



第6章 文件系统

□ 文件系统的功能/需解决的问题

- ❖ 从系统角度看:负责为用户建立、删除、读写、修改和复制文件
- ❖ 从用户的角度看: 实现了按名存取



第6章 文件系统的主要内容

- 文件和文件系统
- 文件逻辑结构
- 外存分配方式
- 目录管理
- 文件共享与文件保护
- 数据一致性控制


本章作业


重点：文件种类；文件的逻辑结构和物理结构；
目录管理；数据一致性；文件保护

难点：目录文件与FCB（通过WINDOWS目录理解），文件保护（类比实例）



6.1 文件和文件系统

 文件、记录和数据项（数据的组成）

 文件类型和文件系统模型

 文件操作

一、文件、记录和数据项

文件

记录在外存上的具有文件名的一组相关信息的集合。

有结构文件

无结构文件

记录

字符流

一组相关数据项的集合

数据项

基本数据项
组合数据项

❖ 文件

❖ 文件属性

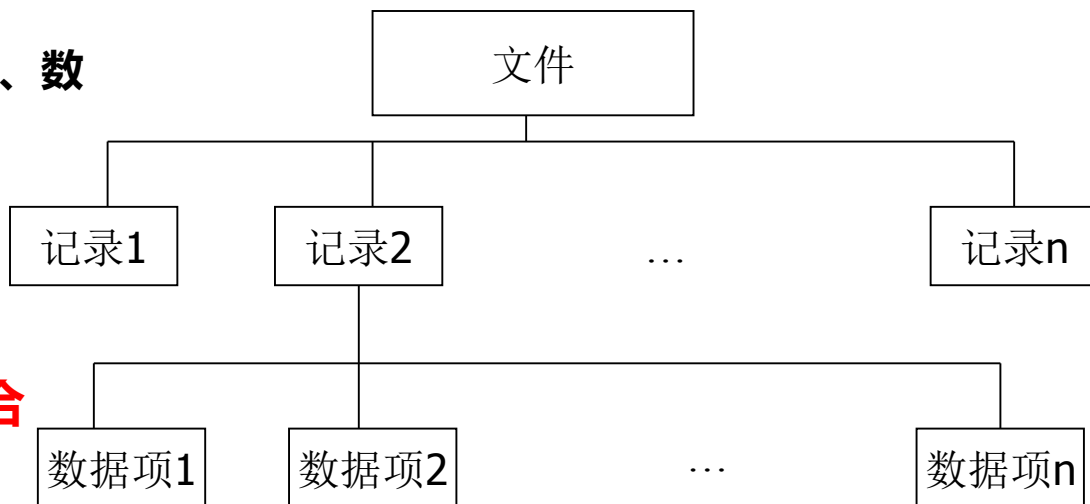
- 文件名、文件类型、文件长度、文件的物理位置、文件的建立日期以及用户对该文件的存取权限等

❖ 文件表示的范围/包含的内容

- 源程序、二进制代码、文本文档、数据、表格、声音和图像等。

❖ 文件的特点*

- 文件具有保存性
- 文件是按名存取
- 文件的内容是一组信息的集合



文件、记录和数据项间的层次关系



二、文件类型 --- 文件名. 扩展名

- 按用途分

- 系统文件
- 用户文件
- 库文件

- 按数据形式分

- 源文件
- 目标文件
- 可执行文件

- 按存取控制属性

- 只读文件
- 读写文件
- 只执行文件
- 不保护文件

- 按文件的逻辑结构分

- 有结构文件（记录式文件）
- 无结构文件（流式文件）

- 按文件的物理结构分

- 顺序文件
- 链接文件
- 索引文件

- 按信息流向分

- 输入文件
- 输出文件
- 输入输出文件

三、文件系统模型

| 文件系统接口 | |
|------------------------|--------------------|
| 对对象操纵 和管理的软件 集合 | 逻辑文件系统层 |
| | 基本I/O管理程序层（文件组织模块） |
| | 基本文件系统层（物理I/O层） |
| | I/O控制层（设备驱动程序） |
| 对象(文件、目录及磁盘存储空间)及其属性说明 | |

- **逻辑文件系统层**：处理文件及记录的相关操作（访问、保护及目录操作）。
- **基本I/O管理程序层**：完成大量与磁盘I/O有关的工作（选择设备，逻辑块号到物理块号的转换，空闲空间管理等）。
- **基本文件系统层**：负责内存与磁盘间的数据块交换（在外存及内存缓冲区的位置）。
- **I/O控制层**：负责启动I/O操作及处理设备发来的中断信号。

四、文件操作

用户通过文件系统所提供的**系统调用**实施对文件的操作。

最基本的操作有：

- 对记录的操作：检索、插入、修改、删除
- 对文件的操作
 - 最基本的：创建、**打开**、**关闭**、删除、读、写、截断
 - 其它的：文件属性类操作、目录类操作

文件的“打开”和“关闭”操作

“打开”：系统将文件的属性（目录信息）从外存复制到内存**打开文件表**中，并返回该表目的编号给用户，建立了用户与文件间的联系。以后若再访问此文件，则利用编号直接在内存中检索，从而节省大量的检索开销，提高了文件的操作速度。

“关闭”：当用户不再需要对该文件的操作时，系统利用关闭文件将文件的属性从内存打开表中删除，从而切断用户与文件间的联系。



6.2

文件逻辑结构

对任一文件存在着两种形式的结构：

 文件的逻辑结构（文件组织）-- 从用户观点出发

 *文件的物理结构（文件的存储结构）

文件在**外存上的存储组织形式**，与存储介质的存储性能有关。
（分为**顺序、链接及索引结构**）

注：文件的逻辑结构和物理结构都将影响文件的检索速度。



文件逻辑结构

对文件的逻辑结构提出的要求：

提高检索；便于修改；降低文件存储费用。

- 文件逻辑结构的类型
- 顺序文件
- 索引文件
- 索引顺序文件



一、文件逻辑结构的类型

❖ 有结构的记录式文件

- 文件构成：由一个以上的记录构成。
- 记录长度：分为定长和变长。
- 分类（按记录的组织）：顺序文件
索引文件
索引顺序文件

❖ 无结构的流式文件

- 文件构成：由字符流构成。
- 长度：字节为单位
- 访问：读写指针
- 注：Unix中所有文件视为流式文件



二、顺序文件

逻辑记录的排序

 串结构：记录顺序与关键字无关，按存入时间的先后排列。

 顺序结构：记录顺序按关键字排列。

对顺序文件的读、写操作

 记录为定长的顺序文件

 记录为变长的顺序文件

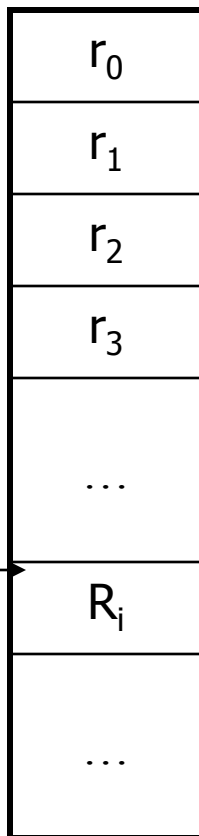


$Rptr = Rptr + 1$

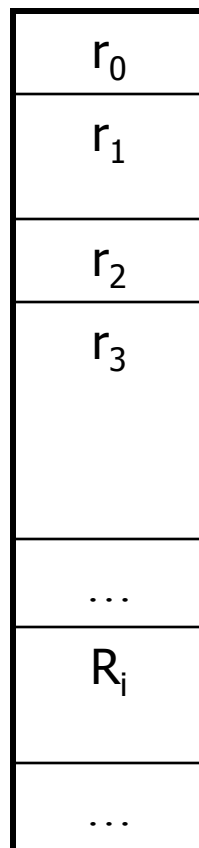
$Rptr = Rptr + L_i$

读指针

$Rptr$



$Rptr$



顺序文件的优缺点

优

- 顺序存取速度较快（批量存取）。
- 对定长记录，还可方便实现直接存取。

缺

 对变长记录，直接存取低效

 不利于文件的动态增长。

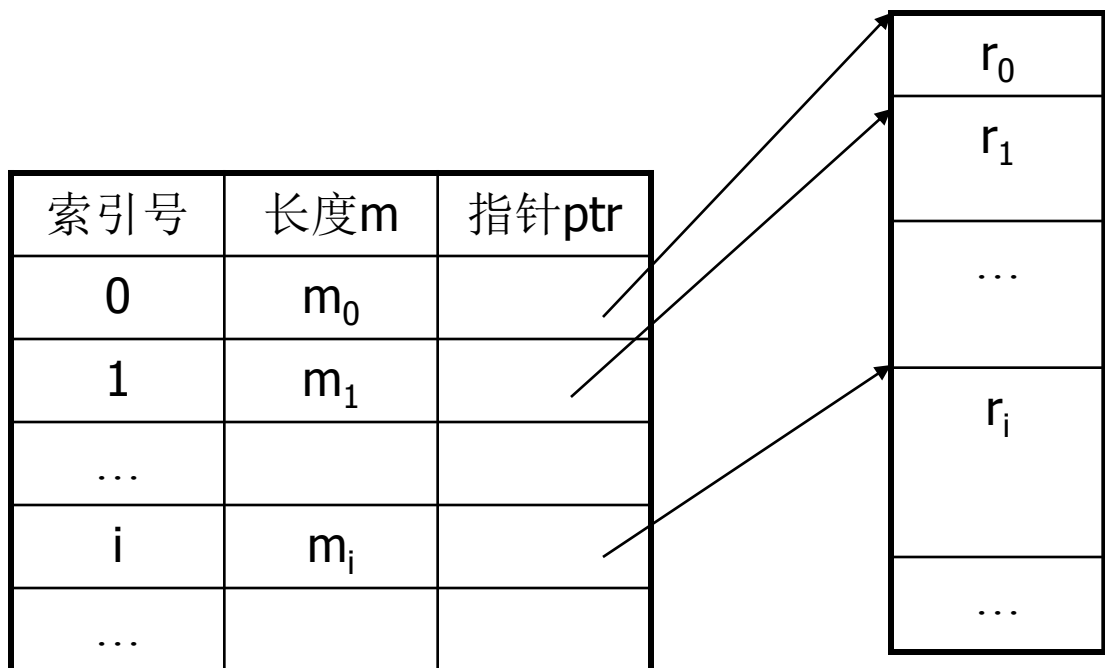
三、索引文件

引入

为解决变长记录文件的直接存取低效问题。

索引文件

为变长记录文件建立一张索引表。



索引表

逻辑文件



索引文件的特点

- 优点
 - 通过索引表可方便地实现直接存取，具有较快的检索速度。
 - 易于进行文件的增删。
- 缺点
 - 索引表的使用增加了存储费用;
 - 索引表的查找策略对文件系统的效率影响很大.
- 注：若索引表很大，可建多级索引

四、索引顺序文件

引入

为解决变长记录文件的直接存取低效且存储费用增加的问题。

索引文件

为顺序文件建立一张索引表。

| 索引号 | 长度m | 指针ptr |
|-----|-------|-------|
| 0 | m_0 | |
| 1 | m_1 | |
| ... | | |
| i | m_i | |
| ... | | |

索引表

| |
|-------|
| r_0 |
| r_1 |
| ... |
| r_i |
| ... |

逻辑文件



索引顺序文件的特点

- 优点

- 通过索引表可方便地实现直接存取，具有较快的检索速度。
- 易于进行文件的增删。

- 缺点

- 索引表的查找策略对文件系统的效率影响很大。



6.3 外存分配方式

- 文件存储单位：簇（cluster）/盘块

文件的存储空间通常由多个分立的簇组成，而每个簇包含若干个连续的扇区(sector)/块。

- 目前常用的外存分配方法：

（1）连续分配（顺序分配）

（2）链接分配

（3）索引分配



(1) 外存分配方法-连续/顺序分配

- Figure 6-7
- 为每一个文件分配一片连续的磁盘块/簇
- 只需要起始块/簇号和长度，适用于预分配方法
- 可以随机存取
- 文件不能增长
- 从逻辑地址映射到物理地址较简单
- 浪费空间：动态存储分配问题
- 可以通过紧缩(compact)将外存空闲空间合并成连续的区域。



连续/顺序分配的主要优缺点

- 主要优点

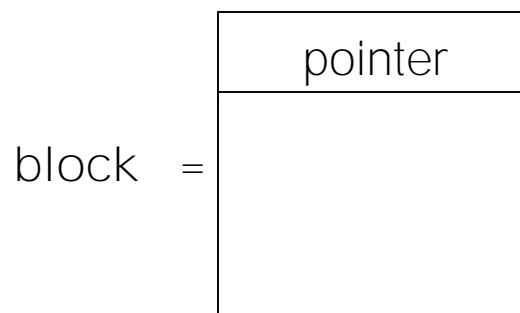
- 顺序访问容易
- 顺序访问速度快

- 缺点

- 要求有连续的存储空间
- 必须事先知道文件的长度
- 存在外部碎片

(2) 外存分配方法-链接分配

- Figure 6-8
- 每个文件是一个磁盘块的链接列表：块可以分散在磁盘各处
- 按所需分配磁盘块，链接在一起
- 在每个块中有指向下一个块的指针
- 只需要起始地址
- 可以通过合并(consolidation)将一个文件的各个簇连续存放，以提高I/O访问性能。





链接分配的优缺点

■ 优点

- 1、无外部碎片，没有磁盘空间浪费
- 2、无需事先知道文件大小。文件动态增长时，可动态分配空闲块。对文件的增、删、改十分方便。
- *3、不需紧缩磁盘空间。

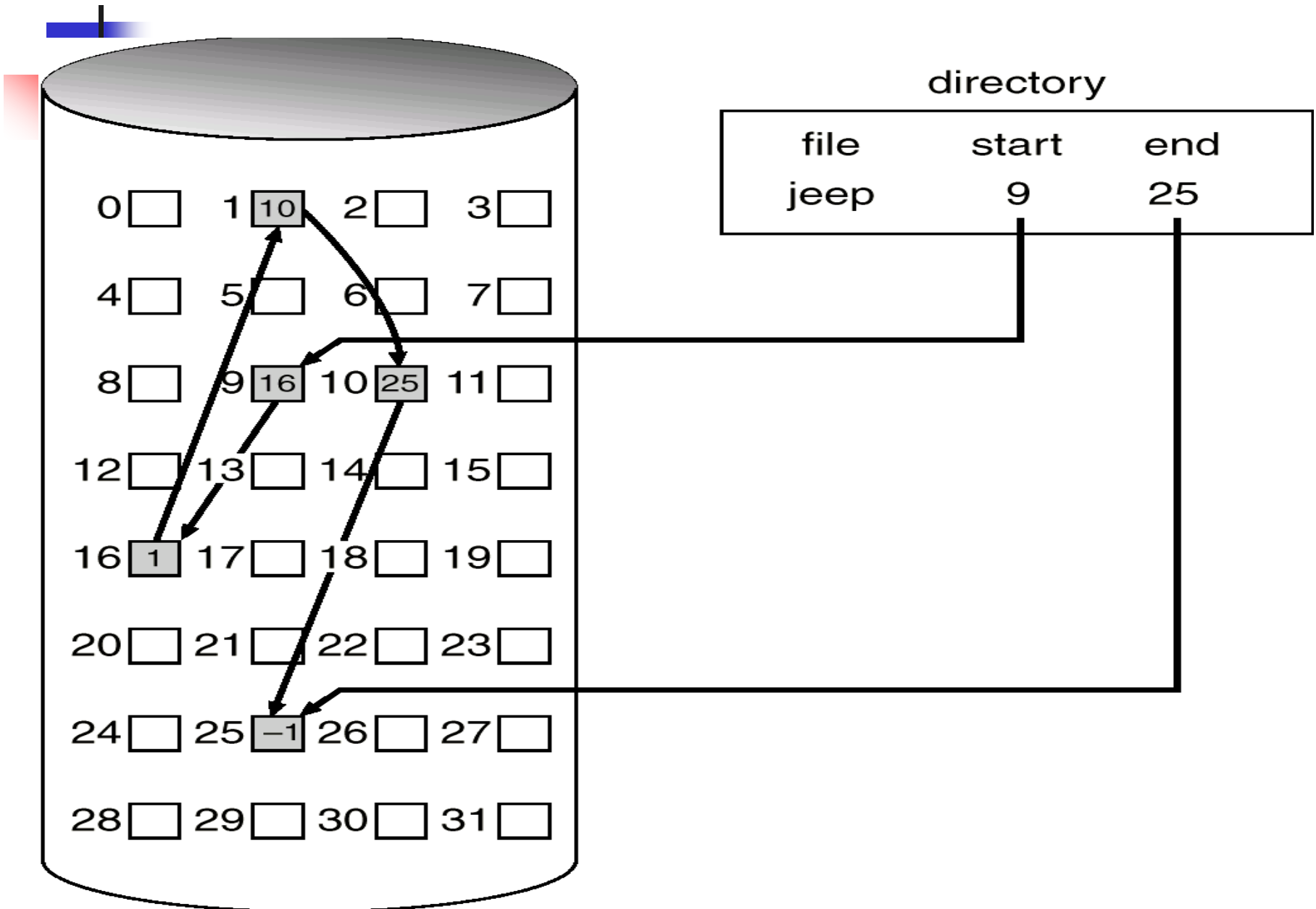
■ 缺点

- 1、不能支持高效随机/直接访问，仅对顺序存取特有效
 - 2、需为指针分配空间。---~~块~~—~~簇~~
 - 3、可靠性较低（指针丢失/损害） → （显式链接如
文件分配表FAT Figure 6-9所示）
- FAT需占用较大的内存空间。



文件分配表 *FAT*--figure 6-9、 6-10

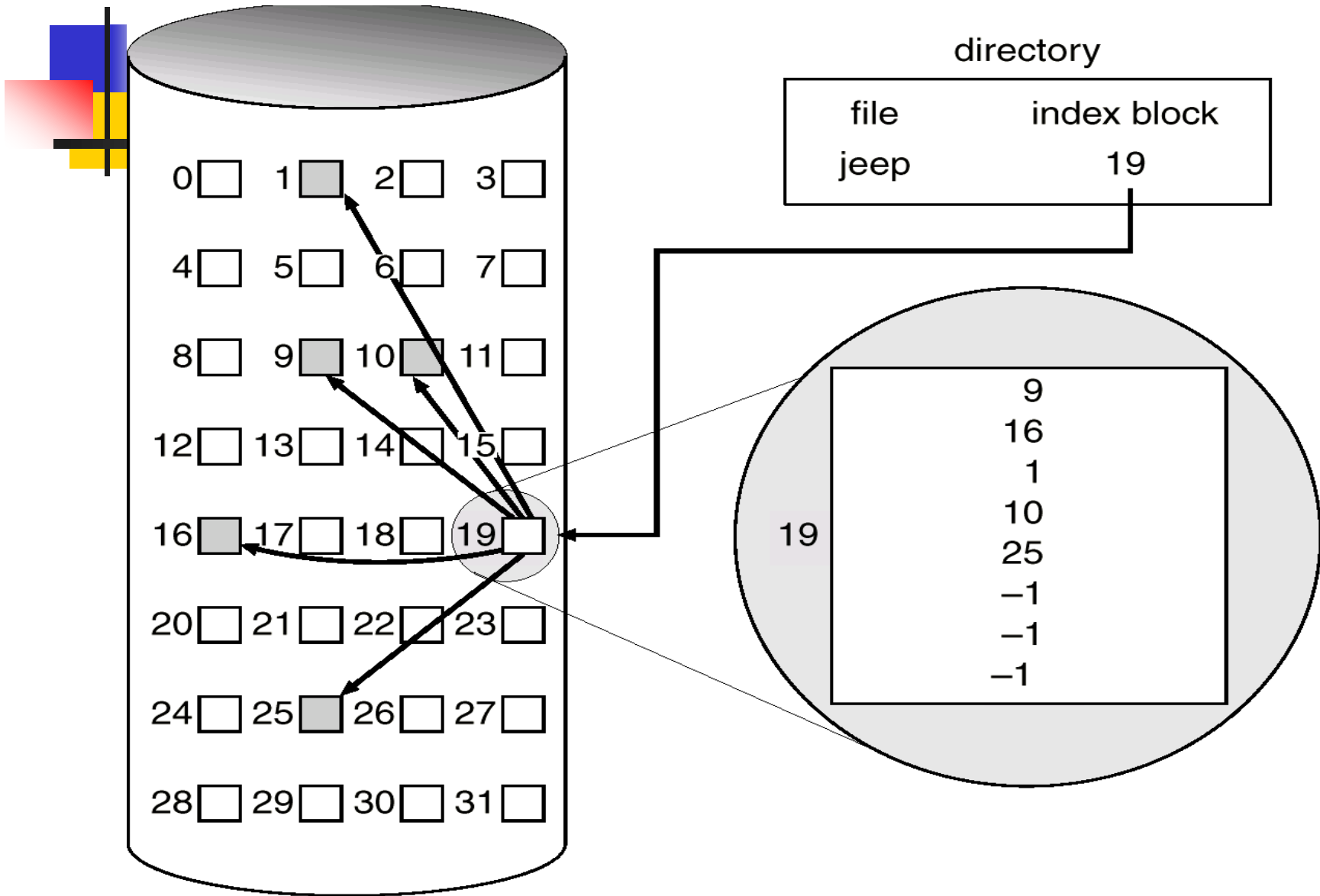
- 用于链接文件各物理块的链接指针，显式地存放在内存的一张链接表中。
- 该表在整个磁盘仅设置一张。
- 表序号为整个磁盘的物理块号 (0---(n-1))
- 表项存入链接指针，即下一个块号。
- 文件的首块号存入相应文件的FCB中。
- 查找在内存的FAT中，故提高了检索速度，同时又减少磁盘的访问次数。
- 被MS-DOS和OS/2等所采用。 P195 Figure 6-10





(3) 外存分配方法-索引分配

- Figure 6-11
- 为每一个文件分配一个索引块（表），再把分配给该文件的所有块号，都记录在该索引块中。故索引块就是一个含有许多块号地址的数组。
- 该索引块的地址由该文件的目录项指出。
- 支持随机/直接存取。
- 不会产生外部碎片。
- 适用于文件较大时。



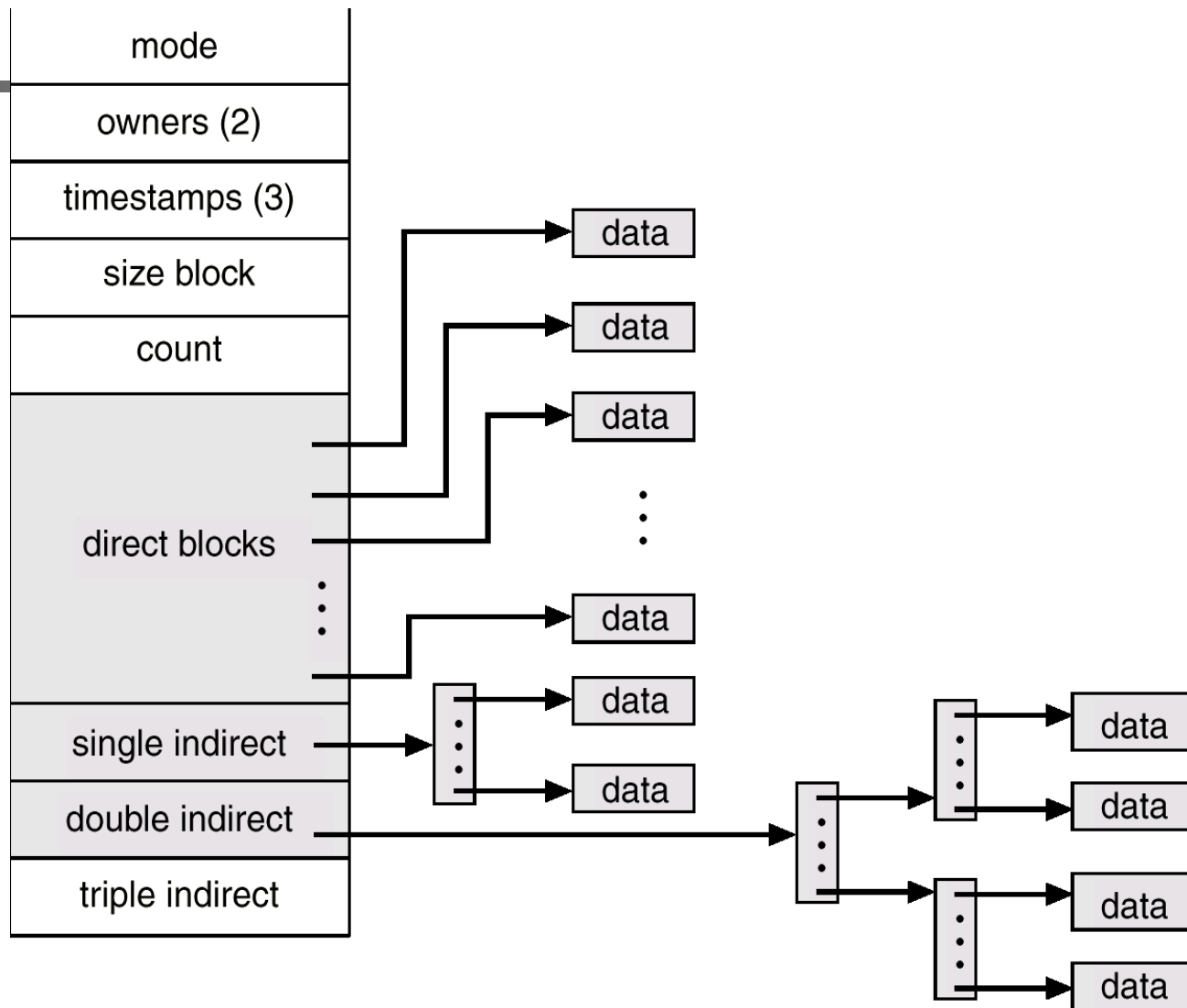
索引分配的几种方式

- 直接索引分配
- 多级索引分配
- 混合索引分配

Figure6-12

UNIX

(4K字节每个块)





6.4 目录管理

对文件目录的管理要求

 实现“按名存取”

 提高对目录的检索速度

 文件共享


 允许文件重名

 文件控制块和索引结点

 单级目录结构

 两级目录结构

 树型目录结构

 目录查询技术



文件控制块和索引结点

从文件管理角度看，文件由**FCB**和**文件体**（文件本身）两部分组成。

■ 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，存放了文件的**有关说明信息**，是文件存在的标志。
- FCB中的信息
 - 基本信息类：文件名、文件长度、类型、属性文件物理位置
 - 存取控制信息类：文件存取权限、用户名、口令、共享计数
 - 使用信息类：文件的建立日期、最后修改日期、保存期限、最后访问日期，



文件控制块（FCB）

- **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

文件控制块和索引结点

索引结点

索引结点引入

磁盘索引结点

存放在磁盘上的索引结点.

(主标识、类型、存取权限、物理地址、长度、连接计数、存取时间)

内存索引结点

存放在内存上的索引结点

(索引结点编号、状态、访问计数、逻辑设备号、链接指针)

| 文件名 | 索引结点编号 |
|-------|--------|
| 文件名1 | |
| 文件名2 | |
| | |



单级目录结构

- 在整个系统中只建立**一张目录表**

| 文件名 | 状态位 | 物理地址 | 文件其它属性 |
|--------|-----|------|--------|
| Alpha | | | |
| Report | | | |
| Text | | | |
| | | | |

优点: 简单，易实现按名存取

缺点: 限制了用户对文件的命名（即易重名）

文件平均检索时间长(查找速度慢)

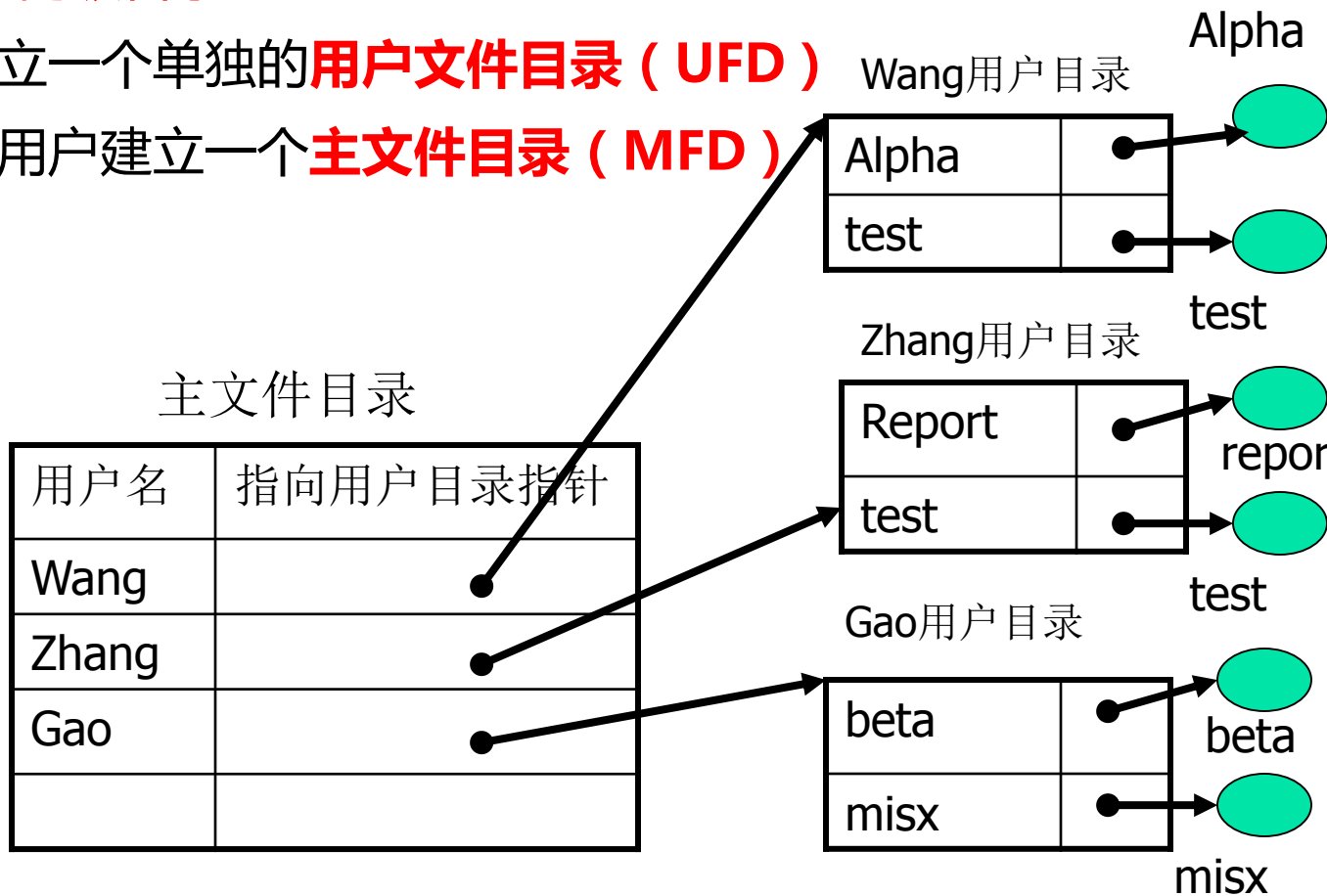
不便于实现文件共享

只适用于单用户环境

两级目录结构

在整个系统中建立两级目录

- 为每个用户建立一个单独的**用户文件目录 (UFD)**
- 系统中为所有用户建立一个**主文件目录 (MFD)**



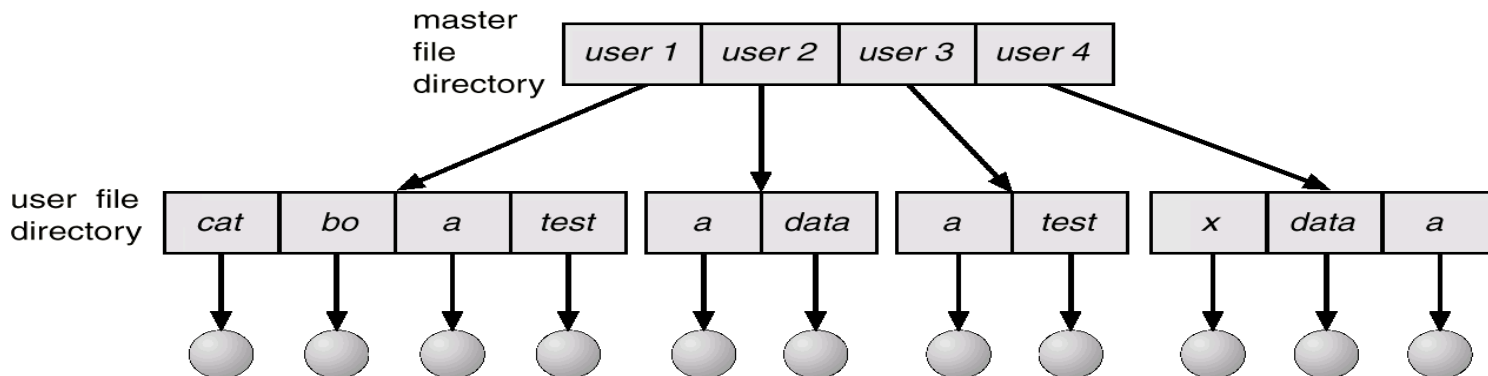
两级目录结构

优点:

- 提高了检索目录的速度；
- 不同用户目录中可重名；
- 不同用户可用不同文件名来访问系统中一共享文件

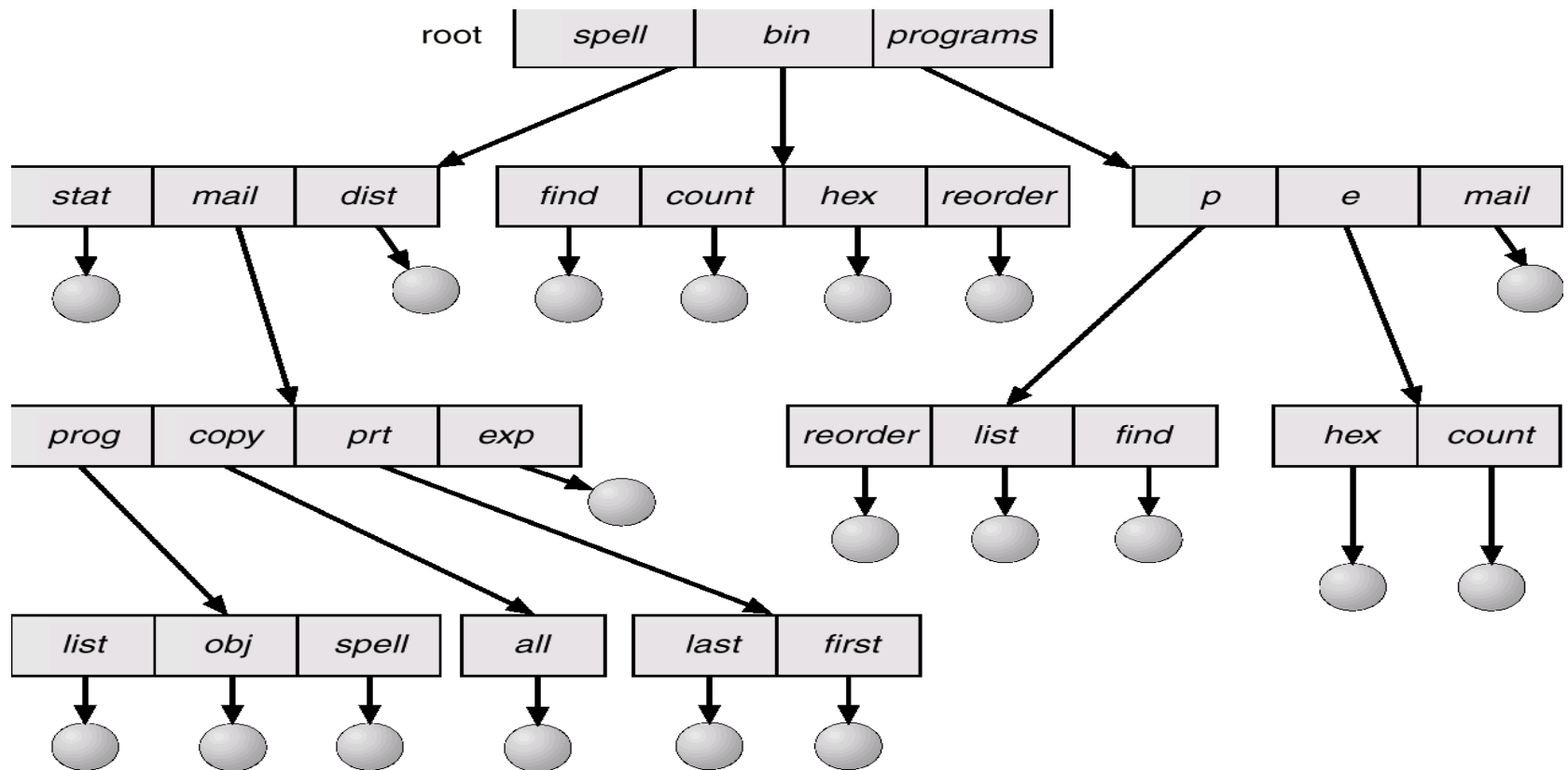
缺点:

- 限制了各用户对文件的共享
- 增加了系统开销，缺乏灵活性，无法反映真实世界复杂的文件结构形式。

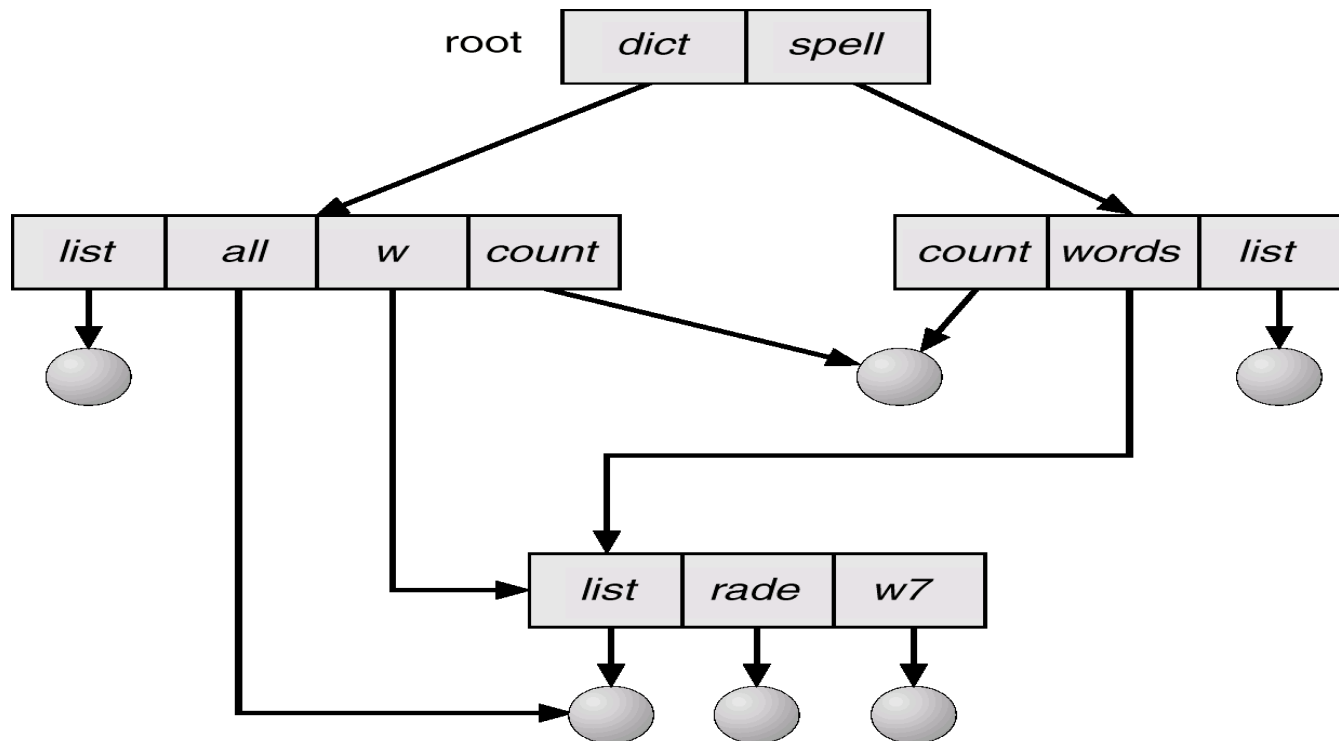


树型目录结构

- 在两级目录中若允许用户建立自己的子目录，则形成3级或多级目录结构（即树型目录结构）



有共享的子目录和文件





树型目录结构

路径名

- 访问数据文件的一条路径。
- 绝对路径、相对路径

当前目录

增加和删除目录

优点

- 层次结构清晰,实现分组,便于管理和保护 ;
- 解决重名问题 ;
- 查找速度加快

缺点

- 查找一个文件按路径名逐层检查,由于每个文件都放在外存,多次访盘影响速度



目录查询技术

数据文件（按名存取）的查询步骤

- 根据用户提供的文件名，对文件目录进行查询，找到该文件的FCB（索引结点）
- 根据FCB（索引结点）所记录的磁盘盘块号，换算出文件在磁盘上的物理位置
- 启动磁盘驱动程序，读该数据文件至内存中。

对目录进行查询的方式

- 线性检索法（顺序检索法）
- Hash方法

目录查询技术-线性检索法 (顺序检索法)

\usr\ast\mbox

根目录

| | |
|----|-----|
| 1 | . |
| 1 | .. |
| 4 | bin |
| 7 | Dev |
| 14 | Lib |
| 9 | Etc |
| 6 | Usr |
| 8 | tmp |

结点6是
\usr的目录

| |
|-----|
| 132 |
|-----|

132#块是
\usr的目录

| | |
|----|------|
| 6 | . |
| 1 | .. |
| 19 | Dick |
| 30 | Erik |
| 51 | Jim |
| 26 | Ast |
| 45 | bal |

结点26是
\usr\ast目录

| |
|-----|
| 496 |
|-----|

496#块是
\usr\ast目录

| | |
|----|--------|
| 26 | . |
| 6 | .. |
| 64 | grants |
| 92 | books |
| 60 | mbox |
| 81 | minix |
| 17 | src |



目录查询技术-Hash方法

- 建立一个Hash索引文件目录，系统利用用户提供的文件名，将它变换为文件目录的索引值，再利用该索引值到目录中去查找，从而找到文件的物理地址。

注：1) 当文件名中用了*，?时，系统无法利用Hash法检索目录，这时须用线性检索法查找目录。

2) 在Hash法中须对“冲突”进行处理。

3) 若在Hash索引文件目录中查询时，相应的目录项为空，则表“文件未找到”。



6.5 文件共享

早期实现文件共享的方法

绕弯路法（低效）

允许每个用户获得一“当前目录”，用户所访问的所有文件均相对于当前目录，若不在，则“向上走”绕弯路去访问其上级目录。

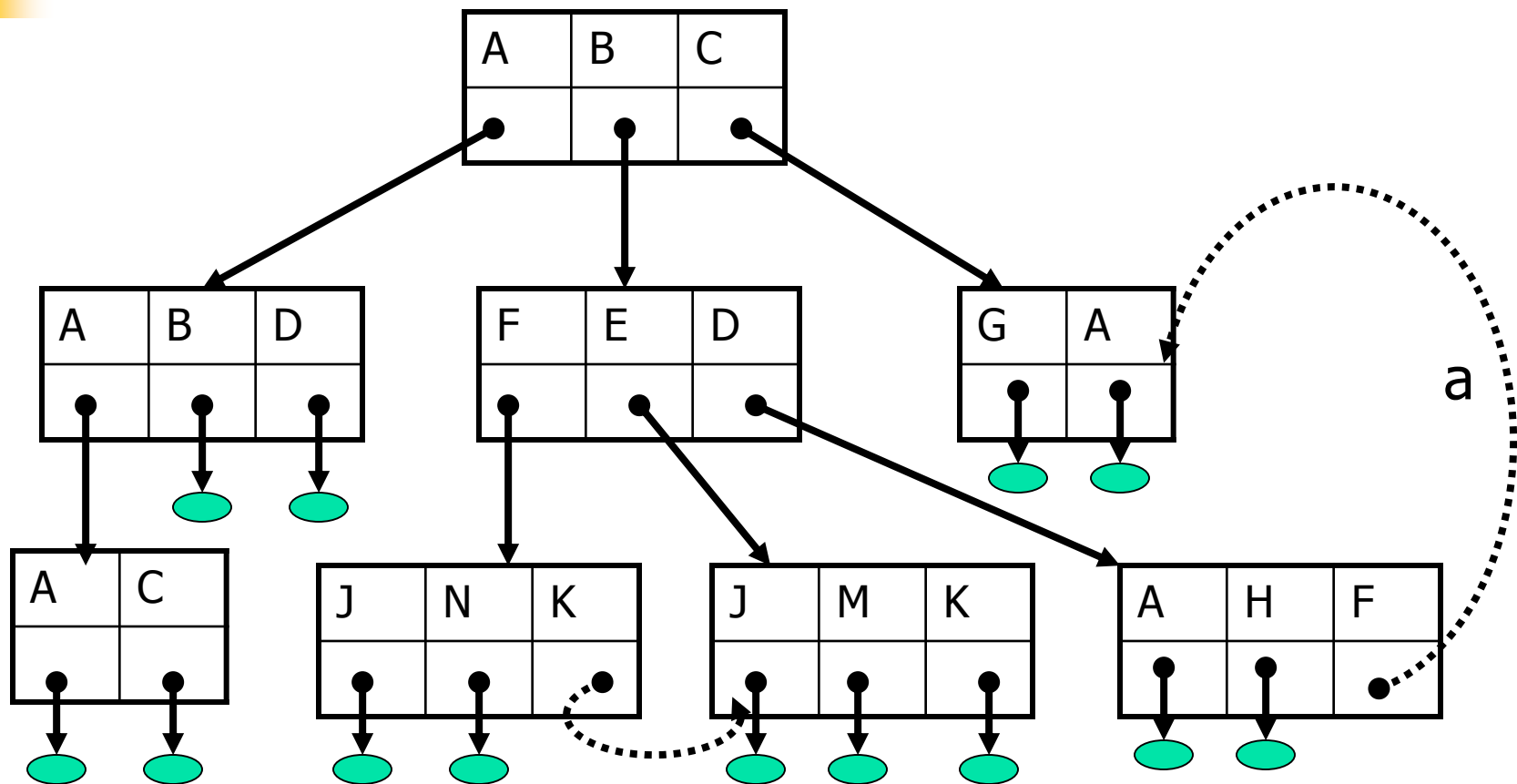
连访法

利用基本文件目录实现文件共享

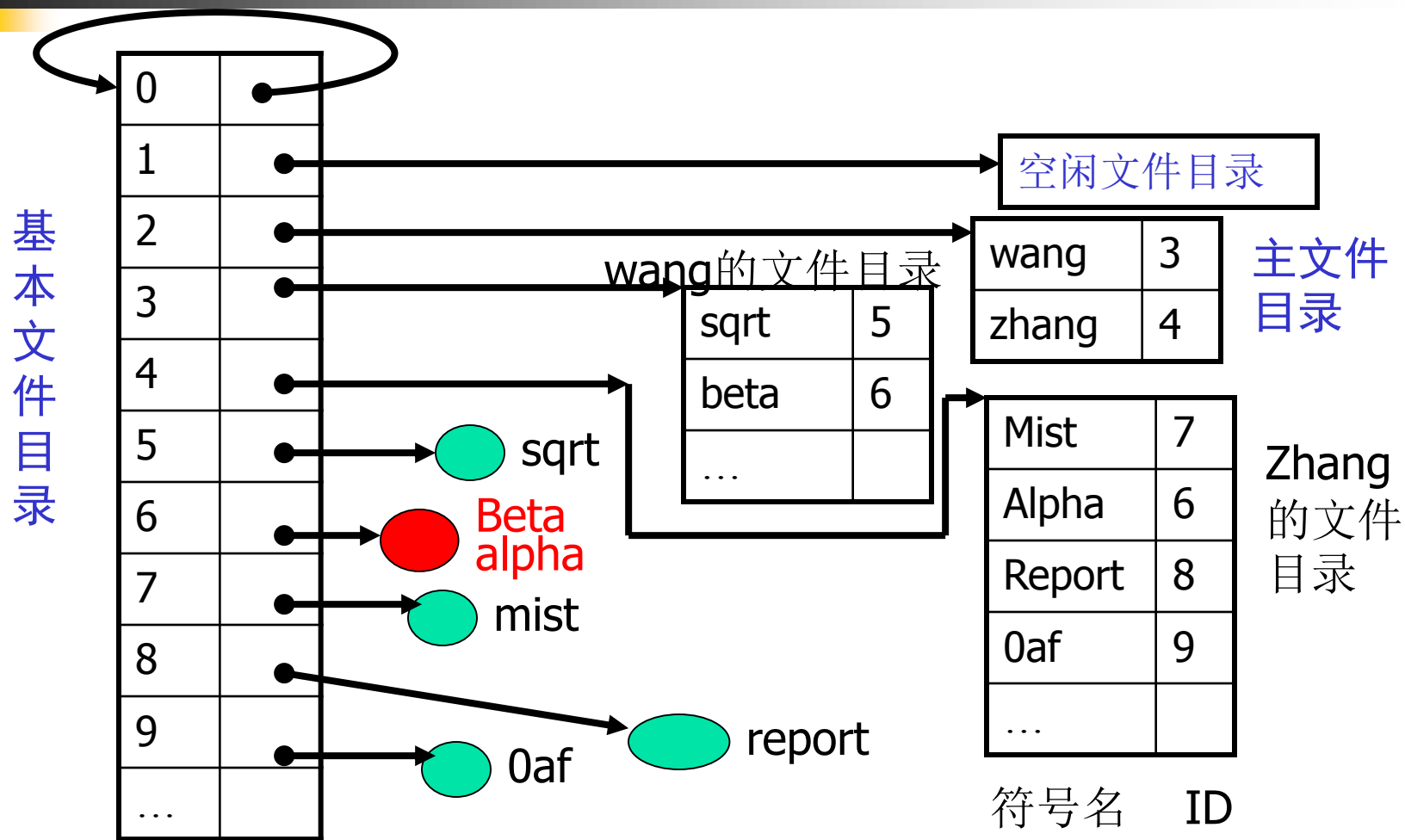
基于索引结点的共享方式

利用符号链实现文件共享

连访法



利用基本文件目录实现文件共享



基于索引结点的共享方式

wang的文件目录

| | |
|--------|---|
| | |
| Test r | ● |
| | |

Lee的文件目录

| | |
|--------|---|
| | |
| Test r | ● |
| | |

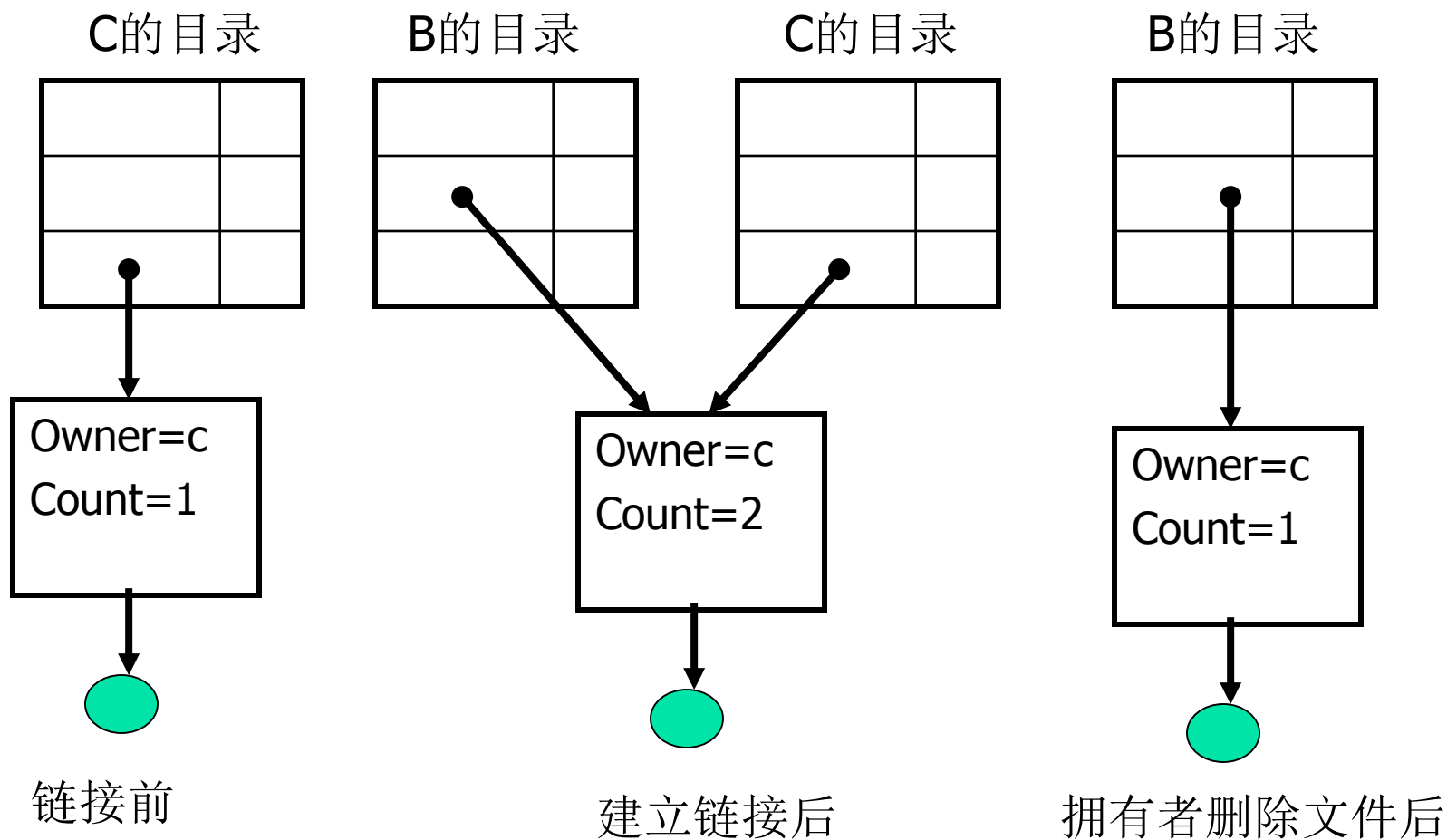
Count=2

文件物理地址

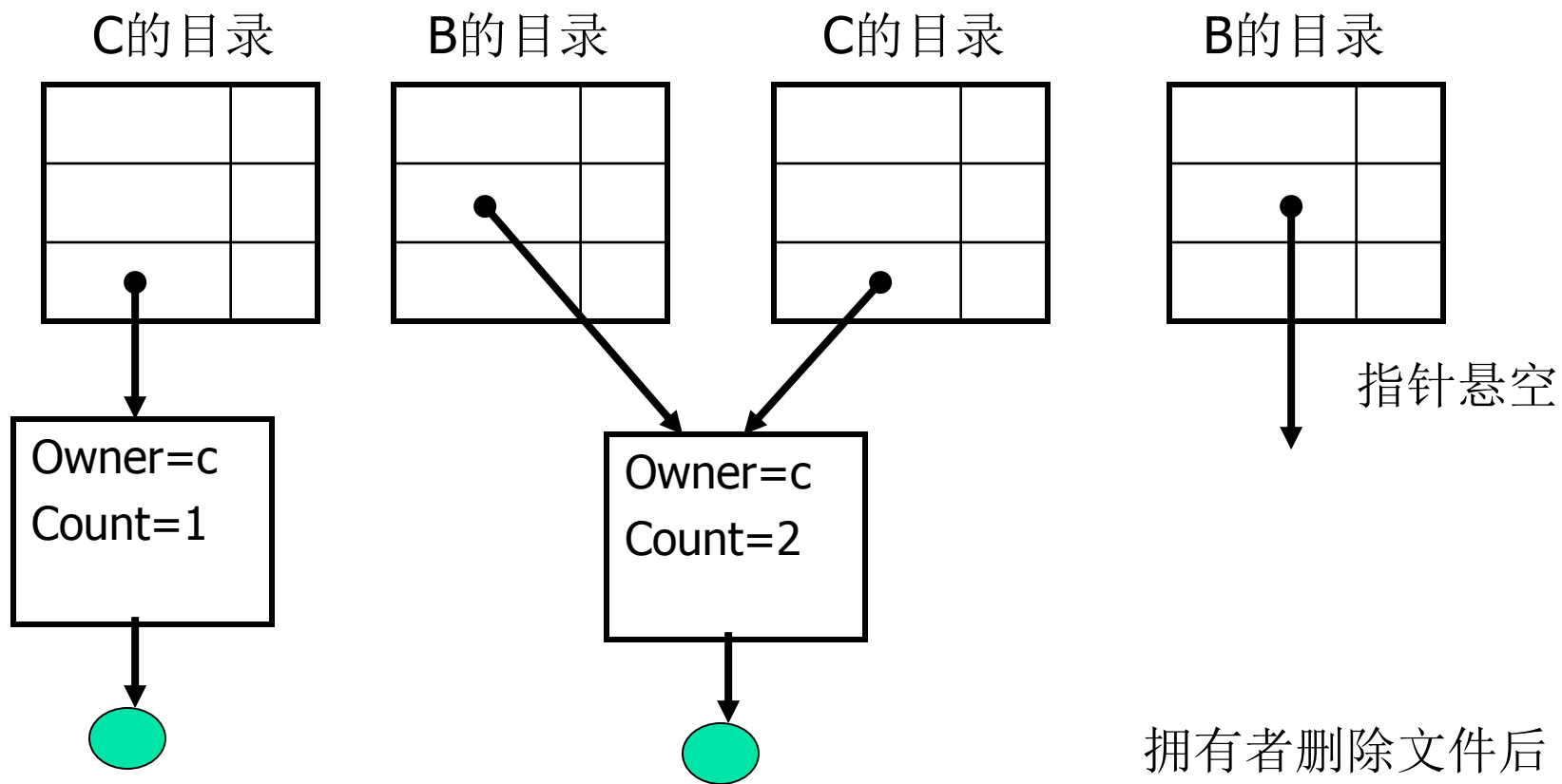
test

注：任何用户对文件进行附加操作或修改，所引起的相应索引结点内容的改变，对其它共享用户均是可见，从而可提供其他用户共享。

基于索引结点的共享方式

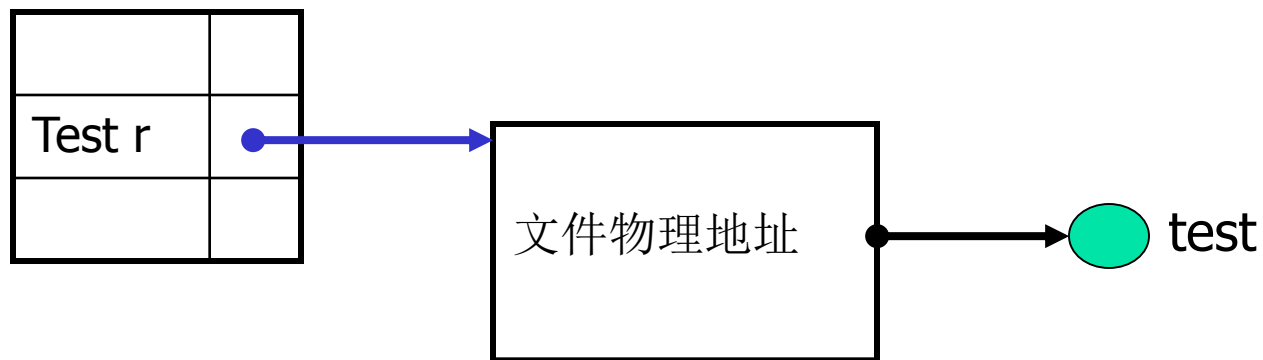


利用符号链实现文件共享



利用符号链实现文件共享

wang的文件目录



Lee的文件目录





利用符号链实现文件共享

- 优点

- 当主文件删除一共享文件时，其它共享文件的用户不会留下一个悬空指针。
- 可链接世界上任何地方的机器中的文件。

- 缺点

- 根据给定的文件路径名去查找目录，将使访问文件的开销大，启动磁盘频率较高。
- 每一共享文件将具有几个文件名，这将增加遍历共享文件的次数。



6.5 文件保护

 影响文件安全性的主要因素

 人为因素


 系统因素

 自然因素

 确保文件系统安全性的措施

 存取控制机制-----人为因素

 系统容错技术-----系统因素

 后备系统-----自然因素

6.5

文件保护——存取控制机制

 保护域

 访问矩阵

 访问矩阵的修改 (拷贝权、所有权、控制权)

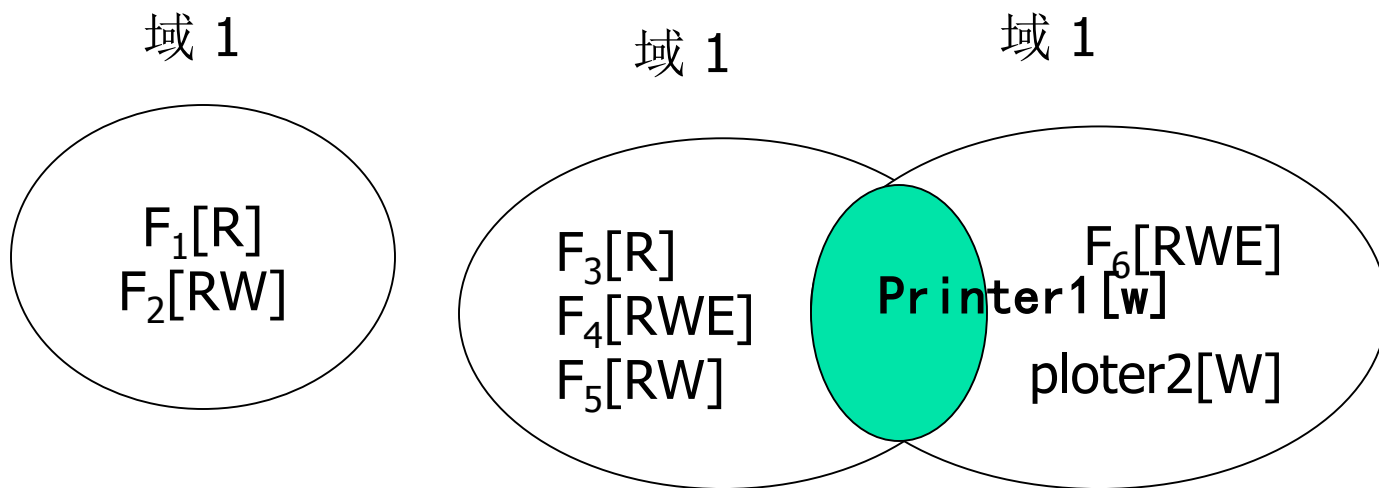
 访问矩阵的实现 (访问控制表、访问权限表)

 分级安全管理

- 系统级安全管理
- 用户级安全管理
- 目录级安全管理
- 文件级安全管理

 磁盘容错技术

保护域



■ 进程与域之间的联系

■ 静态联系

进程的可用资源集在进程的整个生命中是固定的

■ 动态联系

进程的可用资源集在进程的整个生命中是变化的

访问矩阵

| | 文件A | 文件B | 文件C | 打印机 |
|-----|------|------|-----|-----|
| 域D1 | R, W | R | W | |
| 域D2 | W, R | E | | W |
| 域D3 | | R, W | | W |

- 👉 用以描述系统存取控制的矩阵访问矩阵中的行代表域，列代表对象，矩阵中每一项由一组访问权组成。
- R---读；W---写；E----执行
 - 访问权 $\text{access}(I,j)$ 定义了域 D_i 中执行的进程能对对象施加的操作集。
 - 访问权通常由资源的拥有者或管理者所决定。



访问矩阵

- 进程拥有切换权时，可将进程从一个保护域切换到另一保护域，以实现进程和域之间的动态联系。

| | 文件A | 文件B | 文件C | 打印机 | 域D1 | 域D2 | 域D3 |
|-----|------|------|-----|-----|-----|-----|-----|
| 域D1 | R, W | R | W | | | S | |
| 域D2 | W, R | E | | W | | | S |
| 域D3 | | R, W | | W | | | |



访问矩阵

保护实现： 当进程向文件系统提出存取请求时，系统就根据存取控制矩阵将本次请求和该进程对文件的存取权限进行比较，若不匹配就拒绝执行。

特点： 对整个系统中所有文件的访问权限进行集中控制。

缺点： 当用户和文件较多时，存取控制矩阵就较大，占据的存储空间就较多； 查找花费时间较长。



访问矩阵的修改-拷贝权, 所有权, 控制权

进程在某个域中对某对象拥有的访问权可通过拷贝将访问权**扩展**到其它域（同一列即同一对象）中。

| | F1 | F2 | F3 |
|----|----|-----------|-----------|
| D1 | E | | W* |
| D2 | E | R* | E |
| D3 | E | | |

| | F1 | F2 | F3 |
|----|----|-----------|-----------|
| D1 | E | | W* |
| D2 | E | R* | E |
| D3 | E | R | W |

访问矩阵的修改-拷贝权, 所有权, 控制权

利用所有权 (O) 可实现访问权的扩展、增加和删除 (同一列即同一对象)。

| | F1 | F2 | F3 |
|----|-----|------|--------|
| D1 | O,E | | W |
| D2 | | R*,O | R*,O,W |
| D3 | E | | |

| | F1 | F2 | F3 |
|----|--------------|---------|--------------|
| D1 | O,E | | W |
| D2 | | R*,O,W* | R*,O,W |
| D3 | E | W | W |

访问矩阵的修改-拷贝权, 所有权, 控制权

可改变矩阵中同一行（同一域即不同对象）的访问权。

| | F1 | F2 | F3 | F4 | F5 | 打印机1 | 绘图仪2 | D1 | D2 | D3 |
|----|----|-----|-----|-----|-------|------|------|----|----|---------|
| D1 | R | R,W | | | | | | | | |
| D2 | | | R,E | R,W | | W | | | | control |
| D3 | | | | | R,W,E | W | W | | | |





访问矩阵的实现--访问控制表

- 将访问矩阵按列（对象）划分，为每一列建立一张访问控制表ACL。
- 在该表中无原矩阵中的空项。
- 由有序对（域，权集）组成。
- 对象为文件时，常将ACL存放于该文件的FCB/索引结点中，作为存取控制信息。
- 可定义缺省的访问权集。



访问矩阵的实现--访问权限表


👉 将访问矩阵按行（域）划分，形成一行一张访问权限表。

| 类型 | 权力 | 对象 |
|-----|------------|-----------|
| 文件 | R-- | 指向文件3的指针 |
| 文件 | RWE | 指向文件4的指针 |
| 文件 | RW- | 指向文件5的指针 |
| 打印机 | -W- | 指向打印机1的指针 |




分级安全管理-

系统级安全管理


 主要任务

不允许未经核准的用户进入系统

 主要方法

 注册

 登录

 其它措施

■ 系统级安全管理

■ 用户级安全管理

■ 目录级安全管理

■ 文件级安全管理

注册

注册用户表

| |
|-----|
| 用户1 |
| 用户2 |
| 用户3 |
| |

用户2表

| |
|------|
| 用户名 |
| 口令 |
| 上机时间 |
| 终端号 |

- 主要目的

使系统管理员能够掌握要使用系统的诸用户的情况，并保证用户名在系统中的唯一性。

- 实现

- 系统中保存一张用户表。
- 用户使用前须占用用户表一空项注册。
- 用户使用后须由管理员将该用户从用户表中删除。
- 一旦删除，该用户便不能再进入系统，除非再次登录。
- 用户表中用户有限。



登录

注册用户表

| |
|-----|
| 用户1 |
| 用户2 |
| 用户3 |
| |

用户2表

| |
|------|
| 用户名 |
| 口令 |
| 上机时间 |
| 终端号 |

- 主要目的

通过核实该用户的注册名及口令来检查该用户使用系统的合法性。

- 实现

- 输入注册用户名。
- 系统将注册用户名与用户表中的注册名比较。
- 若用户名正确，则输入用户口令。
- 口令正确，则登录成功。
- 用户可使用系统。



其它措施

注册用户表

| |
|-----|
| 用户1 |
| 用户2 |
| 用户3 |
| |

用户2表

| |
|------|
| 用户名 |
| 口令 |
| 上机时间 |
| 终端号 |

- 规定用户定期修改口令
- 限定用户的终端
- 限定用户的上机时间



分级安全管理-

- 系统级安全管理
- **用户级安全管理**
- 目录级安全管理
- 文件级安全管理

用户级安全管理

主要任务

根据用户性质、需求及文件属性给用户分配“文件访问权”。

主要方法

用户分类

- 文件主
- 伙伴
- 一般用户（超级用户、系统操作员、用户、顾客）

文件访问权（建立、删除、打开、读、写、查询、修改、父权）



分级安全管理-

- 系统级安全管理

- 用户级安全管理

- **目录级安全管理**

- 文件级安全管理

目录级安全管理

 主要任务

为保护系统中各种目录的安全，系统对目录操作指定了权限。

 实现方法

- 用户访问一目录时，核心将比较用户访问权和目录中的访问权。
- 若二者中均有该访问权，用户方能获得有效访问权（两权限的交集）。
- 否则，拒绝用户访问。

分级安全管理-

- 系统级安全管理
- 用户级安全管理
- 目录级安全管理

文件级安全管理

主要任务

通过系统管理员或文件主对文件属性的设置，来控制用户对文件的访问。

文件属性

- 只执行 (EO)
- 隐含 (H)
- 索引 (I)
- 修改 (M)
- 只读 (RO)
- 读、写 (RW)
- 共享 (SHA)
- 系统 (SY)

■ 文件级安全管理

□ 实现方法

用户对文件的访问由有效权限和文件属性的交集决定



磁盘容错技术

容错技术是通过在系统中设置冗余部件的办法，来提高系统可靠性的一种技术。

- **磁盘容错技术**：是通过增加冗余的磁盘驱动器、磁盘控制器等方法，来提高磁盘系统可靠性的一种技术。磁盘容错技术又称为系统容错技术SFT (System Fault Tolerance) ,可分为三级：
 - 低级磁盘容错技术SFT-I
 - 中级磁盘容错技术SFT-II
 - 高级磁盘容错技术SFT-III



低级磁盘容错技术SFT-I

- **低级磁盘容错技术SFT-I**

防止因磁盘表面发生缺陷所引起的数据丢失。采用的措施有：

- 双份目录和双份文件分配表
- 热修复重定向和写后读校验

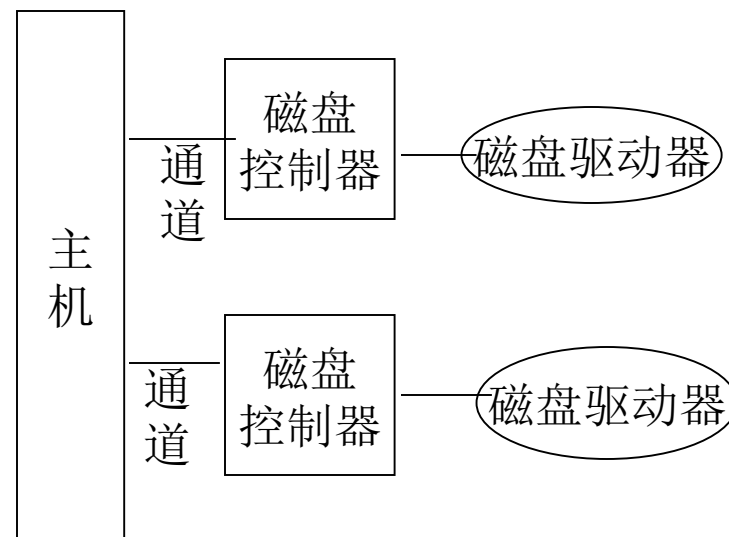
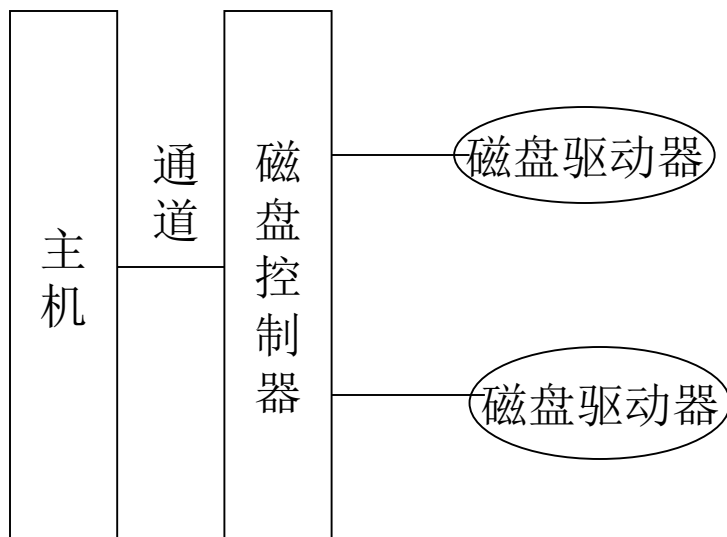
中级磁盘容错技术SFT-II

■ 中级磁盘容错技术SFT-II

防止因磁盘发生故障而引起的数据丢失。采用的措施有：

■ 磁盘镜像

磁盘双工





数据一致性控制

问题：

在数据应用中，当把一个数据分别存储到多个文件中时，可能会出现同一数据不一致性。

问题解决：

配置能保证数据一致性的软件及高度可靠的硬盘（称为稳定存储器），采用冗余技术(磁盘双工方式)可实现。

保证数据一致性的软件手段：

- 事务
- 重要数据结构的一致性检查



事务

当一个数据被分散地存放在一个文件的不同记录或多个文件中时，可采用事务来保证该数据的一致性。

- **事务**

用于访问和修改各种数据项的一程序单位，可被看成是**一系列的读和写操作**。

- 事务操作具有**原子操作**，即对分布在不同位置的同一数据进行读和写操作全部完成后，才能终止事务。若一个失败，则夭折事务，已修改的部分也全部恢复原来的情况。
- 事务操作的原子性借助于存放在稳定存储器中的**事务记录**来实现（每条记录描述了事务运行中的重要操作，如修改操作、开始操作、托付事务和夭折事务等）



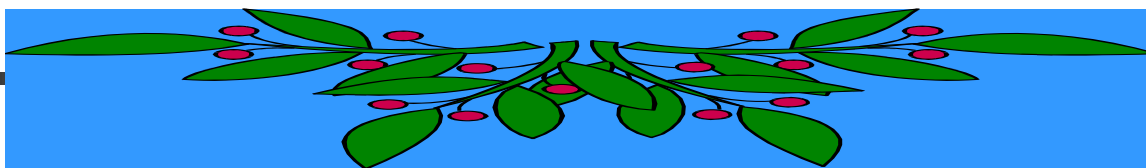
重要数据结构的一致性检查

如果在修改过的磁盘块全部写回之前系统发生故障，则文件系统有可能处于不一致状态。

当未被写回的内容涉及到索引结点、目录或空闲盘块表等重要数据结构时，该问题会更严重。所以往往在崩溃后重新启动时，运行一实用程序进行一致性的检查。

- **盘块号的一致性检查**
- **链接计数的一致性检查**

本章小结



- 1. 文件、文件系统（理解）
- 2. 逻辑结构、物理结构（重点掌握）
- 3. 目录（重点）
- 4. 存储保护（理解）
- 5. 数据一致性（理解）



本章作业

- 1、设某系统磁盘共有**1600**块，块号从**0-1599**，若用位示图管理这**1600**块磁盘空间，问位示图需要多少个字节？
- 2、假定盘块的大小为**1KB**，硬盘的大小为**500MB**，采用显式链接分配方式时，其**FAT**需占用多少存储空间？
- 3、使用文件系统时，为什么通常要显式地进行**OPEN**、**CLOSE**操作？