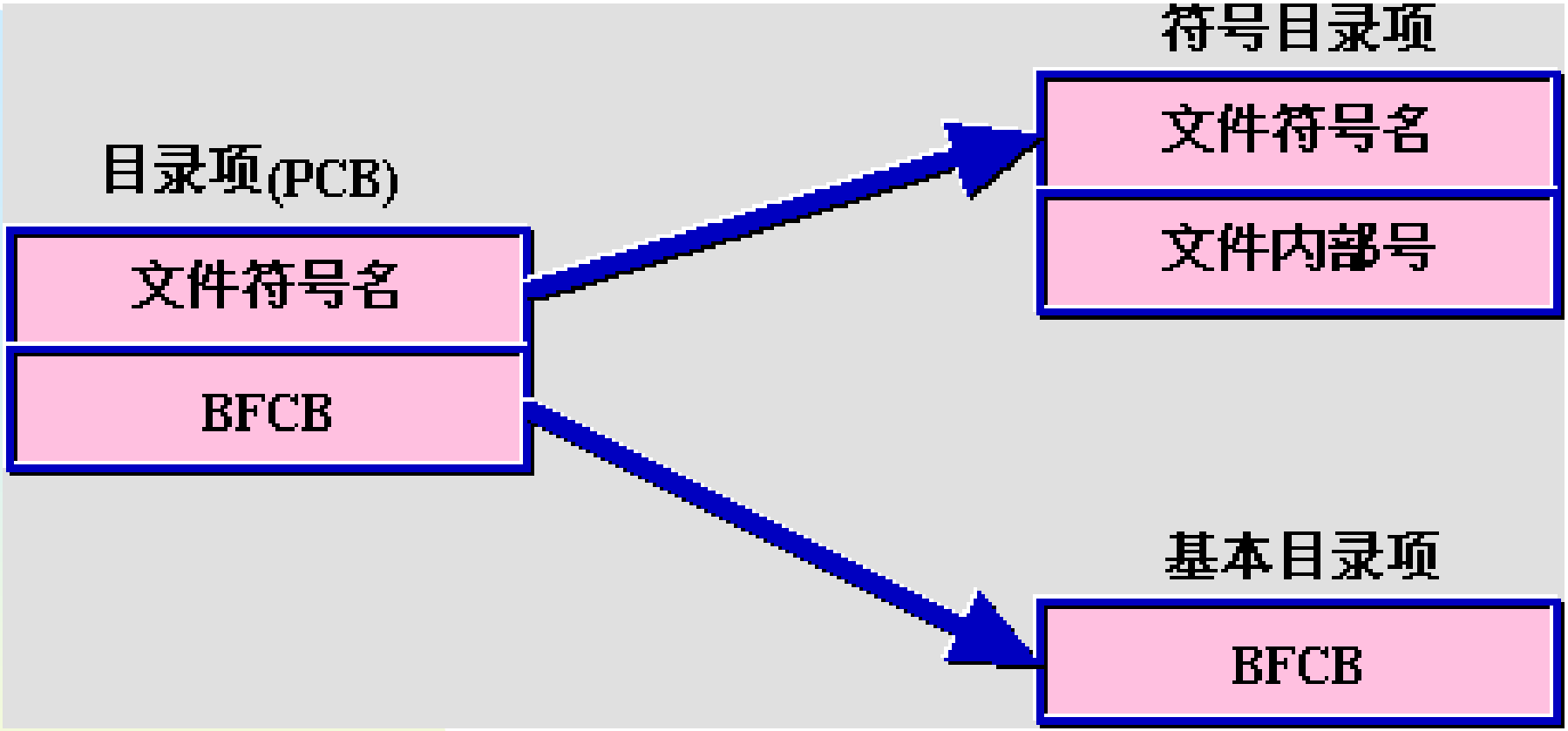
符号目录项　　（次部）

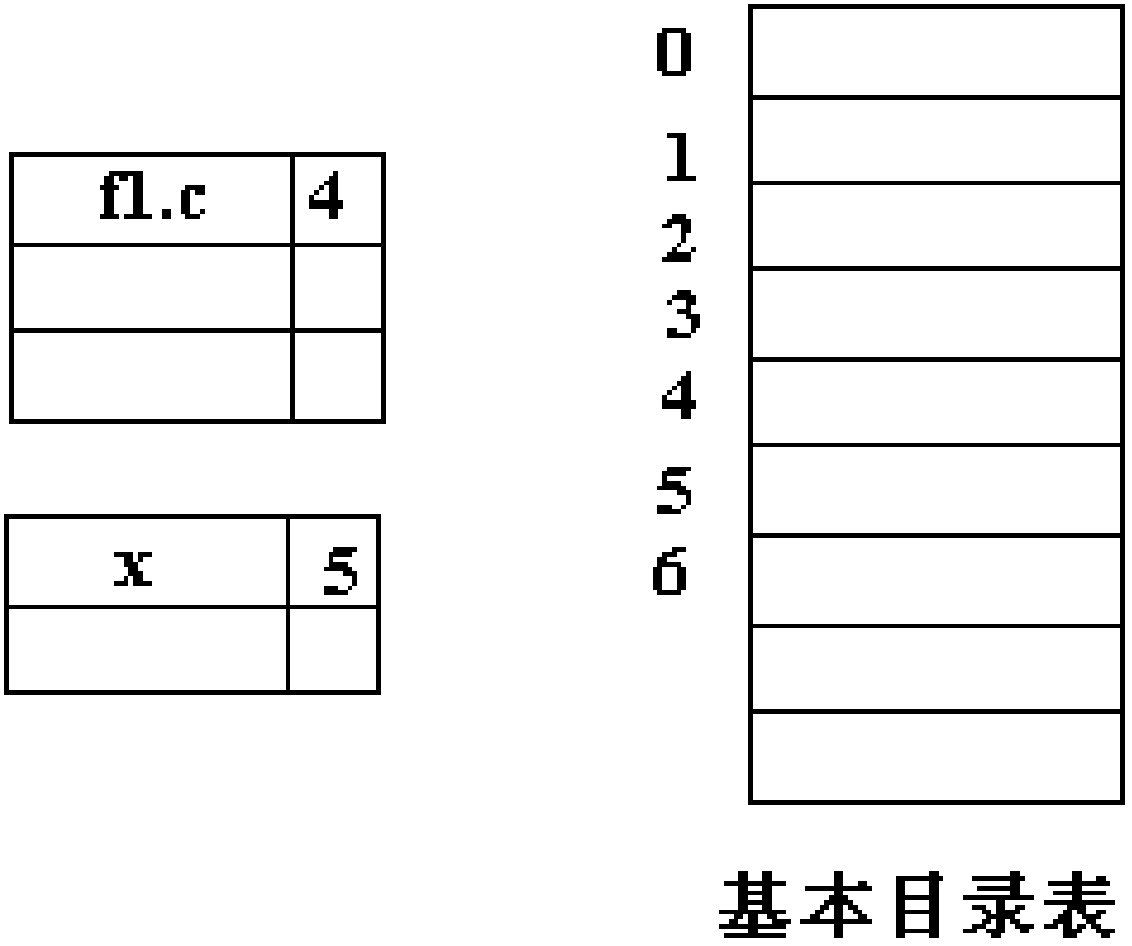
文件名，文件号

基本目录项　　（主部）

除文件名外的所有项目

UNIX：i节点（索引结点）


符号目录项
文件符号名
文件内部号
目录项(PCB)
文件符号名
BFCB
基本目录项
BFCB


fl.c
4
x
5
0
1
2
3
4
5
6
基本目录表

例子

一个FCB有48个字节

符号目录项占 8字节

文件名6字节，文件号2字节

基本目录项占 48-6=42字节

假设，物理块大小512字节

解

分解前： 占512/48=10个FCB

分解后： 占512/8=64个符号目录项或512/42=12个基本目录项

假设： 目录文件有128个目录项

分解前： 占13块

分解后： 符号文件占2块

基本文件占11块

查找一个文件的平均访盘次数

分解前：(1+13)/2=7次

分解后：(1+2)/2 +1 =2.5次

减少了访问硬盘的次数，提高了检索速度

7.5 文件的共享

所谓文件共享指系统允许多个用户或进程共享同一份文件。

在系统中只需要保存共享文件的一个副本。

如果系统不能提供文件共享功能，就意味着凡是需要该文件的用户都要自备此文件的副本。

文件共享的目的

节省存储空间

进程间通过文件交换信息

基于索引结点的文件共享

从一个目录项直接用一个指针（或编号）指向另一个目录项达到共享文件的目的。

这里需要扩充目录项，增加：用户计数。用于记录共享数量。

硬链接

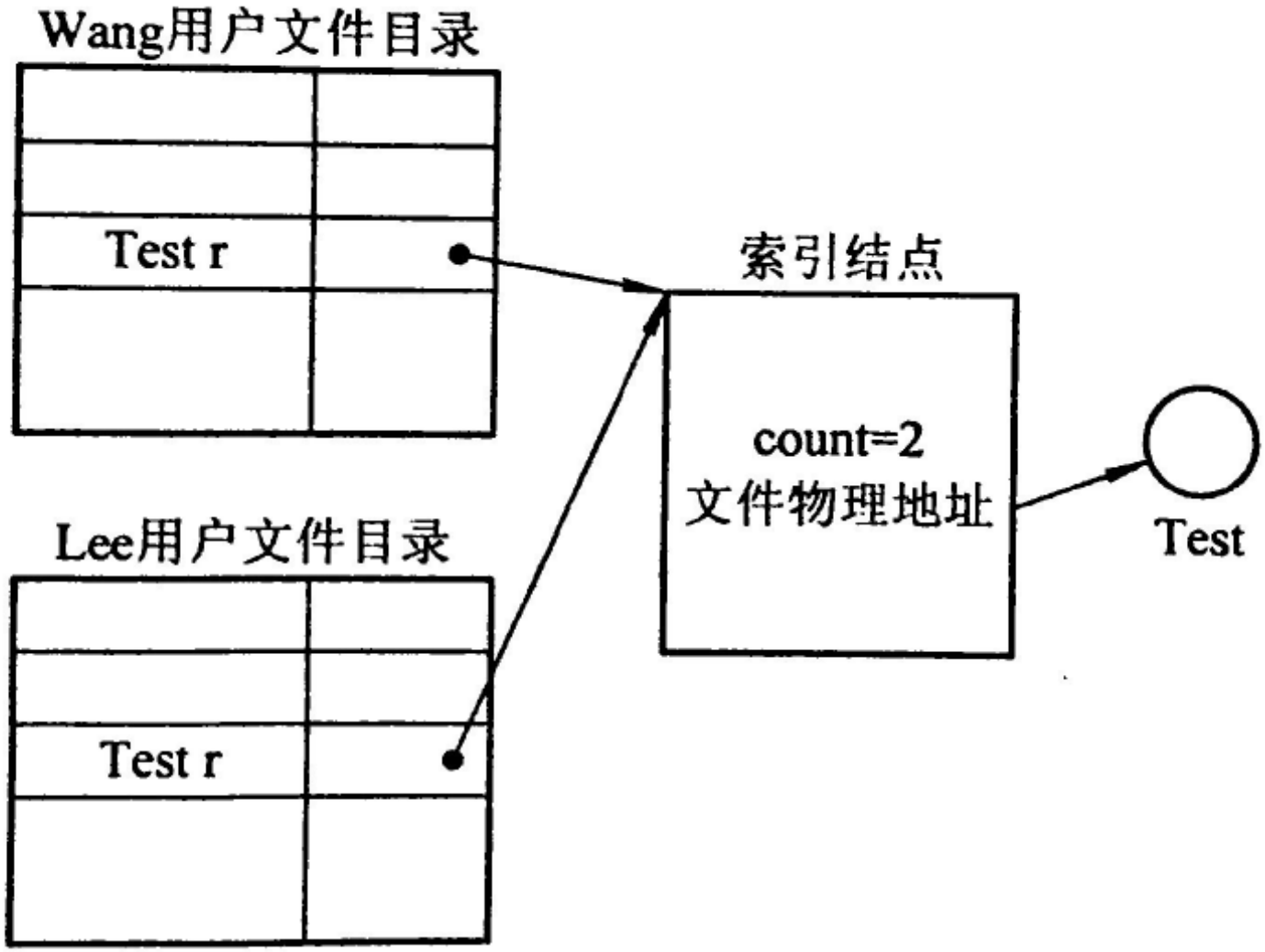


图 7-14　基于索引结点的共享方式

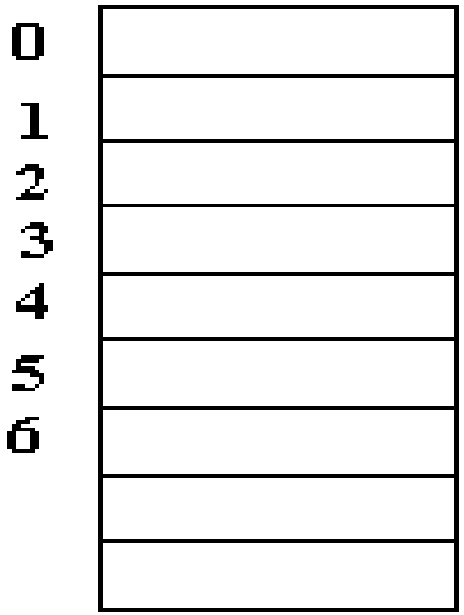
利用符号链实现文件共享

用户A为了共享用户B的Bboot目录下的一个文件f1.c，可以创建一个LINK类型的新文件x，新文件x中仅包含被链接文件f1.c的路径名。A用户对新文件x的访问被系统重定位去访问B的文件f1.c。

软链接

f1.c	4

x	5



基本目录表

文件x的内容：
/Bboot/f1.c

优势：计算机网络环境下可用

问题：系统开销大

7.6 文件的保护

对拥有权限的用户，应该让其进行相应操作，否则，应禁止。

7.6.1 对用户进行分类

7.6.2 对访问权限分类

7.6.3 用访问控制矩阵实现文件保护

7.6.4 存取控制表实现文件保护

7.6.5 用户权限表实现文件保护

7.6.6 用口令实现文件保护

7.6.1 对用户进行分类

按用户对文件访问权力的差别把用户分成几类，然后对每个文件规定各类用户的存取权限。通常将用户分成三类：

文件主

文件主的同组用户或合作用户

其它用户

7.6.2 对访问权限分类

对文件的访问系统首先要检查访问权限，只允许合法的用户访问。文件的存取权限一般有以下几种：

仅允许执行（E）。

仅允许读（R）。

仅允许写（W）

仅允许在文件尾写（A）

仅允许对文件进行修改（U）

允许改变文件的存取枚限（C）

允许取消文件（D）

这几种权限可进行适当的组合。

7.6.3 用访问控制矩阵实现文件保护

一维代表所有用户，一维代表系统中的所有文件。

优点：一目了然

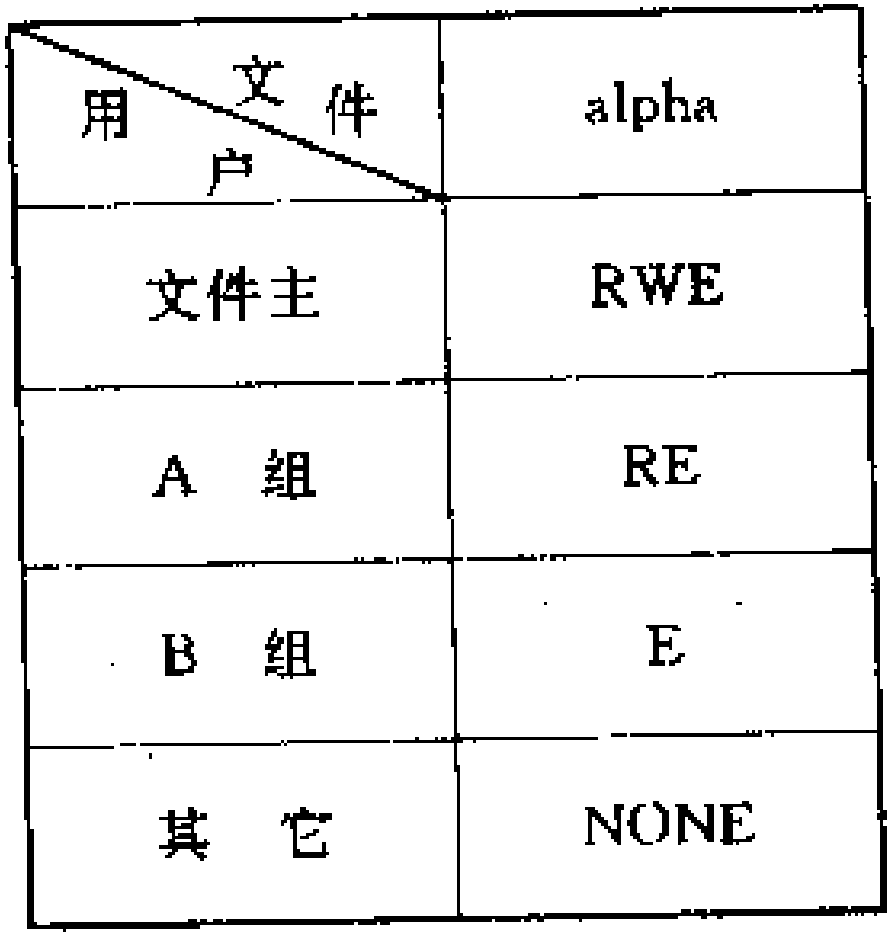
缺点：矩阵往往过大。

用户 \ 文件	1	2	3	4	5	6	7	8	9	10
1	0	0	1	1	0	0	0	0	1	0
2	0	0	1	0	1	0	0	0	0	0
3	0	1	0	1	0	1	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	1	1	0	0
7	1	0	0	0	0	0	0	0	0	1
8	1	0	0	0	0	0	0	0	0	0

访问控制矩阵

7.6.4 存取控制表实现文件保护

存取控制表



用户 \ 文件	alpha
文件主	RWE
A 组	RE
B 组	E
其 它	NONE

7.6.5 用户权限表实现文件保护

用户权限表

文件 \ 用户	A 组
sqrt	RE
test	R
alpha	RE
beta	R
⋮	⋮
abc	RW

7.6.6 用口令实现文件保护

用户为自己的每个文件规定一个口令，有口令者才能访问文件。

优点：简便

缺点：

保护级别少（可访问和不可访问）

保密性差。

不易改变存取控制权限。

7.7 磁盘容错技术

磁盘容错技术：

SFT-1：低级磁盘容错技术，主要用于防止磁盘表面发生缺陷所引起的数据丢失；

SFT-2：中级磁盘容错技术，主要用于防止磁盘驱动器和磁盘控制器故障引起的系统不能正常工作；

SFT-3：高级磁盘容错技术。

7.7.1 第一级容错技术

1、双份目录和双份文件分配表

文件目录和文件分配表是文件管理所需的重要数据结构。

在系统每次启动时都要进行两份目录和分配表的检查。

2、热修复重定向和写后读校验

磁盘表面有少量缺陷时，采取一些补救措施后可继续使用。这些措施主要用于防止将数据写入有缺陷的盘块中。

热修复重定向

写后读校验

热修复重定向

系统将一定的磁盘容量（如2%-3%）作为热修复重定向区。

例如：系统要向第3柱2头1扇区写数据，但发现该扇区是坏的时，便将数据写到热修复区（如200柱16头1扇区）。以后要读3柱2头1扇区的数据时，便从200柱16头1扇区中读。

写后读校验

为了保证所有写入到磁盘的数据都能写入完好的盘块中，应该在每次写数据时，又立即从磁盘上读出该块数据，并同写前的数据进行对比（校验）。若两者不一致，则认为盘块有缺陷，便将该数据写入到热修复区。并对该坏盘块进行登记。

7.7.2 第二级容错技术

第一级容错只能用于防止磁盘表面部分故障造成的数据丢失。如果磁盘驱动器或磁盘控制器发生故障，则第一级容错就无能为力了。

1、磁盘镜像

2、磁盘双工

1、磁盘镜像

磁盘驱动器故障的容错。

在同一磁盘控制器控制下，增设一个完全相同的磁盘驱动器。

每次将数据写主磁盘时，同时将数据也写入到备份磁盘。

一个磁盘驱动器发生故障时，必须立即发出警告，尽快修复。

磁盘利用率为**50%**

2、磁盘双工

将两台磁盘驱动器分别接到两个磁盘控制器上。

两个磁盘上的数据完全相同。

7.7.3 廉价磁盘冗余阵列

RAID(Redundant Arrays of Inexpensive Disk)。由伯克利提出，广泛用于大中型计算机和网络中。

由一台磁盘阵列控制器控制一组磁盘驱动器，组成一个高度可靠、快速的大容量磁盘系统。

1、并行交叉存取

2、RAID分级

3、RAID的优点

1、并行交叉存取

加快访问速度

在系统中有多台磁盘驱动器（N）。

存放数据时，将数据的第一块放在第一个磁盘上，第N块放在第N个磁盘上。

这样可以并行读写，极大地提高了速度。

2、RAID分级

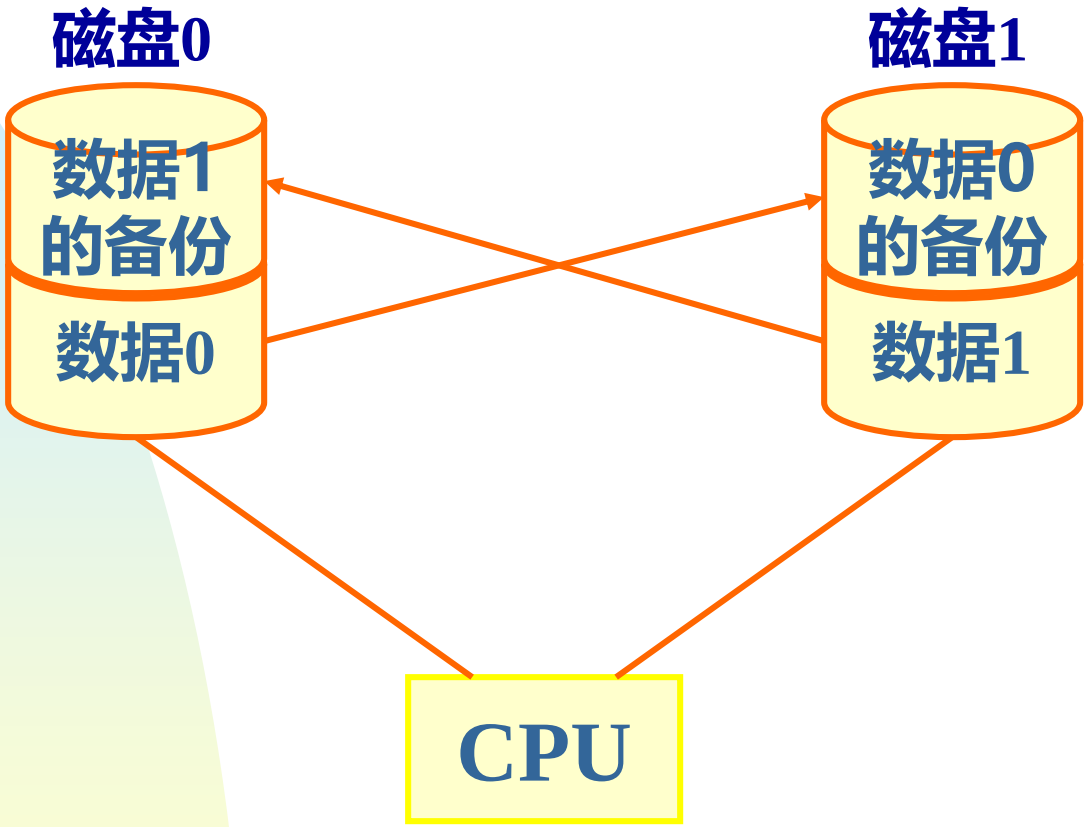
RAID0—RAID7

RAID0 提供并行交叉存取（没有冗余能力）

至少两个盘

RAID1 两个盘，并行交叉存取，并把一个磁盘的数据镜像到另一个磁盘上。利用率**50%**。比传统镜像盘快。

图



RAID3 利用一个校验盘来完成容错。

RAID5 无专门校验盘，校验数据分布在多个盘上。 一个磁盘故障时，控制器可从其他尚存的磁盘上重新恢复/生成丢失的数据而不影响数据的可用性 。常用于I/O较频繁的事务处理。

3、RAID的优点

可靠性高。除了RAID0，其余各级都采用了容错技术。某个磁盘损坏时，不会造成数据丢失。

速度快。可并行存取。

性能/价格比高。利用RAID技术实现的大容量快速存储器，同大型磁盘系统相比，体积和价格都是后都的1/3。可靠性更高

7.7.4 后备系统

容量和安全性考虑，需要后备系统。

1、 后备系统的类型

2 、拷贝方法

1、后备系统的类型

磁带机： 最广泛。

硬盘：

光盘： 很有前途。

2、拷贝方法

完全转储：定期将所有文件拷贝到后援存储器

增量转储：只转储修改过的文件，即两次备份之间的修改，减少系统开销

7.8 文件系统性能的改善

文件系统的性能可表现在多个方面

文件的访问速度

数据的可共享性

文件系统使用的方便性

数据的安全和一致性。

提高磁盘I/O速度的方法

7.8.1 磁盘高速缓存

7.8.2 优化数据分布

7.8.3 其它方法

7.8.1 磁盘高速缓存

磁盘的I/O速度要比内存低4-6个数量级

分配一些内存作为磁盘高速缓存可以极大地提高磁盘I/O速度。

磁盘高速缓存的形式

在内存中开辟一个单独的存储空间作为磁盘高速缓存。

把所有未利用的内存空间变为一个缓冲池，供分页系统和磁盘I/O共享。

置换算法

如果高速缓存已满，则需要进行淘汰。

常用置换算法：最近最久未使用LRU、最少使用LFU等。

周期性写回磁盘

LRU算法中，那些经常被访问的盘块可能会一直保留在高速缓存中，而长期不被写回磁盘中。留下了安全隐患。

解决之道：周期性写回。周期性地强行将已修改盘块写回磁盘。周期一般为几十秒。

7.8.2 优化数据的分布

优化物理块的分布

优化索引结点的分布

优化物理块的分布

优化一个文件的物理块分布，使访问该文件时，磁头的移动距离最小。

物理块连续分配可以减少磁头的移动。

增加物理块的大小也可减少磁头的移动。

优化索引结点的分布

访问文件时，先要访问索引结点，然后再访问文件数据。以前一般将索引结点集中放在磁盘的开始部分，使得索引结点同文件数据之间的平均距离是磁道数的一半。

因此可将索引结点放在中间位置。

进一步可将磁道分组，每组都有索引结点和文件数据。

7.8.3 提高磁盘I/O速度的其它方法

提前读

延迟写

虚拟盘

提前读

在访问文件时经常是顺序访问，因此在读当前块时可以提前读出下一块。

提前读已经被广泛应用：**UNIX**、**OS/2** 、**Netware**等。

延迟写

修改缓存中的数据后一般应立即写回磁盘，但该盘块可能还会被修改，立即写回会带来很大的开销。

置上延迟写标志。直到该盘块淘汰时或周期性写回时。

延迟写也被广泛应用： **UNIX、OS/2** 等。

虚拟盘

利用内存仿真磁盘，又称RAM盘。

虚拟盘同磁盘高速缓存的区别：
虚拟盘的内容完全由用户控制，用户可见。
缓存的内容完全由系统控制，用户不可见。

7.9 数据一致性控制

数据一致性的概念在数据库中出现较多。

7.9.1 事务

7.9.2 检查点

7.9.3 并发控制

7.9.1 事务

1、事务（Transaction）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。

事务具有原子性：事务的操作要么全部完成，要么一个也不做。

事务的操作全部完成时要执行提交（commit）操作。事务失败时要执行退回（Rolled back）操作。

2、事务记录

记录事务运行时所有对数据项的修改信息。

又称运行日志（**Log**）。

该记录包括：

事务名　事务的唯一标识

数据项名　被修改的数据项标识

旧值

新值

当一个事务提交时，将一个提交记录也写入事务记录表中。

3、恢复算法

当系统发生故障后，利用事务记录进行故障恢复。

搜索整个事务记录表：

对于已经提交了的事务，执行**redo**操作

对于未提交事务，执行**undo**操作

7.9.2 检查点

1、检查点（Check points）的作用

随着系统的运行，事务记录表会变得越来越大，这样当发生故障时，搜索整个事务记录表来进行恢复就是一件非常费时的工作。因而引入检查点。

系统每隔一段时间便写一条检查点记录到事务记录表，并进行恢复工作，即：对已经提交的事务执行**redo**，未提交的事务执行**undo**。

这样在检查点这个时刻，系统中数据的一致性和完整性肯定能得到保证。

2、新的恢复算法

在引入检查点后，当发生故障后，只需对最后一个检查点以后开始的事务执行恢复工作。

7.9.3 并发控制

在多进程或多用户系统中，可能有多个事务在并发执行，这些事务对数据的修改应该是互斥的。（参见第四章进程同步）

可以利用**PV**操作来实现互斥。

但在数据库和文件服务器中，应用得最多的还是较简单且灵活的同步机制：锁。

1、互斥锁

当事务访问一数据项时，给它上锁，访问完后开锁。

2、互斥锁和共享锁

互斥锁可以简单实现事务的并发控制，但会影响并发度。

因为当一个事务读一个数据项进，另一事务应该也能读同一数据项，但上互斥锁时没有这种可能。

所以引入共享锁：事务读对象时申请共享锁，若该对象未上锁或上的是共享锁，则可以申请到。事务写对象时申请互斥锁，只有对象未上锁时才能申请到。