

操作系统原理

黄俊杰

2024年9月



第8章 文件系统



大量的程序、数据
保存在外存！
文件？

由谁管理？
用户？
OS？

文件系统



第8章 文件管理



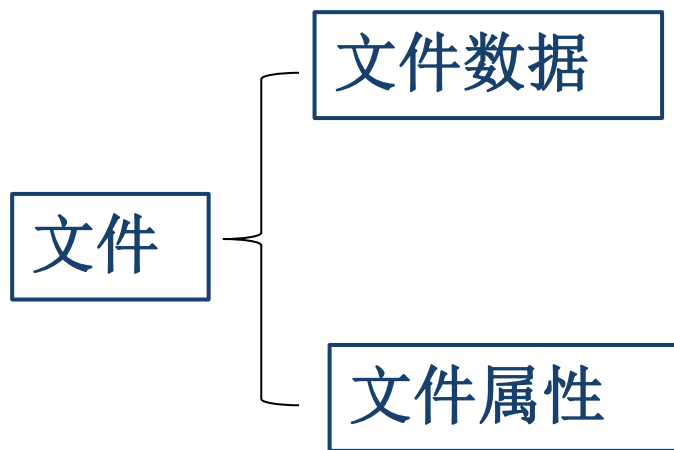
- 8.1 文件与文件系统
- 8.2 文件目录
- 8.3 文件系统组织与实现
 - 文件的逻辑结构
 - 文件的物理结构
 - 外存空间的管理
- 8.4 文件的可靠性与安全
- 8.5 文件系统的层次模型



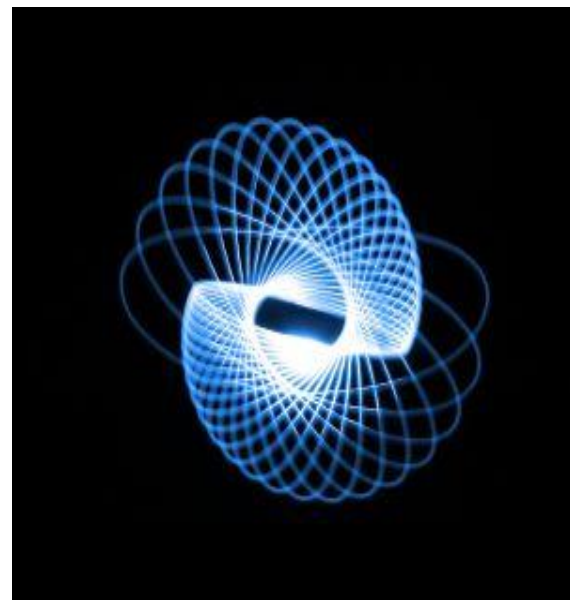
8.1 文件与文件系统

1. 文件的定义

文件是具有标识符（文件名）的一组相关信息集合。

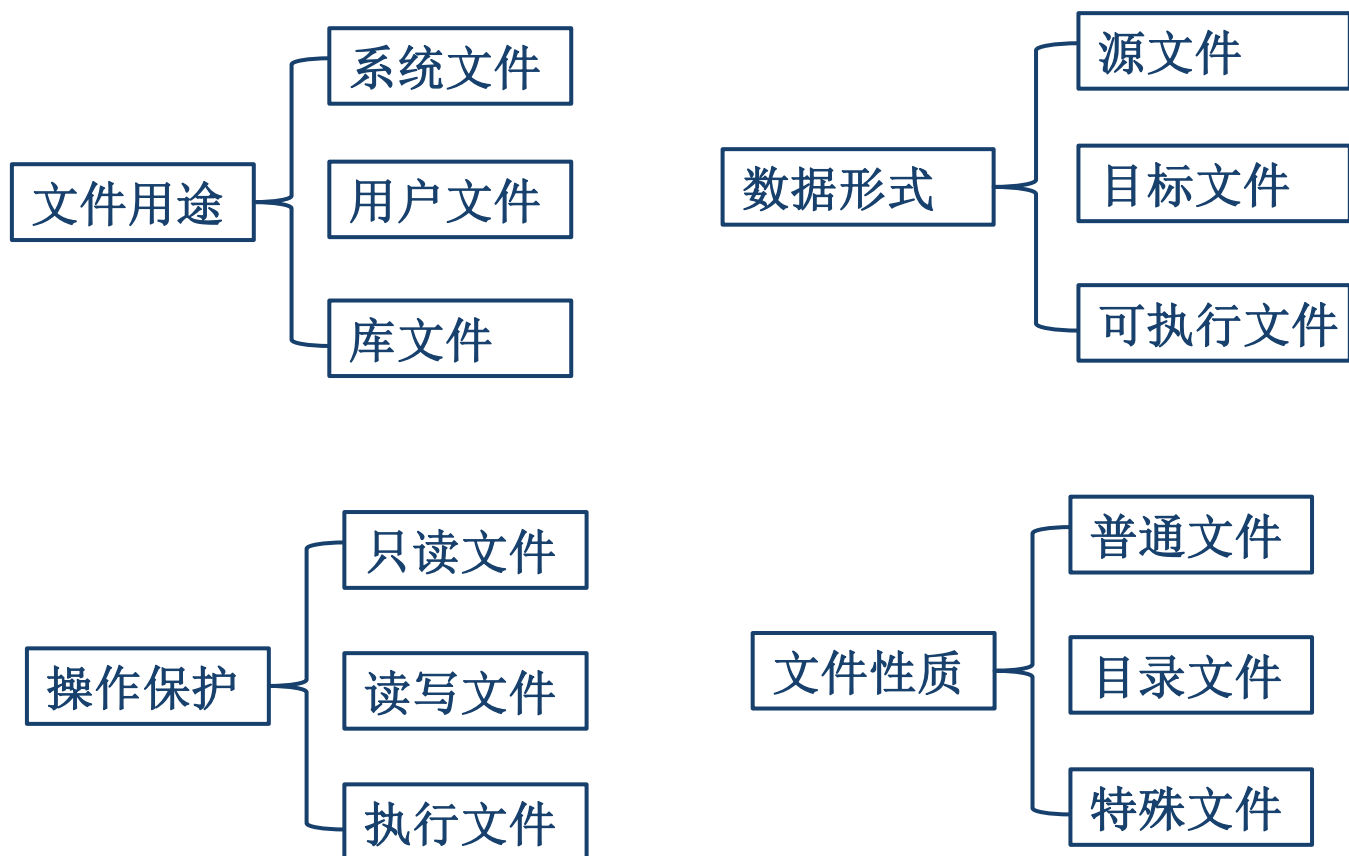


- 1) 文件类型
- 2) 文件长度
- 3) 文件的位置
- 4) 文件的存取控制
- 5) 文件的建立时间



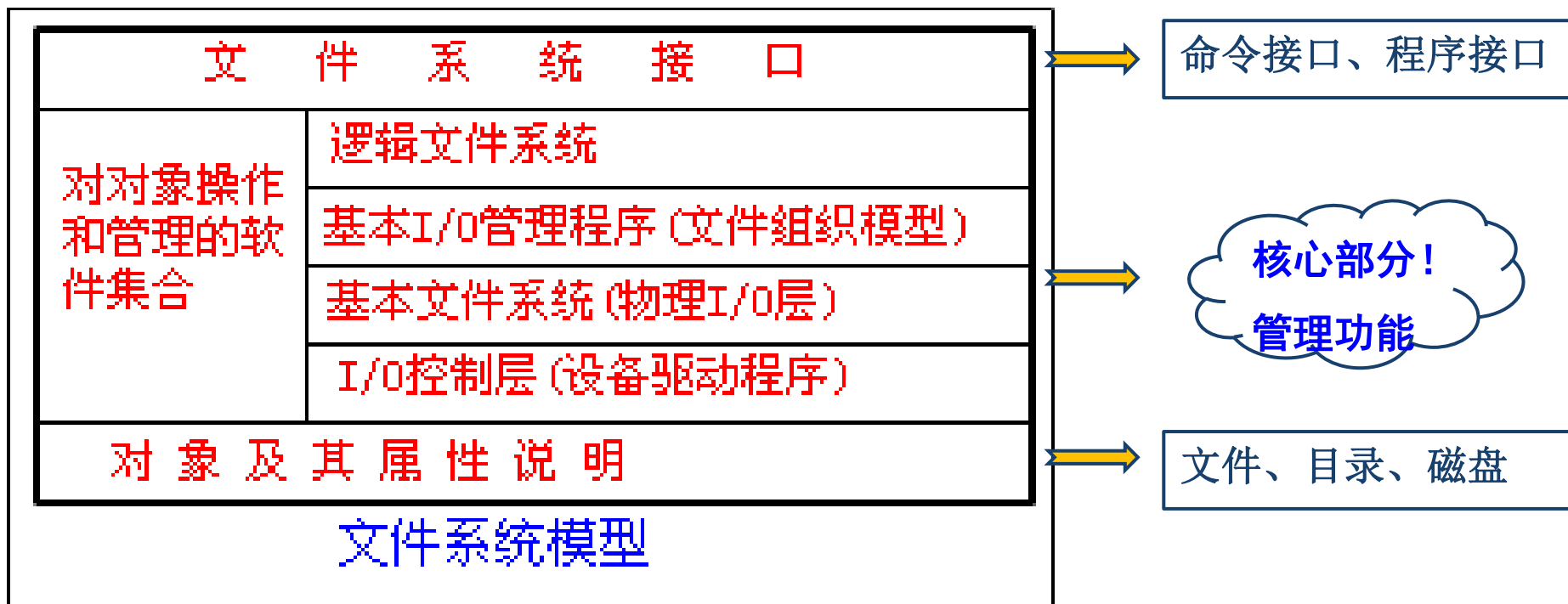
8.1 文件与文件系统

2. 文件的分类



8.1 文件与文件系统

• 3. 文件系统的定义



定义：大量的文件及其属性信息，负责对文件进行操纵和管理，并向用户提供一个使用文件的接口。



8.1 文件与文件系统

4、文件的操作(文件接口)

(1) 创建文件

(2) 删除文件

(3) 打开文件

(4) 读文件

(5) 写文件

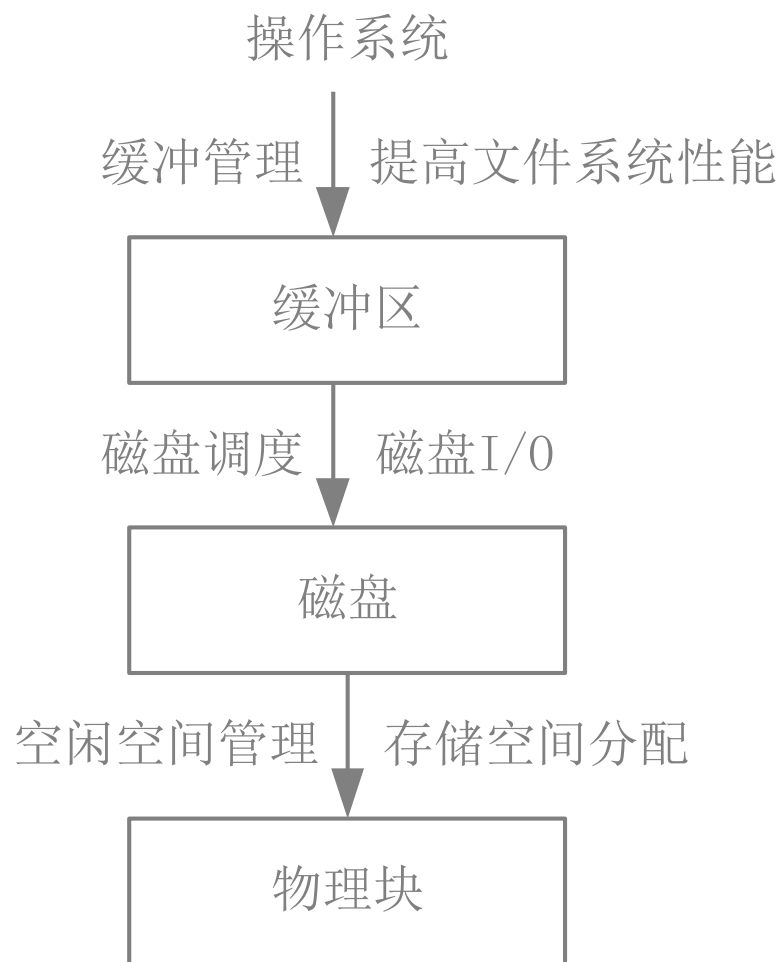
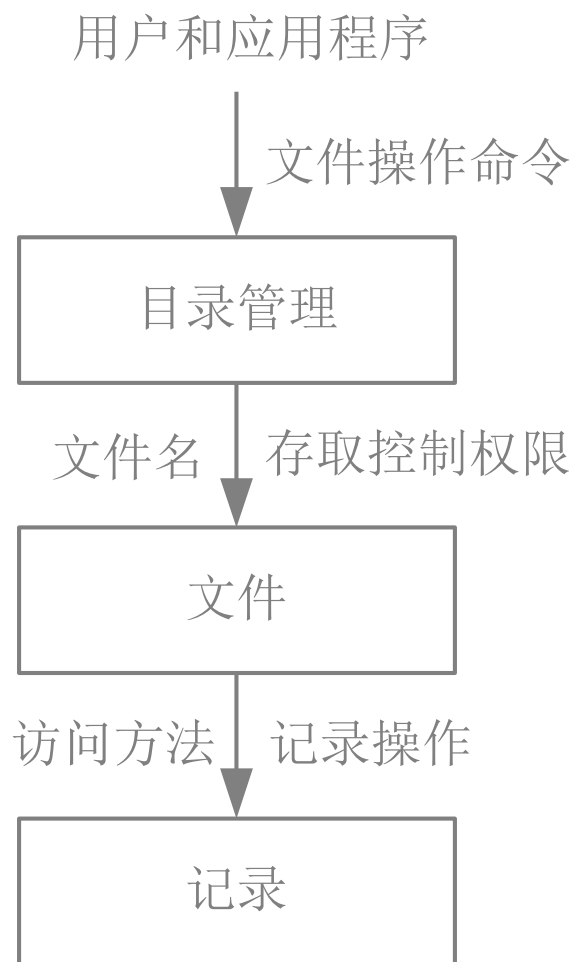
(6) 关闭文件

.....

- 所谓“打开”，是指系统将该文件的属性从外存拷贝到内存打开文件表的一个表目中，并将该表目的编号返回给用户。
- 关闭文件，则相反。
- 优点：节省检索开销，提高操作速度



文件管理功能



8.2 文件目录

大量的文件如何组织？

目录也是以文件形式存在！

文件名	扩展名	文件属性	建立日期	建立时间	文件长度	修改日期	修改时间	第一个磁盘块号



8.2 文件目录

- 在一个计算机系统中保存有许多文件，用户在创建和使用文件时只给出文件的名字，由文件系统根据文件名找到指定文件。
- 为了便于对文件进行管理，设计了文件目录，用于检索系统中的所有文件。
- 文件系统的一个最大特点是“按名存取”，用户只要给出文件的符号名就能方便地存取在外存空间的文件信息，而不必关心文件的具体物理地址。
- 提高检索速度，允许文件同名和支持文件共享等功能。



8.2 文件目录

- 而实现文件符号名到文件物理地址映射的主要环节是检索文件目录。
- 系统为每个文件设置一个描述性数据结构——文件控制块FCB（File Control Block）
 - 文件的基本信息
 - 文件存取控制
 - 文件的使用信息
- 文件目录就是文件控制块的有序集合。



文件控制块FCB

- 文件控制块FCB是系统为管理文件而设置的一个数据结构。
- FCB是文件存在的标志，它记录了系统管理文件所需要的全部信息。
 - 文件名、文件号、用户名；
 - 文件的物理位置、文件长度；
 - 记录大小、文件类型、文件属性；
 - 共享说明；
 - 文件逻辑结构、文件物理结构；
 - 建立文件的日期和时间、最后访问日期和时间、最后修改日期和时间；
 - 口令；
 - 保存期限等。



文件目录与目录文件

1. 文件目录

- 文件与文件控制块是**一一对应**的。
- 文件控制块的有序集合构成文件目录，每个**目录项**即是一个**文件控制块**。
- 给定一个文件名，通过查找**文件目录**便可找到该文件对应的目录项（即FCB）。

2. 目录文件

- 文件目录是需要**长期保存**的，
- 为了实现文件目录的管理，通常将文件目录以文件的形式保存在外存空间，这个文件就被称为**目录文件**。
- 目录文件是**长度固定**的**记录式**文件。



文件目录结构

- 文件目录的组织与管理是文件管理中的一个重要方面
- 一般有一级目录结构、二级目录结构和多级目录结构

1. 一级目录结构

整个系统设置一张线性目录表，表中包括了所有文件的文件控制块，每个文件控制块都指向一个普通文件，如下图所示。



文件目录结构

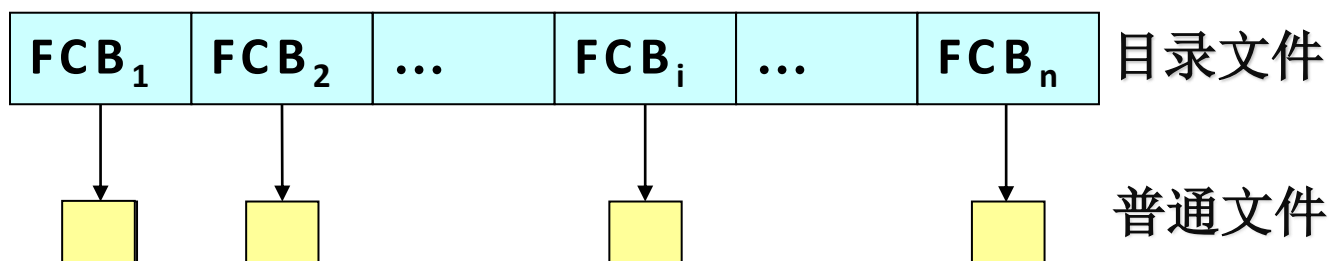


图 一级目录结构

- 一级目录结构的优点是**简单**；
- 缺点是文件**不能重名**，限制了用户对文件的命名。这是不方便的，因为不同用户所建立的文件可能具有相同的名字。

改进→



文件目录结构

2. 二级目录

- 把文件目录分成两级，第一级称为**主文件目录（MFD）**，第二级称为**用户文件目录（UFD）**。
- 每个用户在主文件目录中都有一个**登记项**，记录了用户和该用户的用户文件目录的物理地址。
- 而在用户文件目录中，**存放该用户每一个文件的文件控制块**。
- 其结构如下图所示：



文件目录结构

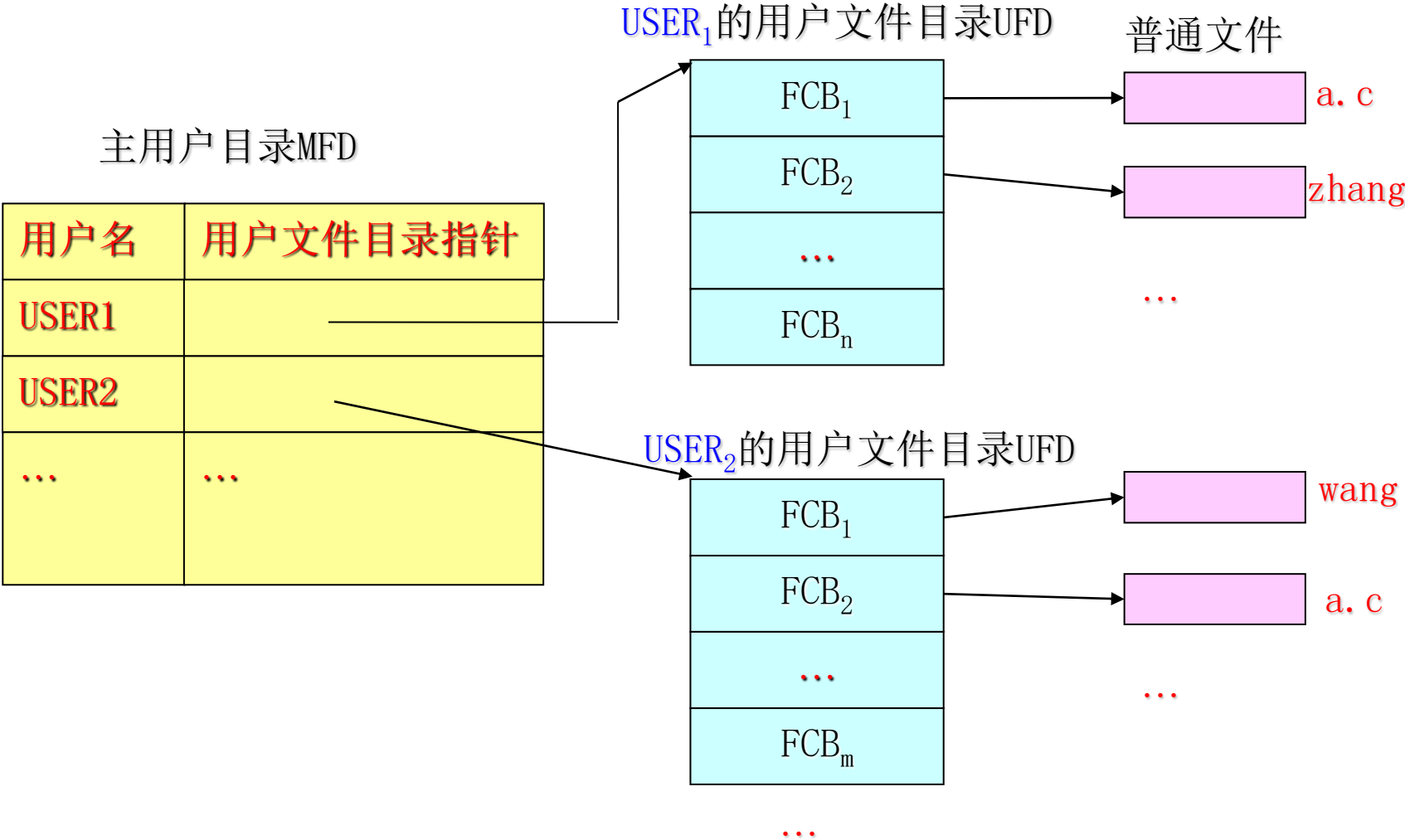


图 二级目录结构



文件目录结构

- 二级目录结构实现了文件从名字空间到外存地址空间的映射：
用户名→文件名→文件内容。
- 其优点是：
 - 有利于文件的管理、共享和保护；
 - 适用于多用户系统；
 - 不同的用户可以命名相同文件名的文件，不会产生混淆，解决了命名冲突问题。
- 其缺点是**不能对文件分类**，当用户文件较多时查找速度慢。

改进→



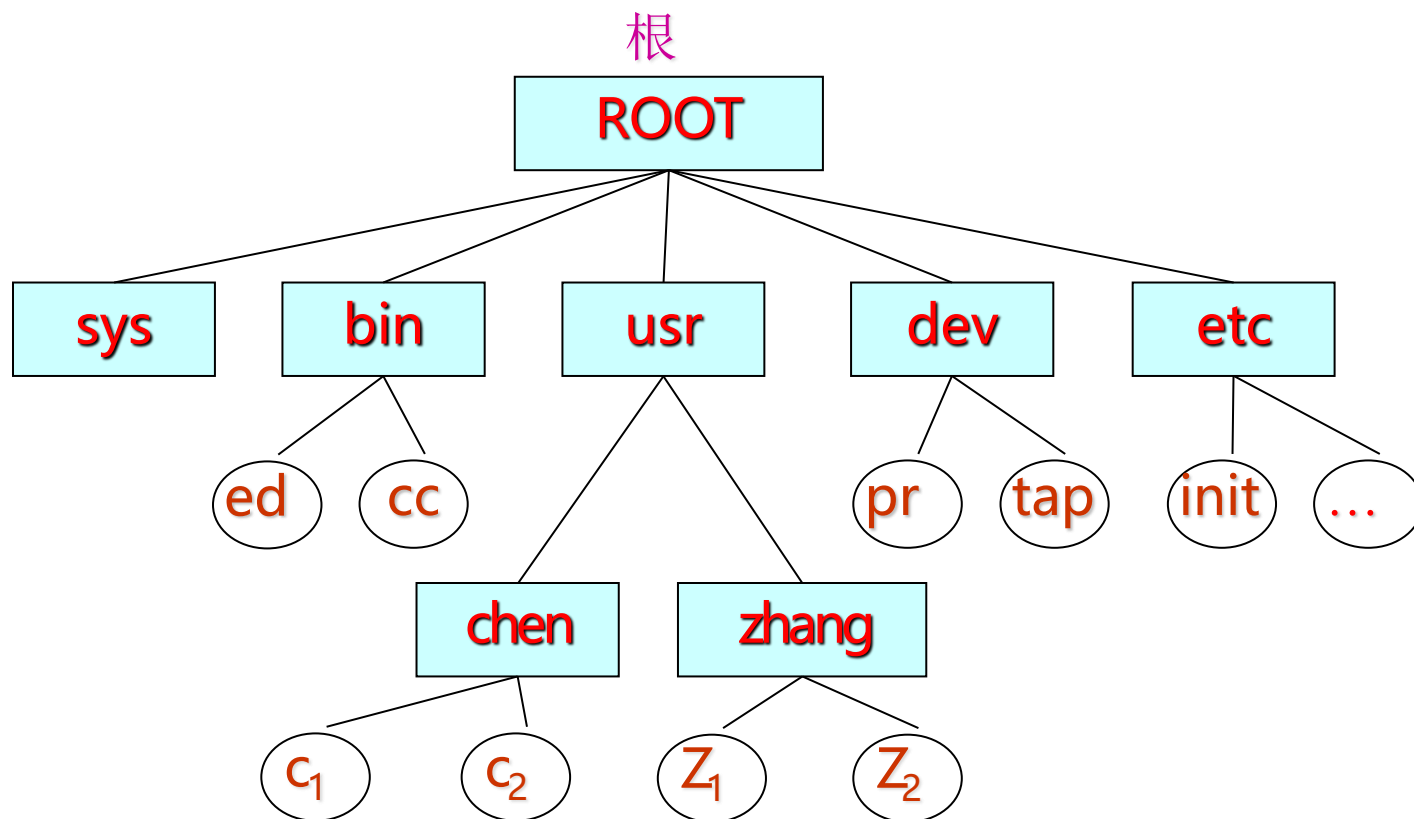
文件目录结构

3. 多级目录结构

多级目录结构是对二级目录结构的进一步改进，又称树型目录结构，如下图所示。



文件目录结构



其中：

- 树叶结点表示普通文件（用圆圈表示）
- 非叶结点表示目录文件（用矩形表示）



文件目录结构

- 树根结点称为根目录，根目录是唯一的，由它开始可以查找所有其他目录文件和普通文件，一般可放在内存。
- 从根结点出发到任一非叶结点或树叶结点都有且仅有一条路径，该路径上的全部分支组成了一个全路径名。
- 采用多级目录结构时，文件名为一个路径名。
- 多级目录结构的优点是便于文件分类，可为每类文件建立一个子目录；查找速度快，因为每个目录下的文件数目较少；可以实现文件共享。
- 多级目录结构的缺点是相对来说比较复杂。

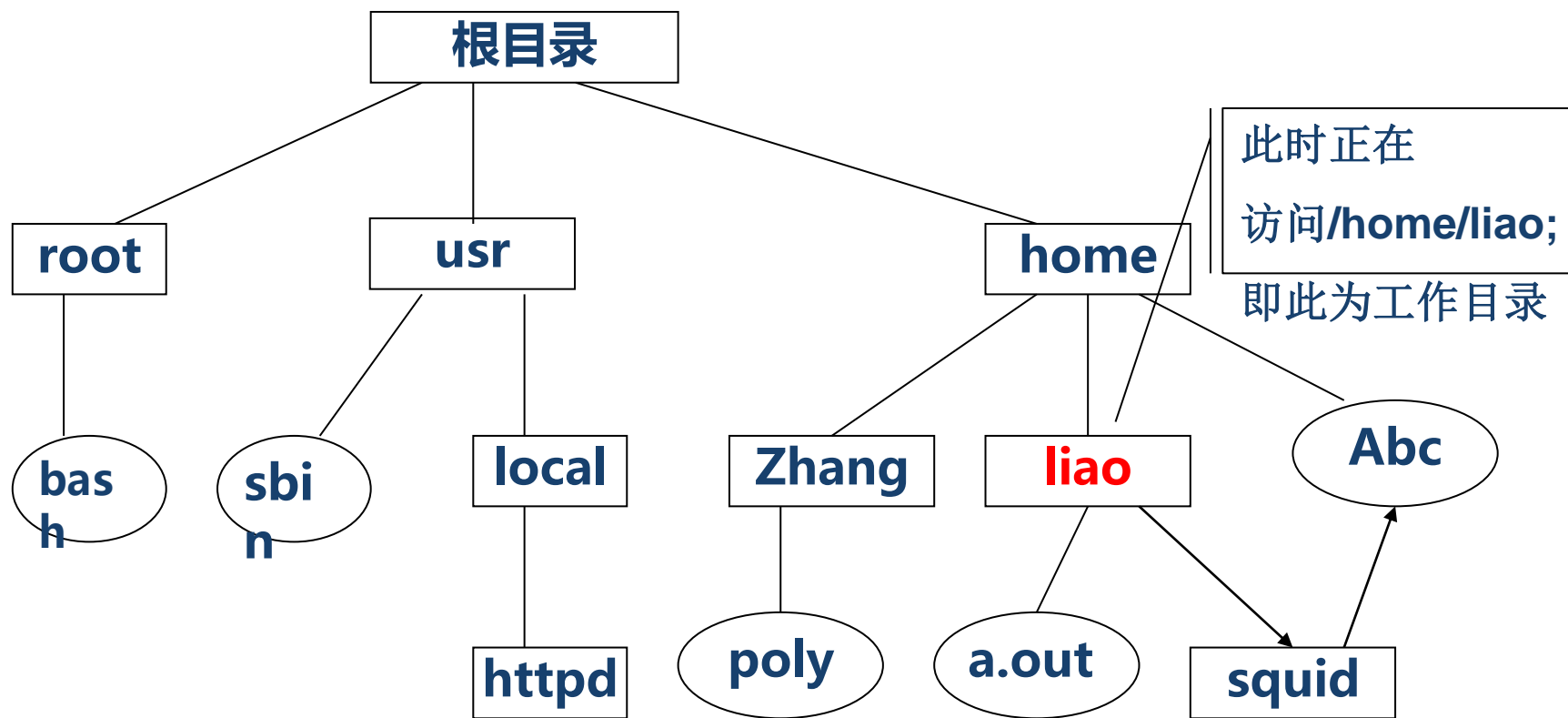


当前目录

- 在一个多层次的树形文件目录结构中，如果每次都从根结点开始检索，很不方便，通常各目录文件放在外存，故影响访问速度，尤其是当层次较多时检索要耗费很多时间。
- 系统提供一个目前正在使用的工作目录，称为当前目录。
- 查找文件时既可以从根目录开始，也可以从当前目录开始向下检索。
- 若从当前目录开始，路径名只要给出从当前目录开始到所要访问文件的相对路径名即可。这样检索路径缩短，检索速度提高。



8.2 文件目录



a.out的绝对路径: /home/liao/a.out

a.out的相对路径: ./a.out

因此, 减少了
搜索范围!



8.2 文件目录

- 文件目录比较大，因此它一般会放在磁盘上。查找的时候通过要把存放文件目录的磁盘块（**存放着FCBs**）一个一个的读到内存中进行查找，对于文件数目比较多的时候，需要启动N多次磁盘



目录查询技术

如基于索引节点：
查找/usr/ast/mbox



8.2 文件目录

/usr/ast/mbox

根目录表

文件名	索引节点
.	1
..	1
bin	4
dev	7
lib	14
etc	9
usr	6
tmp	8

索引节点表 (外存)

索引节点	文件类型	属性	物理地址
1	d	...	
...	
6	d	...	132
...
26	d		496
...
60	f	...	200
...

(1) 在根目录表中查找usr目录

(2) 读入6号索引节点到内存



8.2 文件目录

/usr/ast/mbox

usr目录文件

文件名	索引节点
.	6
..	1
are	19
jkl	30
hui	51
ast	26
lkm	45

索引节点表

索引节点	文件类型	属性	物理地址
1	d	...	
...	
6	d	...	132
...
26	d	...	496
...
60	f	...	200
...

3) 从132号盘块读入usr目录文件，查找ast

(4) 读入26号索引节点到内存



8.2 文件目录

/usr/ast/mbox

ast目录文件

文件名	索引节点
.	26
..	6
gran	64
book	92
mbo x	60
mini	81
scr	17

索引节点表

索引节点	文件类型	属性	物理地址
1	d
...
6	d	...	132
...
26	d	...	496
...
60	f	...	200
...

(5) 从496号盘块读入ast目录文件，查找mbox

(6) 读入60号索引节点到内存

(7) 从200号盘块读入mbox文件，查找结束



8.3 文件系统的组织与实现

用户关心：

文件内容（或记录）



文件逻辑结构

1. 有结构的文件 (记录型文件)

定长记录型

变长记录型

2. 无结构文件 (流式文件)

系统关心：文件
存储、提取



文件物理结构

1. 连续文件

2. 链接文件

3. 索引文件

同样是系统关心：
提高外存空间的
利用率



外在空间管理

1. 空闲表

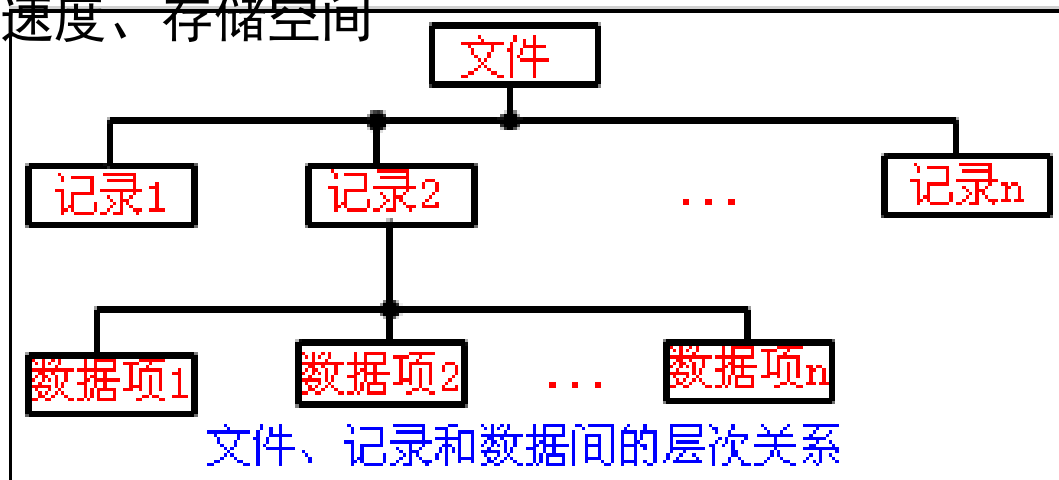
2. 位示图

3. 组块链接



8.3.1 文件的逻辑结构

- 需要考虑：速度、存储空间



记录型文件的组织方式

1、顺序文件

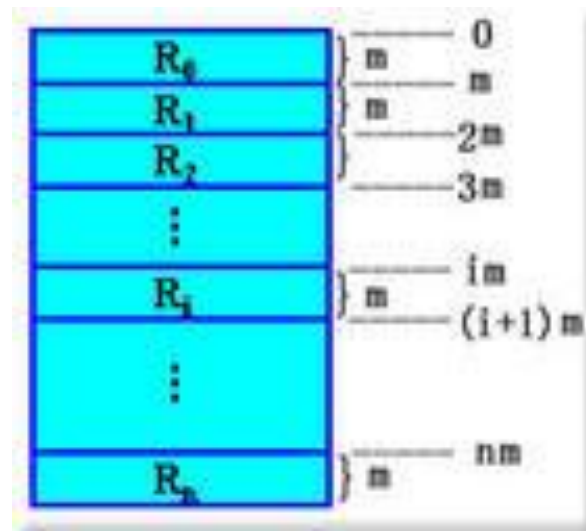
2、索引文件

3、索引顺序文件



8.3.1 文件的逻辑结构

- 1、顺序文件
 - 1) 支持读/写操作
 - 2) 优点：批量存取效率相当高
 - 3) 缺点：查找、修改和插入单个记录是效率低



定长记录文件

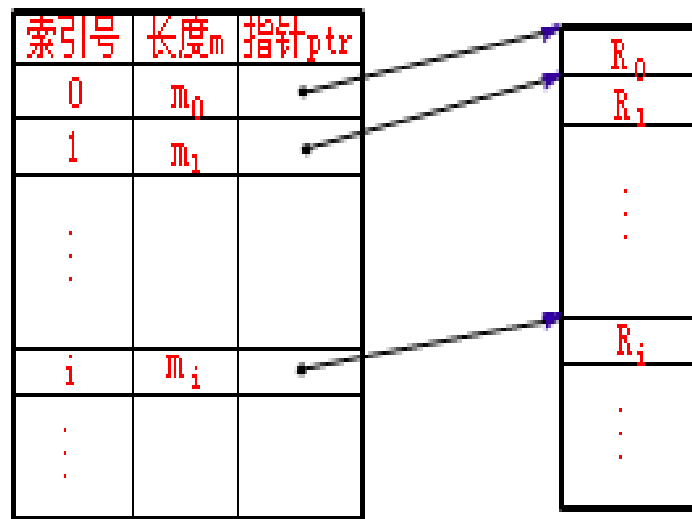


8.3.1 文件的逻辑结构

2、索引文件

- 优：随机访问时速度快
- 缺：额外存储空间存放

索引表



索引表

索引文件的组织

逻辑文件



8.3.1 文件的逻辑结构

3、索引顺序文件

1) 读/写操作

2) 优点：折中

键	逻辑地址	姓名	其它属性
An Qi		An Qi	
Bao Rong		An Kang	
Chen Lin			
		Bao Rong	

索引顺序文件



8.3.1 文件的逻辑结构

■ 不同逻辑结构的检索效率比较:

记录个数	顺序文件	索引顺序文件
N	$N/2$	\sqrt{N}
10,000	5,000	100
10^6	500,000	1000

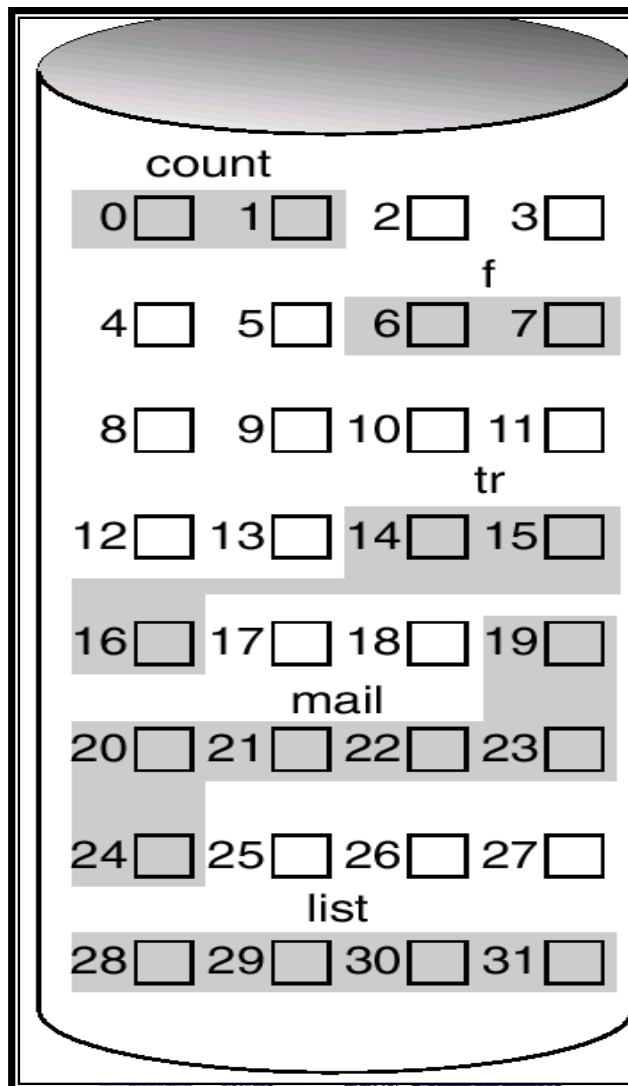


8.3.2 文件的物理结构（外存分配方式）

- 连续结构
- 链接结构
- 索引结构
- 考虑：速度、存储空间



连续结构



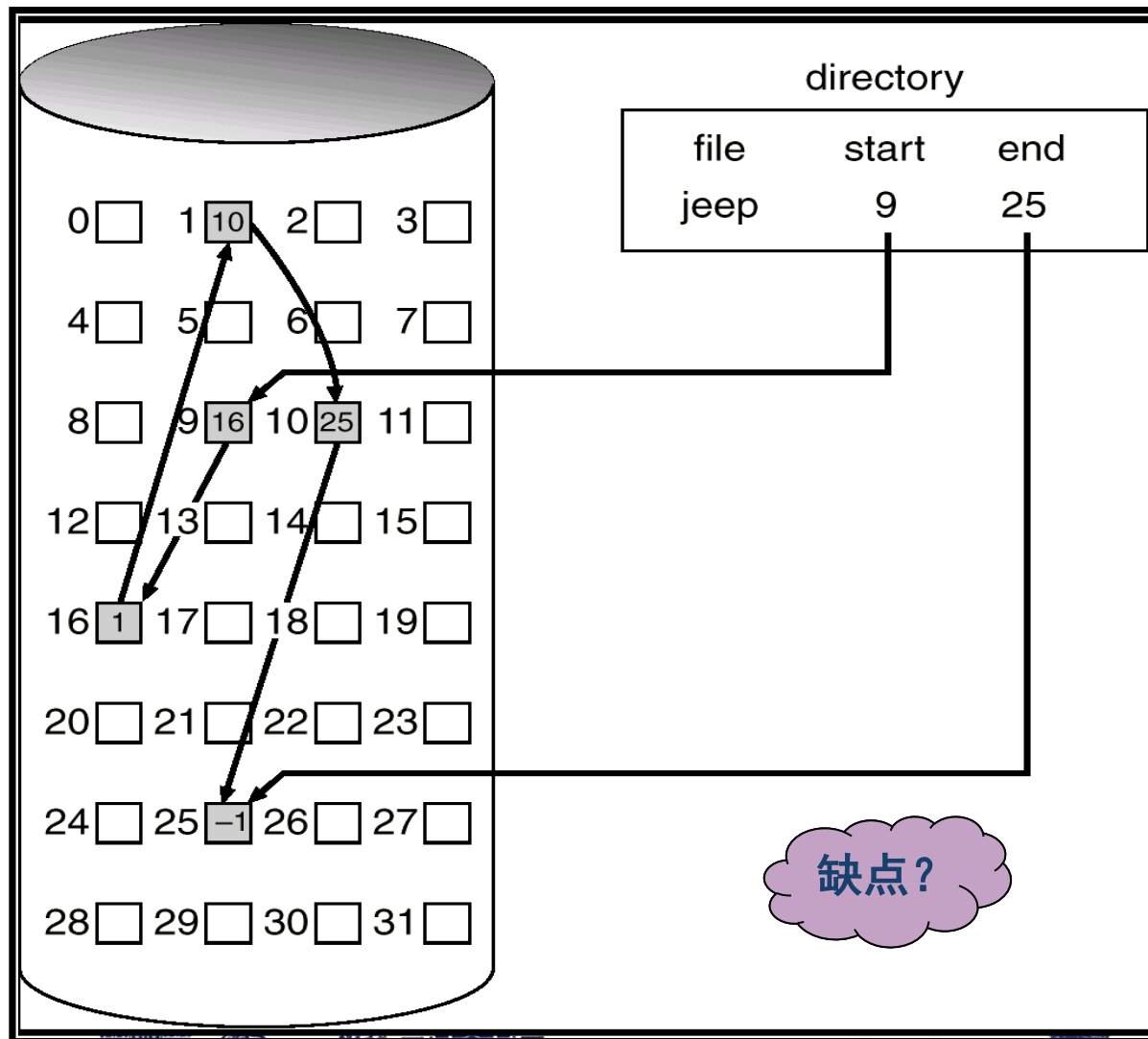
directory

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

缺点：要求有连续
存储空间且不得于
文件动态增长

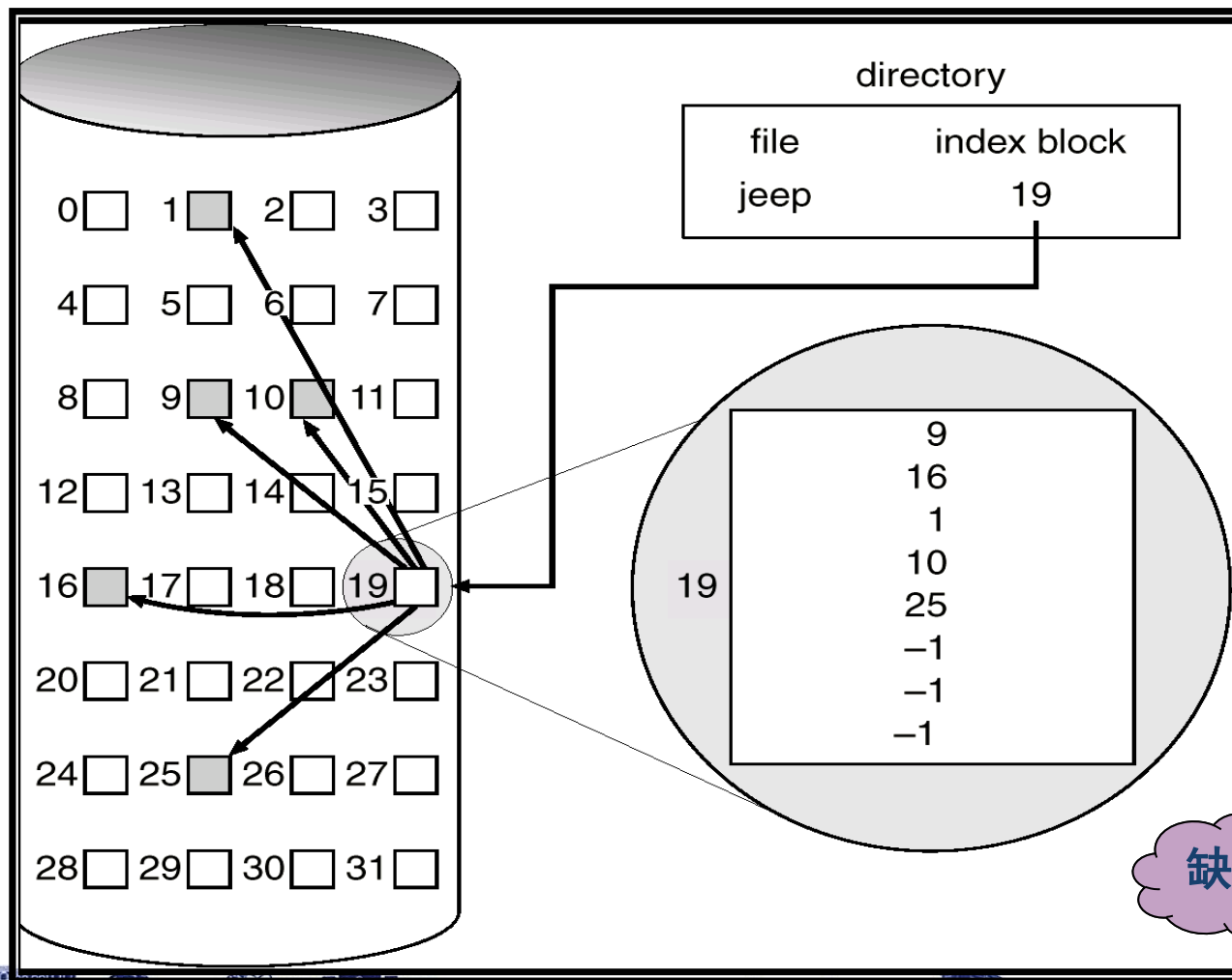


链接结构



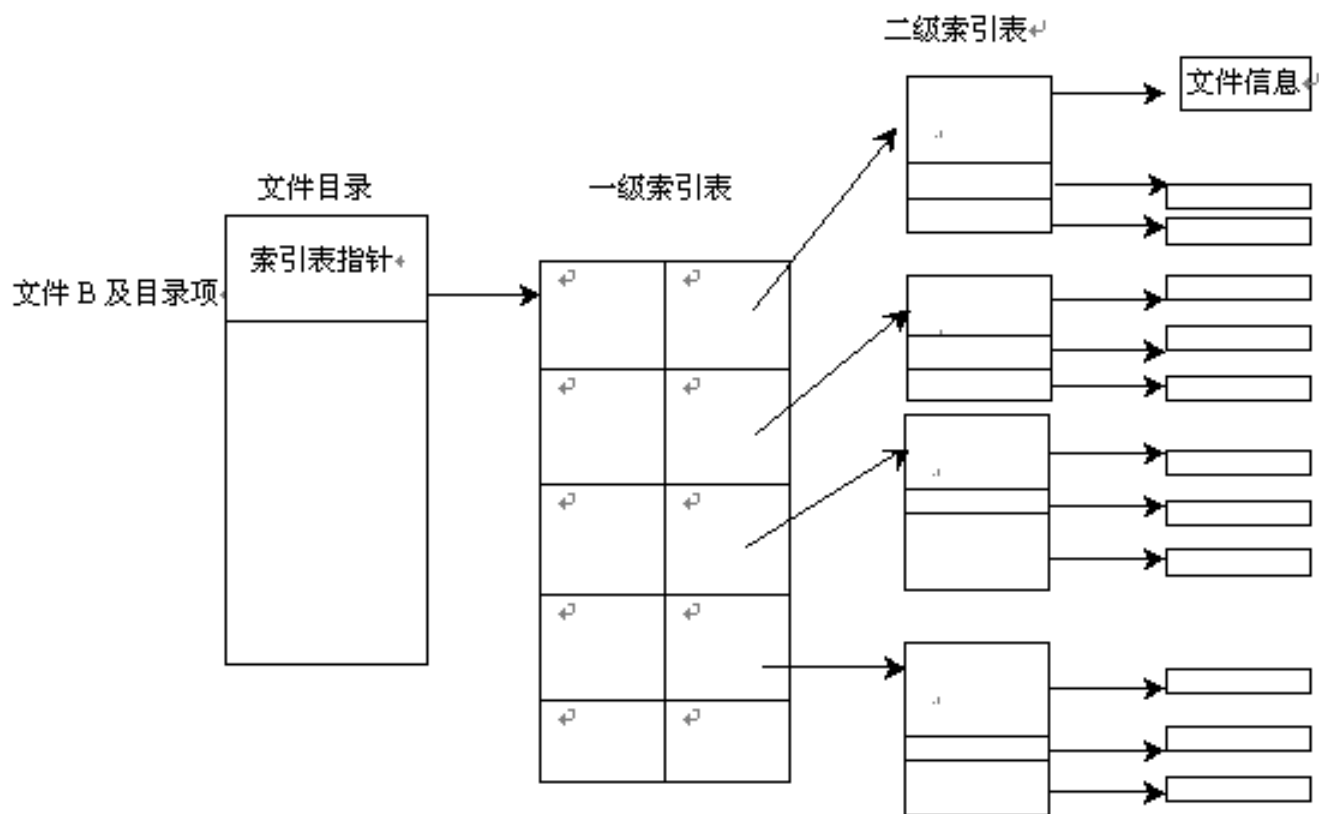
索引结构

单级索引分配



索引结构

多级索引分配



8.3.3. 外存空间的管理



- 空闲表法
- 位示图
- 空闲链表法
- 成组链接



空闲表法和空闲链表法

- 1. 空闲表法

序号	1	2	3	4	5
第一个空闲块号	3	8	20	31	-
空闲块个数	2	4	3	5	-
空闲块号	3 . 4	8 . 9 . 10 . 11	20 . 21 . 22	31 . 32 . 33 . 34 . 35	-



位示图

对应物理块: $b = n * i + j$

$(1, 7) \rightarrow 16 * 1 + 7 = 23$

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

位示图

物理块对应位示图哪一位

$$i = b \text{ DIV } n$$

$$j = b \text{ MOD } n$$

如 物理块 30 对应于 $(1, 14)$

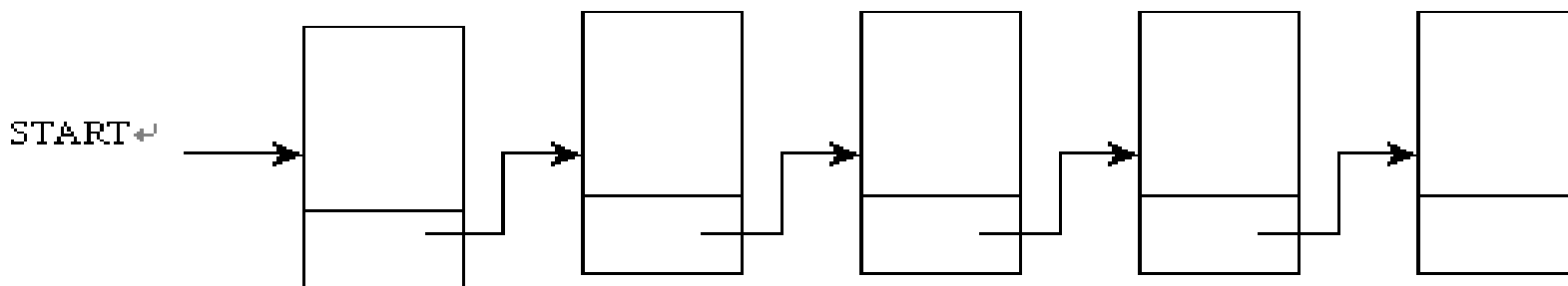
$$i = 30 \text{ DIV } 16 = 1$$

$$j = 30 \text{ MOD } 16 = 14$$



空闲块链接法

■ 空闲块（区）链



- 优点：实现简单，但工作效率低，因为每当在链上增加或移去空闲块时，都需要对空闲块链做较大的调整，因而会有较大的系统开销。
- 对空闲块链管理技术的改进方法是采用**成组空闲块链表**，即利用盘空闲块管理盘上的空闲块，每个磁盘块记录尽可能多的空闲块。



成组链接法（UNIX实现）

- 一、空闲盘块的组织。
 - 空闲盘块栈的第一个记录存放着下一组空闲块的索引盘块；其后为可以使用的空闲块的编号
- 二、空闲盘块的分配与回收
 - 分配：到s.free(0)时，由于该块内容为下一组的盘号，将内容加入空闲盘块号栈中，再分配。
 - 回收：到s.free(100)时，将空闲盘块栈中内容放入新到的回收块中，将该回收块作为栈底。
 - 评：好像是麻烦了点！

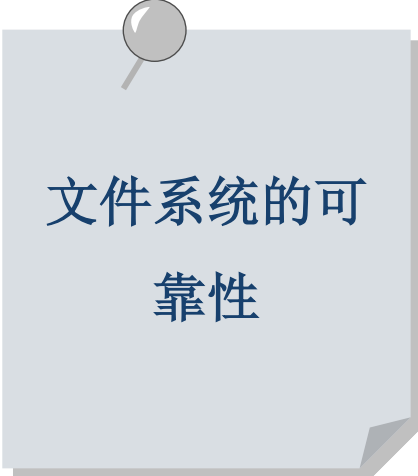


成组链接法

经过一段时间的分配、回收后.....



8.4 文件的可靠性与安全



文件系统的可
可靠性



文件保护



8.4.1 文件系统一致性问题

■ 盘块号一致性检查

- 系统构造一张表，表中为每个盘块设立两个计数器。一个计数器记录该块在文件中出现的次数，另一个记录该块在空闲块表中出现的次数。

■ 链接数一致性检查

- 在**UNIX**系统中，每个目录项含有一个索引结点号，用于指向文件的索引结点。对于一个共享文件，其索引结点号会在目录中出现多次。
- 在共享文件的索引结点中有一个链接计数**count**，用于指向共享该文件的用户数。在正常情况下，这两个数据应该是一致的，否则就出现了不一致性错误。



UNIX一致性检查工作过程：

两张表，每块对应一个表中的计数器，初值为0

- 表一：记录了每块在文件中出现的次数
- 表二：记录了每块在空闲块表中出现的次数



块号

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

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

使用中的块

0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

空闲块

文件系统状态：一致



块号

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

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

使用中的块

0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

空闲块

文件系统状态：块丢失



块号

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

1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

使用中的块

0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

空闲块

文件系统状态：空闲表中有重复块



块号

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

1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

使用中的块

0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

空闲块

文件系统状态：重复数据块

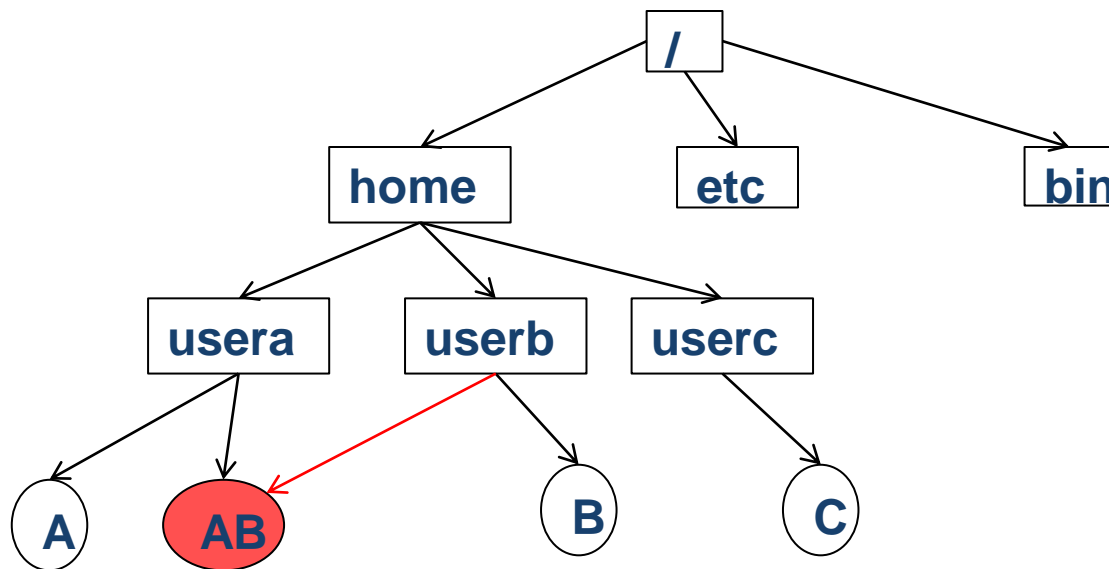


数据一致性控制

- **事务：**是用于访问和修改各种数据项的一个程序单位。
 - 提交（**Commit**）：当读写操作完成时，再以托付操作来终止事务。
 - 放弃（**Abort**）：当读写操作失败时，则执行夭折操作。
 - 日志记录**Log**：记录事务运行时数据项修改的全部信息。
- **恢复算法**利用以下两个过程：
 - Undo(Ti)**：把所有事务Ti修改过的数据，恢复为修改前的值。
 - Redo(Ti)**：把所有事务Ti修改过的数据，设置为新值。
- **检查点：**引入检查点的目的是使对事务记录表中事务记录的请求的清理工作经常化。



8.4.2 文件的共享与保护



但是存在问题！！
如果usera修改
文件长度，则...

Usera文件目录

文件名	首地址	长度	其他属性
A			
AB	100	125	...

Userb文件目录

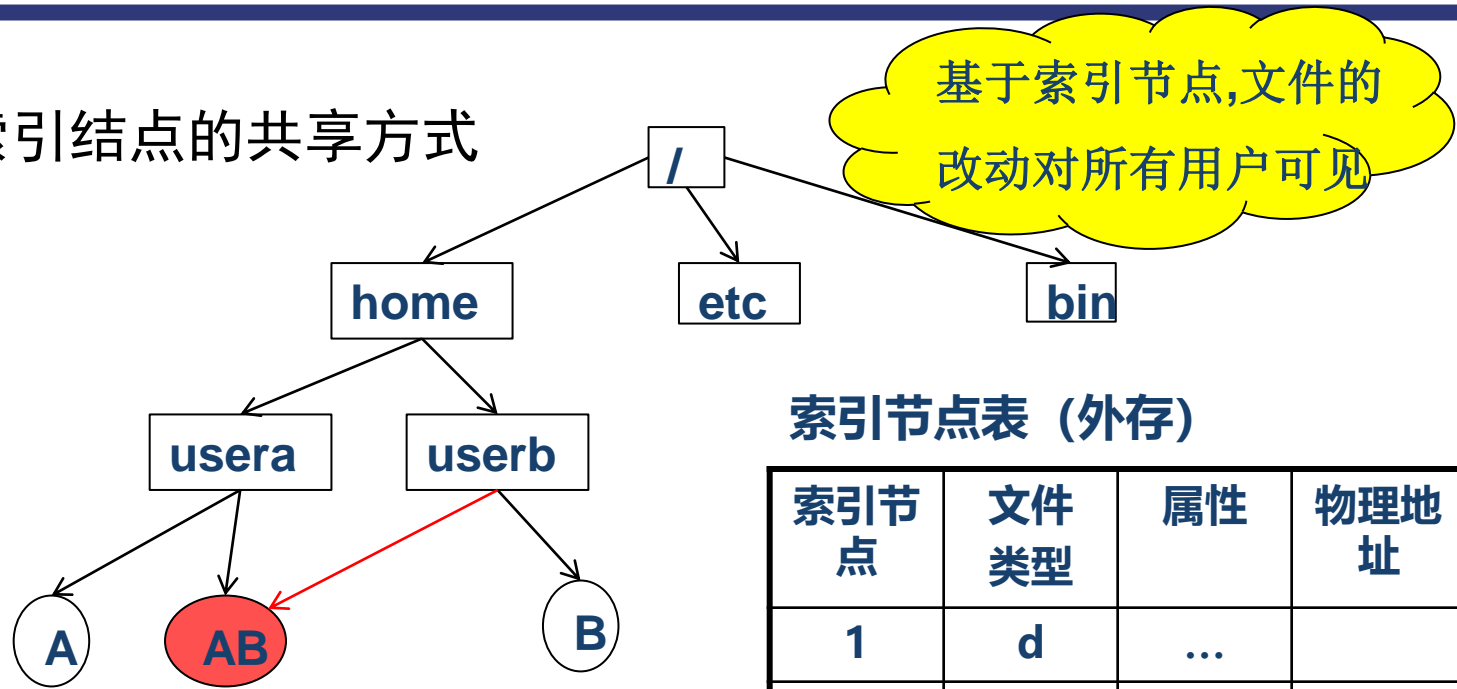
文件名	首地址	长度	其他属性
B			
AB	100	12	...

不一致！！



8.4.2 文件的共享与保护

- 1. 基于索引结点的共享方式



索引节点表 (外存)

索引节点	文件类型	属性	物理地址
1	d	...	
...	
6	d	...	132
...
26	d		496
...
60	f	...	200
			...

Usera文件目录

文件名	索引节点
A	
AB	60

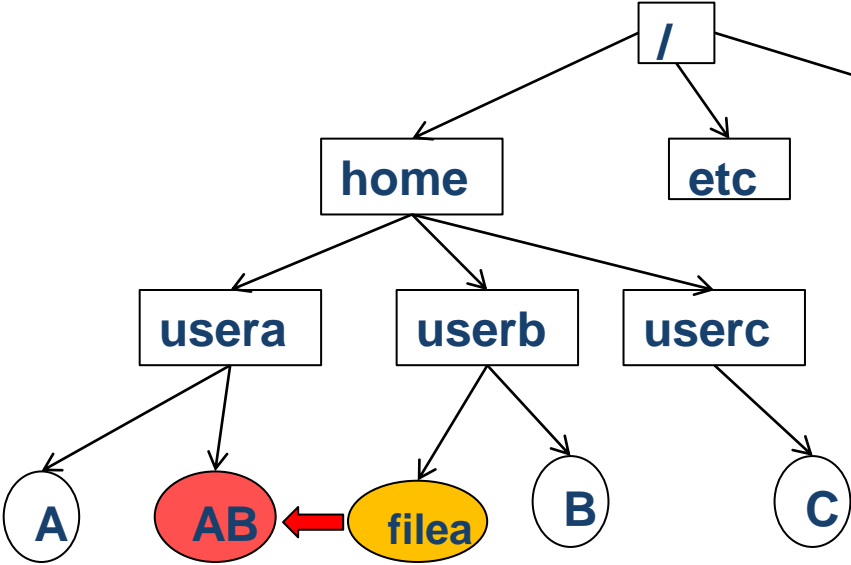
Userb文件目录

文件名	索引节点
B	
AB	60



8.4.2 文件的共享与保护

■ 2. 基于符号链的共享方式



索引节点表（外存）

索引节点	文件类型	属性	物理地址
1	d	...	
...	
6	d	...	132
...
26	f		496
...
60	link	/home/usera/A	
		B	

Usera文件目录

文件名	索引节点
A	
AB	26

Userb文件目录

文件名	索引节点
B	
File a	60



8.4.2 文件的共享与保护

- 1. 分级安全管理

- 1. 系统级安全管理
- 2. 用户级安全管理
- 3. 目录级安全管理
- 4. 文件级安全管理

- 2. 文件的存取权限



- 3. 文件安全

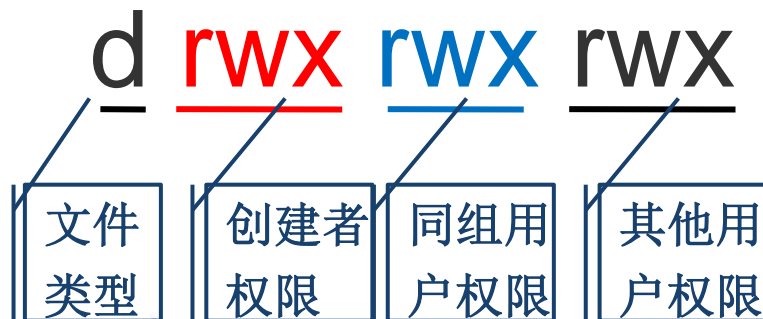
- 1. 数据丢失
- 2. 防范入侵者
- 3. 病毒防御



8.4.2 文件的共享与保护

■ 2. 文件的存取权限

- drwx----- 4 user wheel 512 Nov 25 17:23 Mail
- -rw-rw-r-- 1 user wheel 149 Dec 4 14:18 Makefile
- -rwxr-xr-x 1 user wheel 3212 Dec 4 12:36 a.out
- drwxr-xr-x 1 user wheel 512 Dec 14 17:03 bin
- -rw-r--r-- 1 user wheel 143 Dec 4 12:36 hello.c
- drwxr-xr-x 2 user wheel 1024 Oct 16 1997 public_html
- drwxrwxrwx 2 user wheel 512 Jan 3 14:07 tmp



8.4.2 文件的共享与保护

文件权限	用户1权限			用户2权限			用户3权限		
	R	W	X	R	W	X	R	W	X
文件A	√	×	√	√	√	√	√	√	√
文件B	√	×	√	×	×	×	×	×	√
文件C	√	√	√	√	√	√	√	√	√
文件D	√	√	√	√	√	√	√	√	√

存取控制矩阵



文件系统总结



- 文件 \Rightarrow 帮助用户将信息放在磁盘上
- 一个长长的字符序列 \Rightarrow 盘块集合 \Rightarrow 文件系统完成映射
- 操作(文件名,偏移) \Rightarrow 找到文件头(**FCB**), 找到磁盘盘块
- 系统中会有很多文件 \Rightarrow 为方便寻找, 组织成树状目录
- 目录表明某些文件在一个集合中 \Rightarrow 目录也是文件, 其内容是 \langle 文件名, **FCB**的指针 \rangle
- 树状目录 \Rightarrow 文件路径 \Rightarrow 路径解析
- **目录+FCB+文件盘块+空闲盘块 = 文件系统**
- 从可以运转到良好运转 \Rightarrow 高效、保护、容错、分布...



Do you have any questions ?



世界需要新的操作系统

- 每一个大型软件系统都值得尊重
- Windows老迈龙钟，历史负担太重，微软自己的创新Midori胎死腹中，因为无法承受在新的框架中重新实现一遍Windows的全部功能，只能在原地进行重构
- Linux里大部分开发人员只关心服务器的世界，就像一个专注于在甲板下面锅炉房里干活的锅炉工
- MacOS, iOS封闭在苹果的硬件生态里
- Android为了弥补Linux的缺点打上了一个厚厚的中间层，不断在做着妥协
- GNU Hurd作为GNU项目“最后的组件”一直未能产品化，原因是“微内核消息传递机制debug太困难”？
- Unix的后继者Plan 9于2002年发布了最后一个版本，它的余热随着作者融入了Go



参考资料

- 廖剑伟，操作系统，西南大学，重庆，2023

