

# 第七章 文件管理



# 第七章 文件管理

在现代计算机系统中，要用到大量的程序和数据，由于内存容量有限，且不能长期保存，故而平时总是把他们以文件的形式存放在外存中，需要时调入内存。

但用户不能够胜任管理文件的工作，于是在**OS**中又增加了文件管理功能，构成一个文件系统，负责管理在外存上的文件，**把文件的存取、共享和保护等手段提供给用户，方便了用户，保证了文件的安全，提高了系统资源的利用率。**



# 第七章 文件管理

**7.1 文件和文件系统**

**7.2 文件的逻辑结构**

**7.3 文件目录**

**7.4 文件共享**



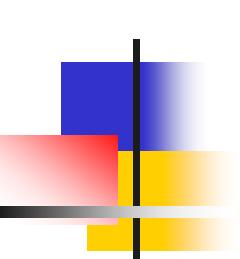
# 第七章 文件管理

**7.1 文件和文件系统**

**7.2 文件的逻辑结构**

**7.3 文件目录**

**7.4 文件共享**



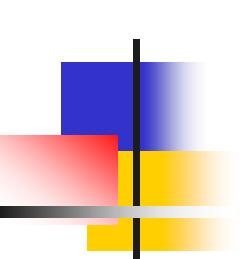
# 7.1 文件和文件系统

**7.1.1** 数据项、记录和文件

**7.1.2** 文件名和类型

**7.1.3** 文件系统的层次结构

**7.1.4** 文件操作



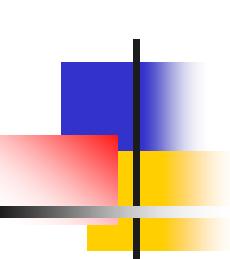
# 7.1 文件和文件系统

**7.1.1 数据项、记录和文件**

**7.1.2 文件名和类型**

**7.1.3 文件系统的层次结构**

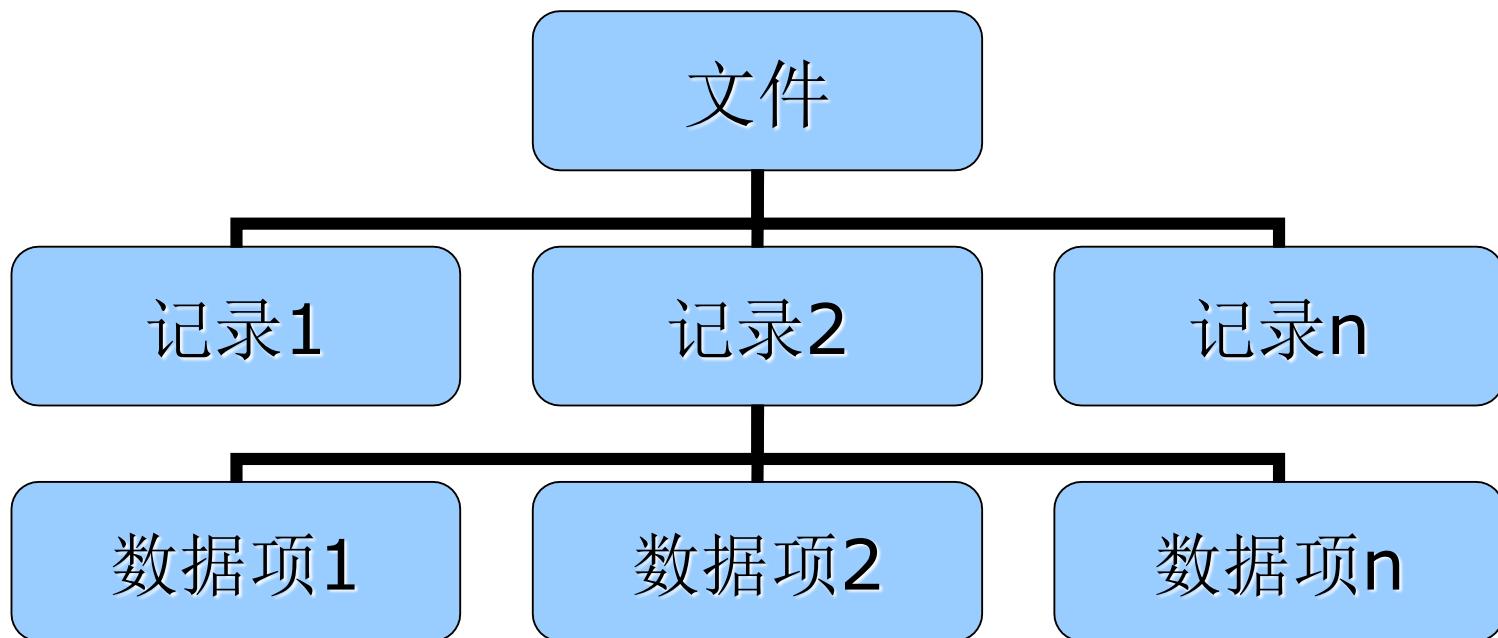
**7.1.4 文件操作**

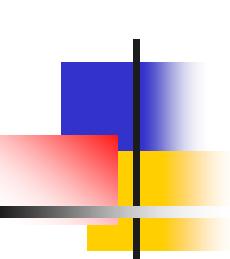


## 7.1.1 数据项、记录和文件

- 现代**OS**中几乎毫无例外的通过文件系统来组织和管理计算机中存储的数据；或者说文件系统的管理功能，是通过把它所管理的程序和数据组织成一系列文件的方法来实现的。
- **文件**则是指具有文件名的若干相关元素的集合。
- 基于文件系统的概念，可以把数据组成为**数据项、记录和文件**三级。

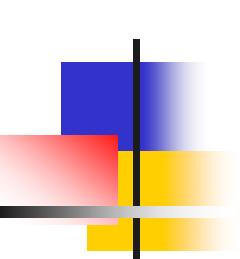
# 文件、记录和数据项之间的关系





# 文件属性

- **文件类型**: 可以从不同的角度来规定文件的类型。如源文件、目标文件及可执行文件。
- **文件长度**: 指文件的当前长度，长度的单位可以是字节、字或块，也可能是最大允许的长度。
- **文件的物理位置**: 通常是指示文件在哪一个设备上及在该设备的哪一个位置的指针。
- **文件的建立时间**: 指最后一次的修改时间等。



# 7.1 文件和文件系统

**7.1.1 数据项、记录和文件**

**7.1.2 文件名和类型**

**7.1.3 文件系统的层次结构**

**7.1.4 文件操作**

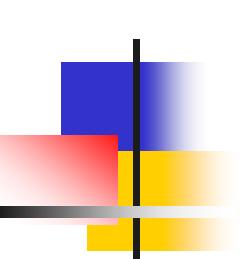
## 7.1.2 文件名和类型

### ■ 文件和扩展名

- 文件名：在不同的系统之间，对文件名的规定是不同的。
- 扩展名：又称后缀名，用于指示文件的类型。

### ■ 文件类型

- 按用途分类：系统文件、用户文件和库文件。
- 按文件中数据的形式分类：源文件、目标文件和可执行文件
- 按存取控制属性分类：只执行文件、只读文件和读写文件。



# 7.1 文件和文件系统

**7.1.1** 数据项、记录和文件

**7.1.2** 文件名和类型

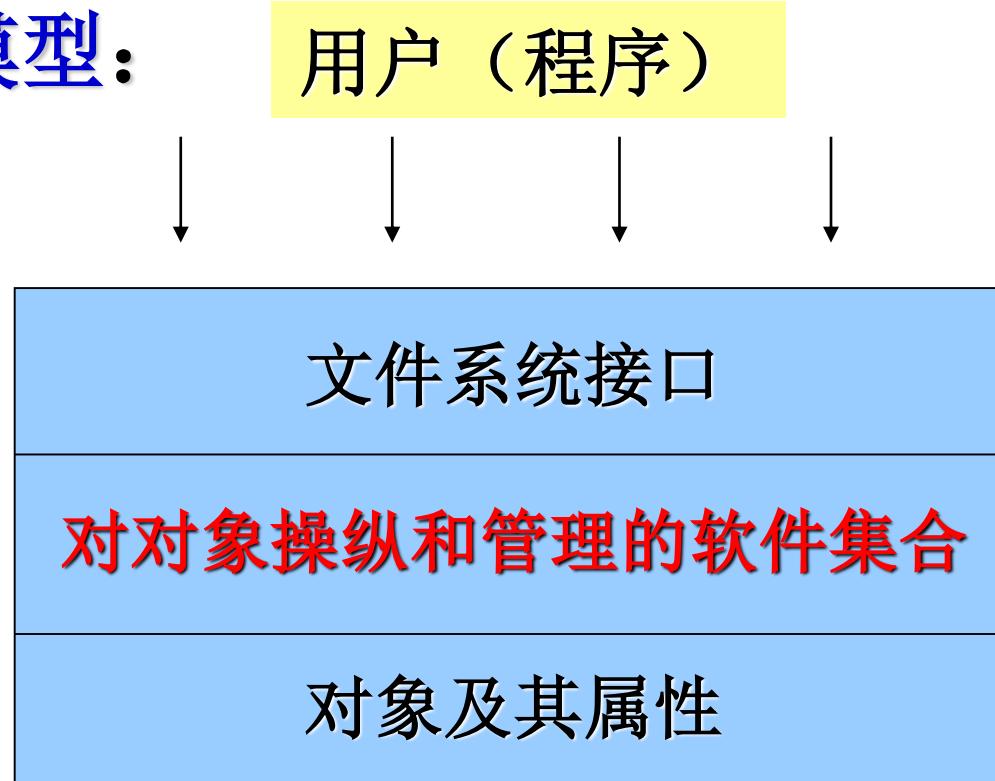
**7.1.3** 文件系统的层次结构

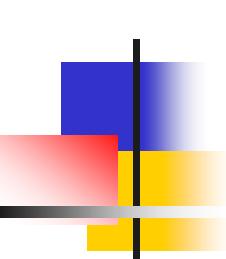
**7.1.4** 文件操作

## 7.1.3 文件系统的层次结构

### ■ 文件系统模型：

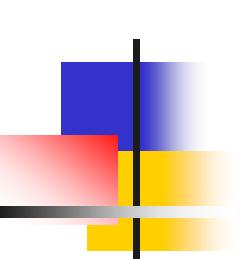
三个层次





# 文件系统模型

- 对象及其属性。文件管理系统的对象有：文件、目录和磁盘存储空间。
- 对对象操纵和管理的软件集合。是文件管理的核心部分。实现了文件系统的大部分功能——对文件存储空间的管理、对文件目录的管理、将文件的地址转换机制、对文件读写管理以及对文件的共享和保护。
- 文件系统的接口。命令接口（用户与文件系统）和程序接口（用户程序和文件系统）。



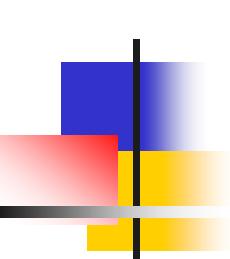
# 7.1 文件和文件系统

**7.1.1** 数据项、记录和文件

**7.1.2** 文件名和类型

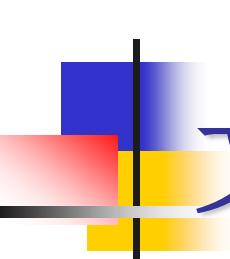
**7.1.3** 文件系统的层次结构

**7.1.4** 文件操作



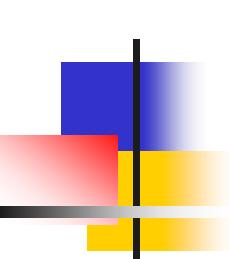
## 7.1.4 文件操作

- 用户通过文件系统所提供的系统调用实施对文件的操作。最基本的文件操作有：**创建文件、删除文件、读文件、写文件、截断文件和设置文件的读/写位置。**
- 但对于一个实际的**OS**，为了方便用户使用文件而提供了更多地对文件的操作，如打开和关闭一个文件及改变文件名等操作。



# 文件的“打开”和“关闭”操作

- 所谓“打开”，是指系统将指名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。以后当用户再要求对该文件操作时，便可利用系统所返回的索引号向系统提出操作请求。此时可直接利用索引号到打开文件表中查找，避免了再次检索。这样不仅节省大量检索开销而且显著提高操作速度。
- 当用户不再需要对该文件实施相应的操作时，可利用“关闭”此文件，OS将会把该文件从打开文件表中的表目上删除。



# 第七章 文件管理

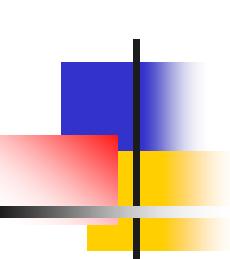
**7.1 文件和文件系统**

**7.2 文件的逻辑结构**

**7.3 文件目录**

**7.4 文件共享**

**7.5 文件保护**



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

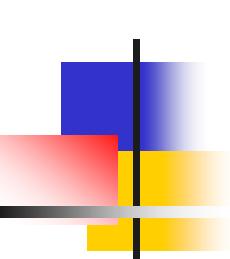
**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

**7.2.5** 索引顺序文件

**7.2.6** 直接文件和哈希文件



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

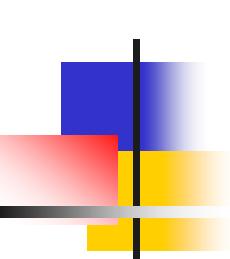
**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

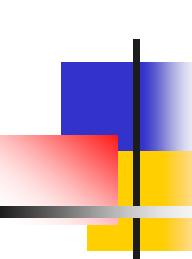
**7.2.5** 索引顺序文件

**7.2.6** 直接文件和哈希文件



# 文件的结构

- 通常文件是由一系列的记录组成的。文件系统设计的关键要素，是将这些记录构成一个文件的方法，以及将一个文件存储到外存上的方法。事实上任何一个文件都存在着以下两种形式的结构：
  1. **文件的逻辑结构**。从用户观点出发所观察到的文件组织形式，是用户可以直接处理的数据及其结构，它独立于文件的物理特性，又称为文件组织。
  2. **文件的物理结构**。又称为文件的存储结构，是指文件在外存上的存储组织形式。与存储介质的存储性能和采用的外存分配方式有关。



## 7.2.1 文件逻辑结构的类型

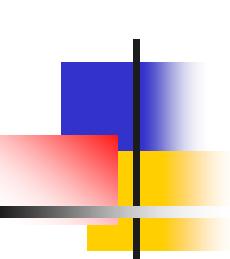
可以分为两大类：

- **有结构文件**

- 是指由一个以上的记录构成的文件，故又把它称为记录式文件；
- 根据记录的长度可分为**定长记录文件；不定长记录文件。**

- **无结构文件**

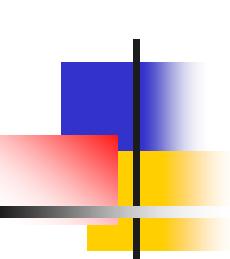
- 这是指由字符流构成的文件，故又称为是流式文件。



# 有结构文件

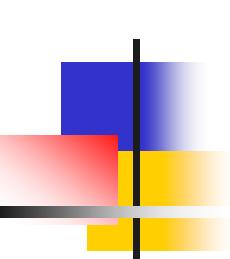
根据记录的组织方式分为下列文件：

- **顺序文件**。由一系列记录按某种顺序排列所形成的文件。通常是定长记录。
- **索引文件**。当记录可变长时，通常为之建立一张索引表，并为每个记录设置一个表项以加快对记录检索的速度。
- **索引顺序文件**。上述两种方式的结合。为文件建立一张索引表，为每一组记录中的第一个记录设置一个表项。



# 无结构文件

- 如果说大量的数据结构和数据库，是采用有结构的文件形式的话，则大量的源程序、可执行文件、库函数等，所采用的就是无结构的文件形式，即**流式文件**。其长度以字节为单位。对流式文件的访问，则是采用读写指针来指出下一个要访问的字符。
- 可以把流式文件看作是记录文件的一个特例。



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

**7.2.5** 索引顺序文件

**7.2.6** 直接文件和哈希文件

## 7.2.2 顺序文件

- 逻辑记录的排序
  - **串结构**: 各记录之间的顺序与关键字无关。通常由时间来决定。
  - **顺序结构**: 文件中的所有记录按关键字排列。可以按关键字的长短或英文字母顺序排序。顺序结构的检索效率更高

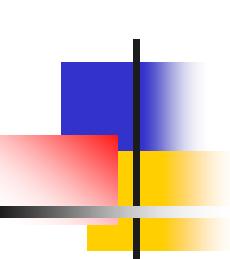
### ■ 顺序文件的优缺点

#### 优点:

- 顺序文件的最佳应用场合是在对诸记录进行批量存取时，即每次操作一大批记录。只有顺序文件才能存储在磁带上，并能有效的工作。

#### 缺点:

- 在交互应用的场合，如果进程操作对象是单个记录，顺序文件的性能就可能很差。当文件较大时更差。
- 如想增加或删除一个记录都比较困难。



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

**7.2.5** 索引顺序文件

**7.2.6** 直接文件和哈希文件

## 7.2.3 记录寻址

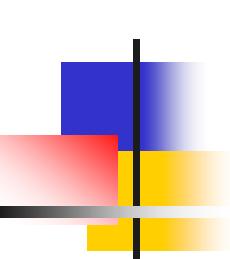
为了访问顺序文件中的一条记录，首先应找到该记录的地址。

### ■ 隐式寻址方式：

- 对于定长记录的顺序文件，如果已知当前记录的逻辑地址，很容易确定下一个记录的逻辑地址。
- 对于变长记录的顺序文件，与顺序读或写时的情况相似，只是每次都需要从正在读（写）的记录中读出该记录的长度。

### ■ 显示寻址方式：

- 该方式可用于对定长记录的文件实现直接或随机访问。因为任何记录的位置都很容易通过记录长度计算出来。
- 而对于可变长度记录的文件则不能利用显示寻址方式实现直接或随机访问，必须增加适当的支持机构方能实现。



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

**7.2.2** 顺序文件

**7.2.3** 记录寻址

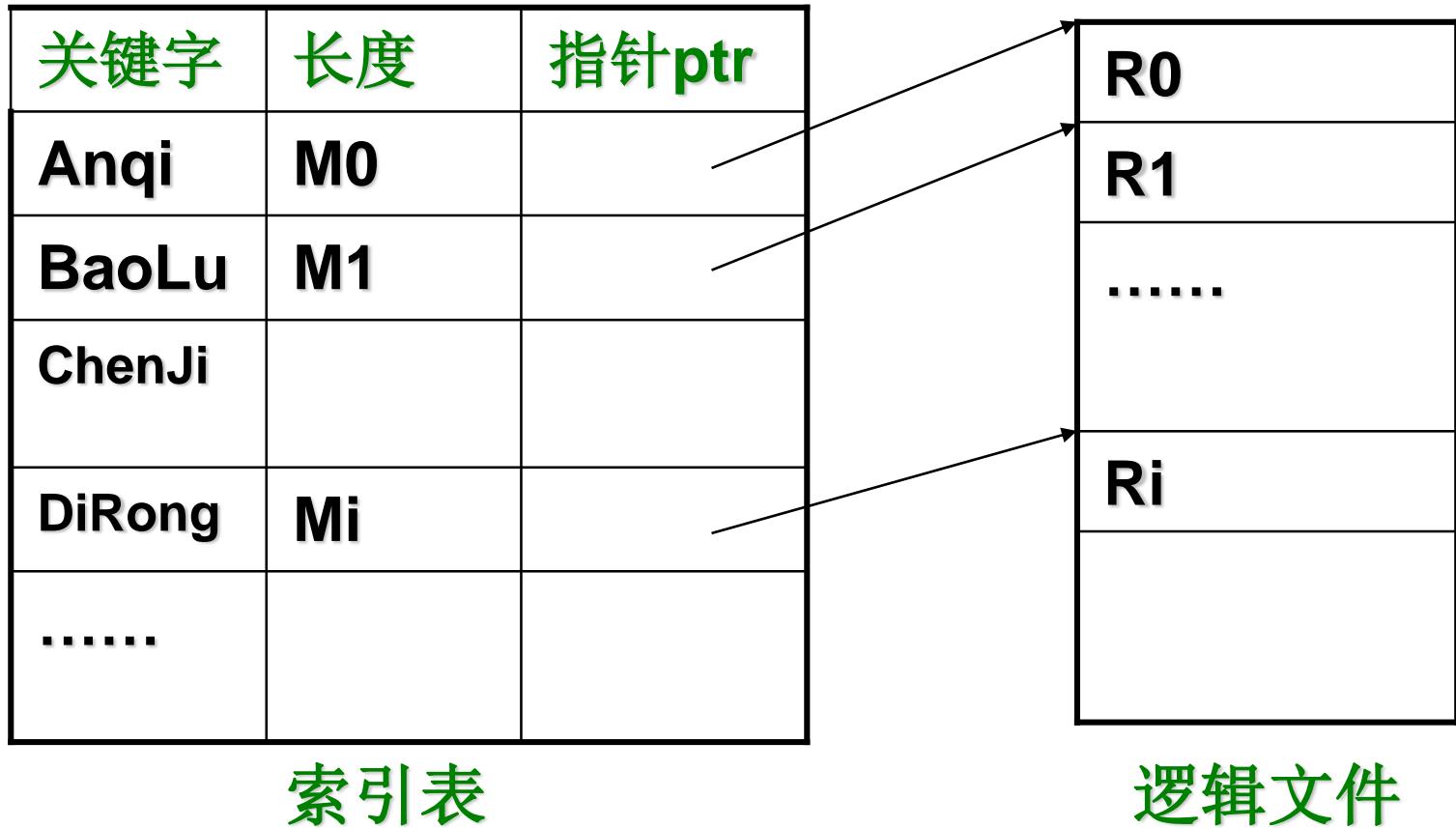
**7.2.4 索引文件**

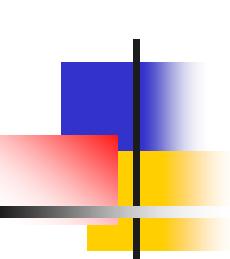
**7.2.5 索引顺序文件**

**7.2.6 直接文件和哈希文件**

## 7.2.4 索引文件

### 索引文件的组织





## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

**7.2.5** 索引顺序文件

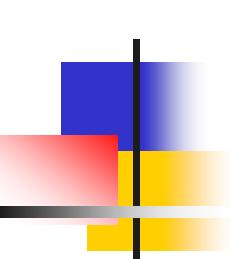
**7.2.6** 直接文件和哈希文件

## 7.2.5 索引顺序文件

索引顺序文件示意图

键	逻辑地址
An Qi	
BaoRong	
ChenLin	

姓名	其他属性
An Qi	
An Kang	
BaoRong	



## 7.2 文件的逻辑结构

**7.2.1** 文件逻辑结构的类型

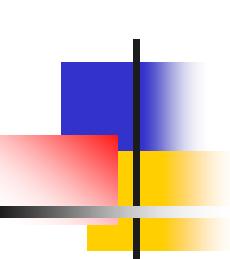
**7.2.2** 顺序文件

**7.2.3** 记录寻址

**7.2.4** 索引文件

**7.2.5** 索引顺序文件

**7.2.6** 直接文件和哈希文件



## 7.2.6 直接文件和哈希文件

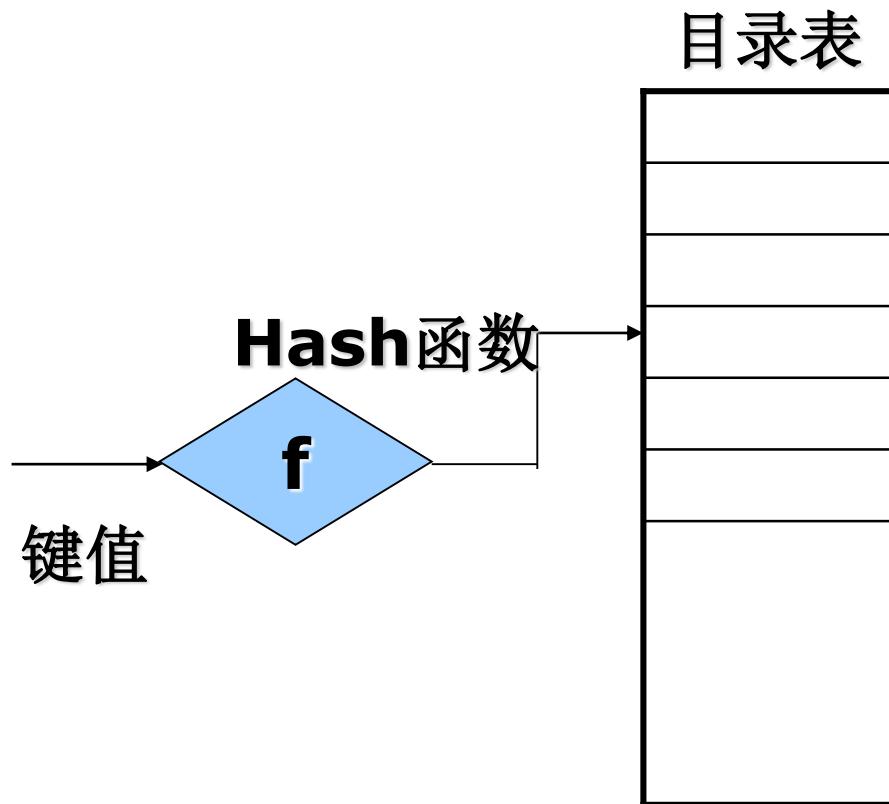
### ■ 直接文件

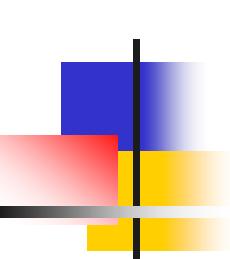
采用前述几种文件结构对记录进行存取时，都须利用给定的记录键值，先对线性表进行检索，以找到指定记录的物理地址。然而对直接文件，则可根据给定的记录键值，直接获得指定记录的物理地址。换言之，记录键值本身就决定了记录的物理地址，组织直接文件的关键，在于用什么方法进行从记录值到物理地址的转换。

## 7.2.6 直接文件和哈希文件

### Hash文件的逻辑结构

- 若令K为记录键值，用A作为通过**Hash**函数转换所形成的该记录在目录表中对应表目的位置，则有关系  $A=H(K)$ 。通常把 **Hash**函数作为标准函数存于系统中，供存取文件时调用。





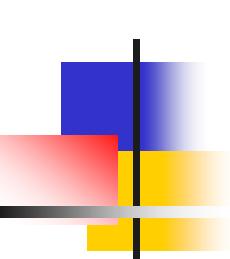
# 第七章 文件管理

**7.1 文件和文件系统**

**7.2 文件的逻辑结构**

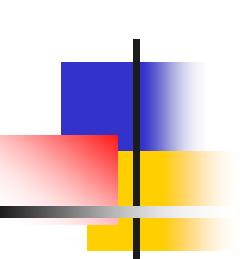
**7.3 文件目录**

**7.4 文件共享**



## 7.3 文件目录

- 通常在现代计算机系统中，都要存储大量的文件。为了能对这些文件实施有效的管理，必须对它们加以妥善组织，这主要是通过文件目录来实现的。**文件目录也是一种数据结构，用于标识系统中的文件及其物理地址，供检索时使用。**对目录管理的要求如下：
  - 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。



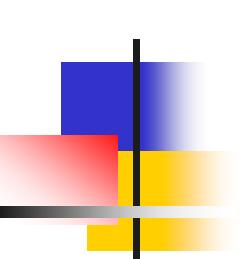
## 7.3 文件目录

**7.3.1** 文件控制块和索引结点

**7.3.2** 简单的文件目录

**7.3.3** 树形结构目录

**7.3.4** 目录查询技术



## 7.3 文件目录

**7.3.1 文件控制块和索引结点**

**7.3.2 简单的文件目录**

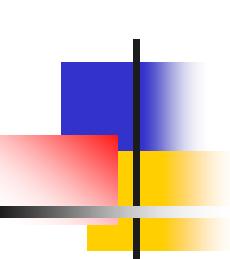
**7.3.3 树形结构目录**

**7.3.4 目录查询技术**

## 7.3.1 文件控制块和索引结点

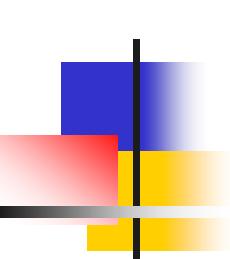
为了能对一个文件进行正确的存取，必须为文件设置用于描述和控制文件的数据结构，称之为“**文件控制块（FCB）**”文件管理程序可借助于文件控制块中的信息对文件施以各种操作。**文件与文件控制块一一对应**，而人们把文件控制块的有序集合称为**文件目录**，即一个文件控制块就是一个文件目录项。通常一个文件目录也被看作是一个文件，称为**目录文件**。

- **FCB**通常含有以下三类信息：
  - **基本信息类**。包括：文件名，文件物理位置，文件逻辑结构，文件的物理结构。
  - **存取控制信息类**。包括：文件主的存取权限，核准用户的存取权限和一般用户的存取权限。
  - **使用信息类**。包括：文件的建立日期和时间、文件上次修改的日期和时间及当前使用信息。



# MS-DOS的文件控制块

文件名	扩展名	属性	备用	时间	日期	第一块号	盘块数



# 索引结点

## ■ 索引结点的引入

文件目录通常是存放在磁盘上的，当文件很多时，文件目录要占用大量的盘块。在检索目录文件的时候，需要将目录调入内存后比较文件名，但是只用到文件名，而不需要其它那些对文件的描述信息。所以便把文件名与文件信息分开，使文件描述信息单独形成一个索引结点。

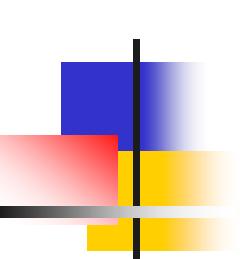
见P234图7-8

## 磁盘索引结点

- 这是存放在磁盘上的索引结点。每个文件有唯一的一个磁盘索引结点，主要包括以下内容：
  - 文件主标识符。
  - 文件类型。
  - 文件存取权限。
  - 文件物理地址。
  - 文件长度。
  - 文件链接计数。
  - 文件存取时间。

## 内存索引结点

- 这是放在内存中的索引结点。当文件被打开后，将磁盘索引结点拷贝到内存索引结点中以便使用。**比磁盘索引结点又增加了以下内容：**
  - 索引结点编号；状态；访问计数；
  - 文件所属文件系统的逻辑设备号；链接指针。



## 7.3 文件目录

**7.3.1** 文件控制块和索引结点

**7.3.2** 简单的文件目录

**7.3.3** 树形结构目录

**7.3.4** 目录查询技术

## 7.3.2 简单的文件目录

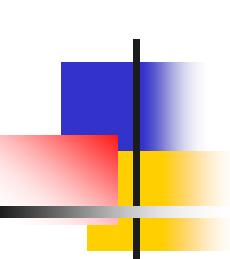
- 目录结构的组织，关系到文件系统的存取速度，也关系到文件的共享性和安全性。因此，组织好文件的目录是设计好文件系统的重要环节。**目前常用的目录结构形式有单级目录、两级目录和多级目录。**我们分别来介绍。

### 单级目录结构

- 最简单的目录结构。整个文件系统中只建立一张目录表，每个文件一个目录项，目录项含有文件相关信息。状态位表明每个目录项是否空闲。

文件名	物理地址	文件说明	状态位
文件名1			
文件名2			
.....			

文件不允许重名



# 单级目录特点

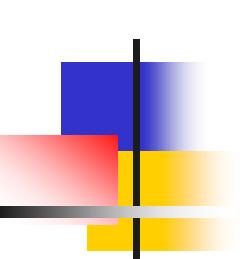
- 优点：简单且能实现目录管理的基本功能。
- 缺点：
  - 查找速度慢。
  - 不允许重名。名字过多难于记忆。多用户环境下重名难以避免。
  - 不便于实现文件共享。

# 两级目录结构

- 为了克服单级目录所存在的缺点，可以为每个用户建立一个单独的**用户文件目录UFD**。由该用户所有文件的文件控制块组成。
- 在系统中再建立一个**主文件目录MFD**。在主文件目录中每个用户目录文件都占有一个目录项，其中包括用户名和指向该用户文件的指针。**如图p236**。

## 两级目录的特点

- 基本克服了单级目录的缺点，并具有以下**优点**：
  - 提高了检索目录的速度。
  - 在不同的目录中可以有相同的文件名。
  - 不同用户还可以使用不同的文件名来访问系统中的同一个共享文件。
- 存在的问题是各用户之间被完全隔离了，无法进行合作。



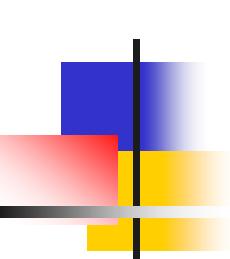
## 7.3 文件目录

**7.3.1** 文件控制块和索引结点

**7.3.2** 简单的文件目录

**7.3.3** 树形结构目录

**7.3.4** 目录查询技术



## 7.3.3 树形结构目录

- 目录结构

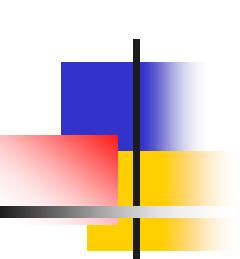
大型系统通常采用三级或三级以上的目录结构，以提高目录的检索速度和文件系统的性能。多级目录结构又称为**树型目录结构**，主目录称为**根目录**，数据文件为**树叶**，其它目录为**结点**。

- 路径名。

- 当前目录。

- 多级目录结构

**见图P237**



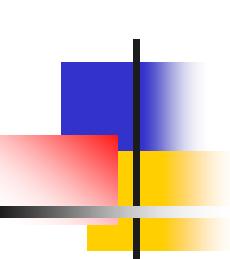
## 7.3 文件目录

**7.3.1** 文件控制块和索引结点

**7.3.2** 简单的文件目录

**7.3.3** 树形结构目录

**7.3.4** 目录查询技术



## 7.3.4 目录查询技术

- 当用户要访问一个已存文件时，系统首先利用用户提供文件名对目录进行查询，找出该文件控制块或对应索引结点；然后根据**FCB**或索引结点中所记录的文件物理地址，换算出文件在磁盘上的物理位置；最后通过磁盘驱动程序，将所需文件读入内存。
- 目前对目录进行查询的方式有两种：线性检索法和**Hash**方法。

# 线性检索法

- 又称为顺序检索法。  
假定用户给定的文件路径是  
**/usr/ast/mbox**, 则查找过程

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 号盘块是  
/usr 的目录

6	•
1	• •
19	dick
30	erik
51	jim
26	ast
45	bal

结点 26 是  
/usr/ast 的目录

496
-----

496 号盘块是  
/usr/ast 的目录

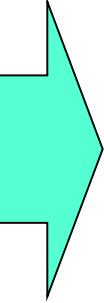
26	•
6	• •
64	grants
92	books
60	mbox
81	minik
17	src

在结点 6 中查找  
usr 字段

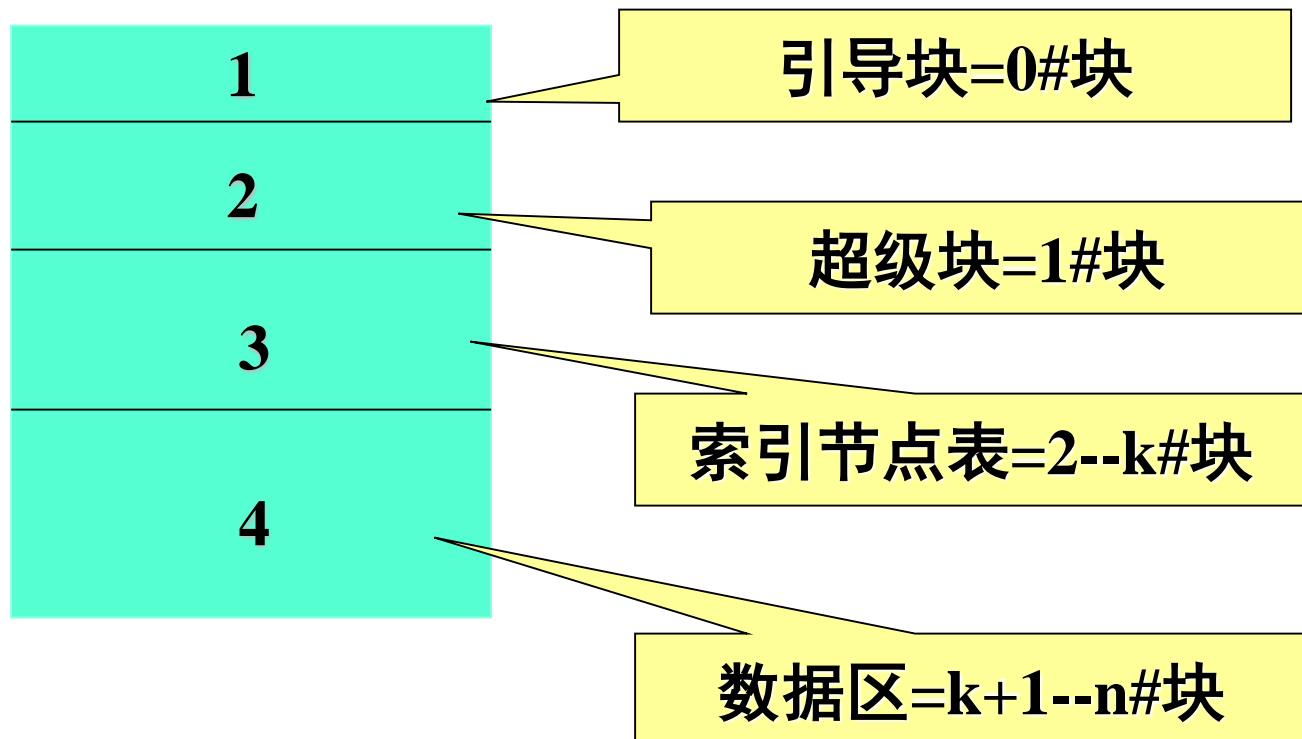
# 内容回顾

## 7.3 文件目录

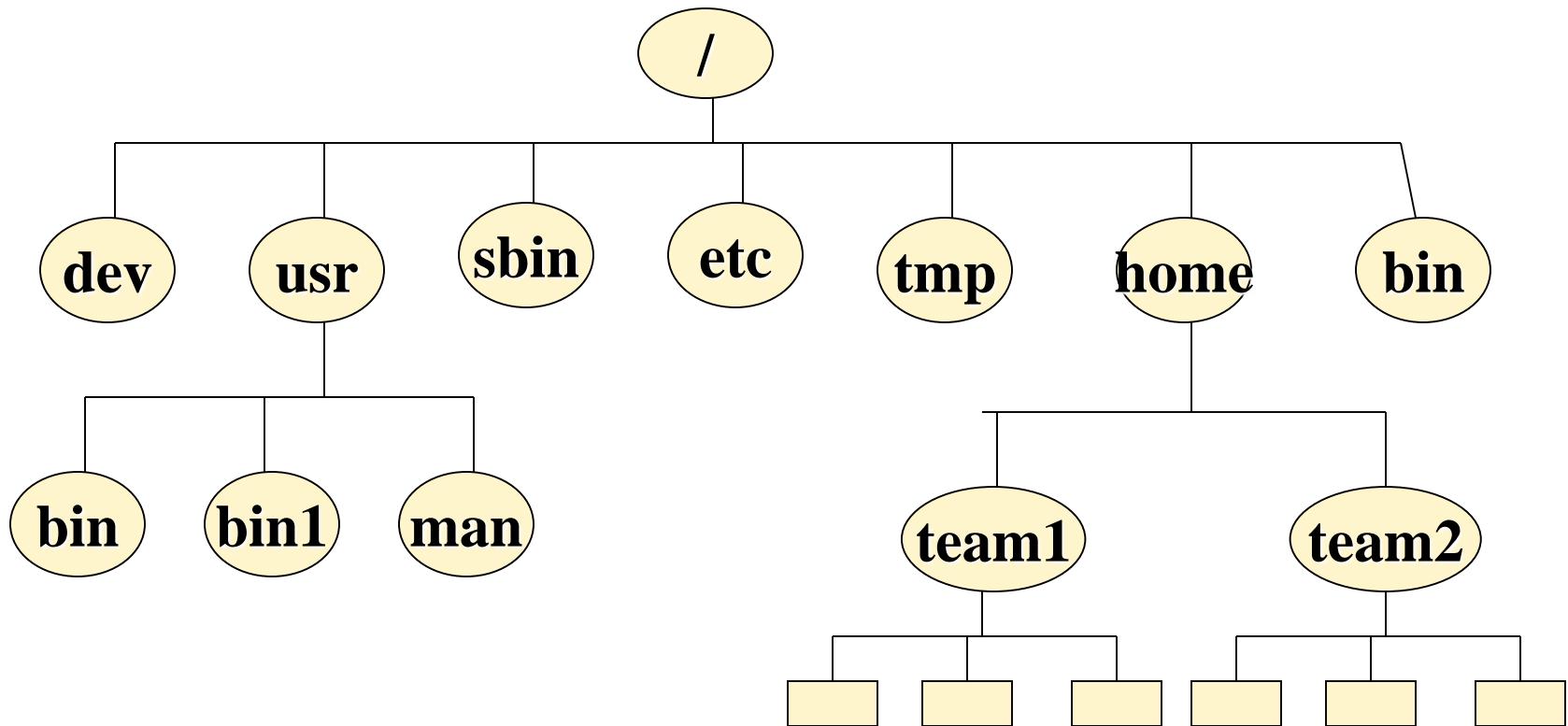
目录管理的要求

- 
- 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。

# UNIX文件系统由四部分构成：



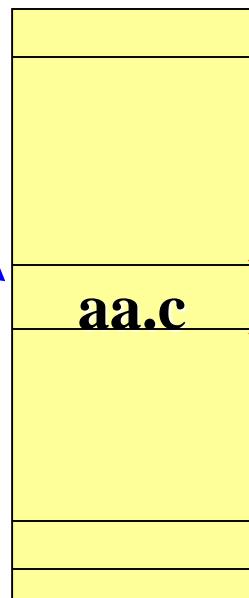
# UNIX目录结构—分层的倒置树状结构



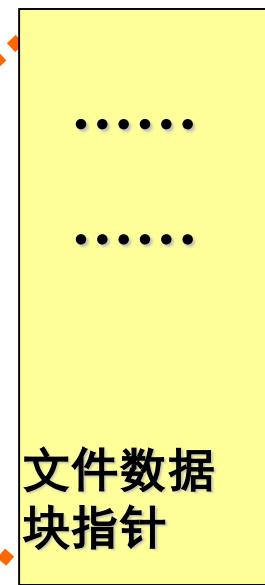
# 目录中的文件如何与磁盘文件对应：

目录中内容

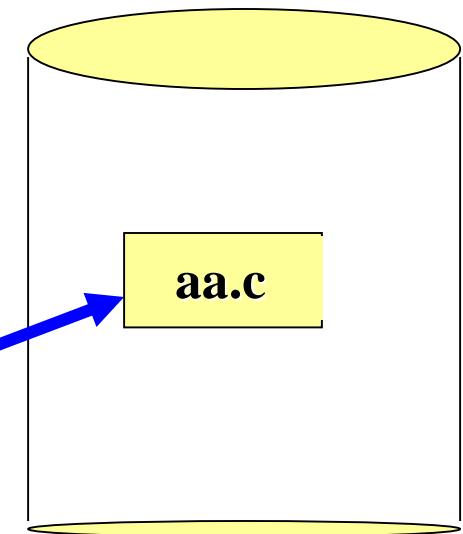
1076	....
1080	aa.c
2276	bb.c
3476	ff.c



索引表



aa.c索引节点



磁盘设备

例、某磁盘的文件系统采用混合索引分配方式，其FCB中共有13个地址项，第0~9地址项为直接地址，第10个地址为一次间接地址，第11个地址为二次间接地址，第12个地址为一次间接地址。若盘块大小为512B，每个盘块号用3B描述，为方便计算设每个盘块最多存170个盘块地址。问：

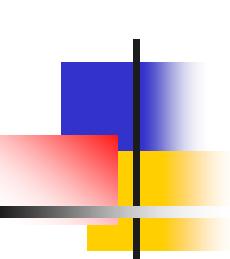
- 1) 该文件系统允许文件的最大长度是多少？
- 2) 将文件的字节偏移量5000、15000、150000转换为物理块号和块内地址
- 3) 设某个文件的FCB已在内存，但其他信息均在外存，为了访问该文件中某个位置的内容，最少和最多需几次访问磁盘？

答：1)  $(10 + 170 + 170 \times 170 + 170 \times 170 \times 170) * 512 B = 2471040B$

3) 由于FCB已在内存，为访问文件中某位置的内容，最少需1次访问磁盘（即可通过直接地址直接读文件块）；最多需4次访问磁盘（第1次为读3次间址，第2次为读2次间址，第3次为读1次间址，第4次为读文件盘块）。

可从FCB的第10个地址项得到一次间址块的地址，并从一次间址块的第19项（即该块的第57~59三个字节中）获得对应的物理块号，块内地址为152.

2)  $150000 / 512$ , 商为292, 余数为496。 $9+170 < 292 < 10+170+170 \times 170$ , 而 $292 - (10+170) = 112$ 。 $112 / 170$ , 商为0, 余数为112. 故可从FCB的第11个地址项得到二次间址块的地址，并从二次间址块的第0项一次间址地址，再从该一次间址的第112项中获得对应的物理块号，块内地址为496.



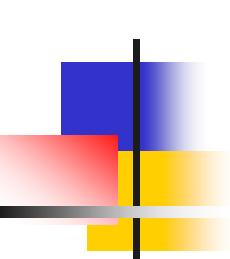
# 第七章 文件管理

**7.1 文件和文件系统**

**7.2 文件的逻辑结构**

**7.3 文件目录**

**7.4 文件共享**

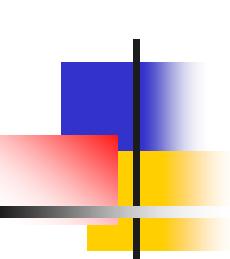


# 关于共享

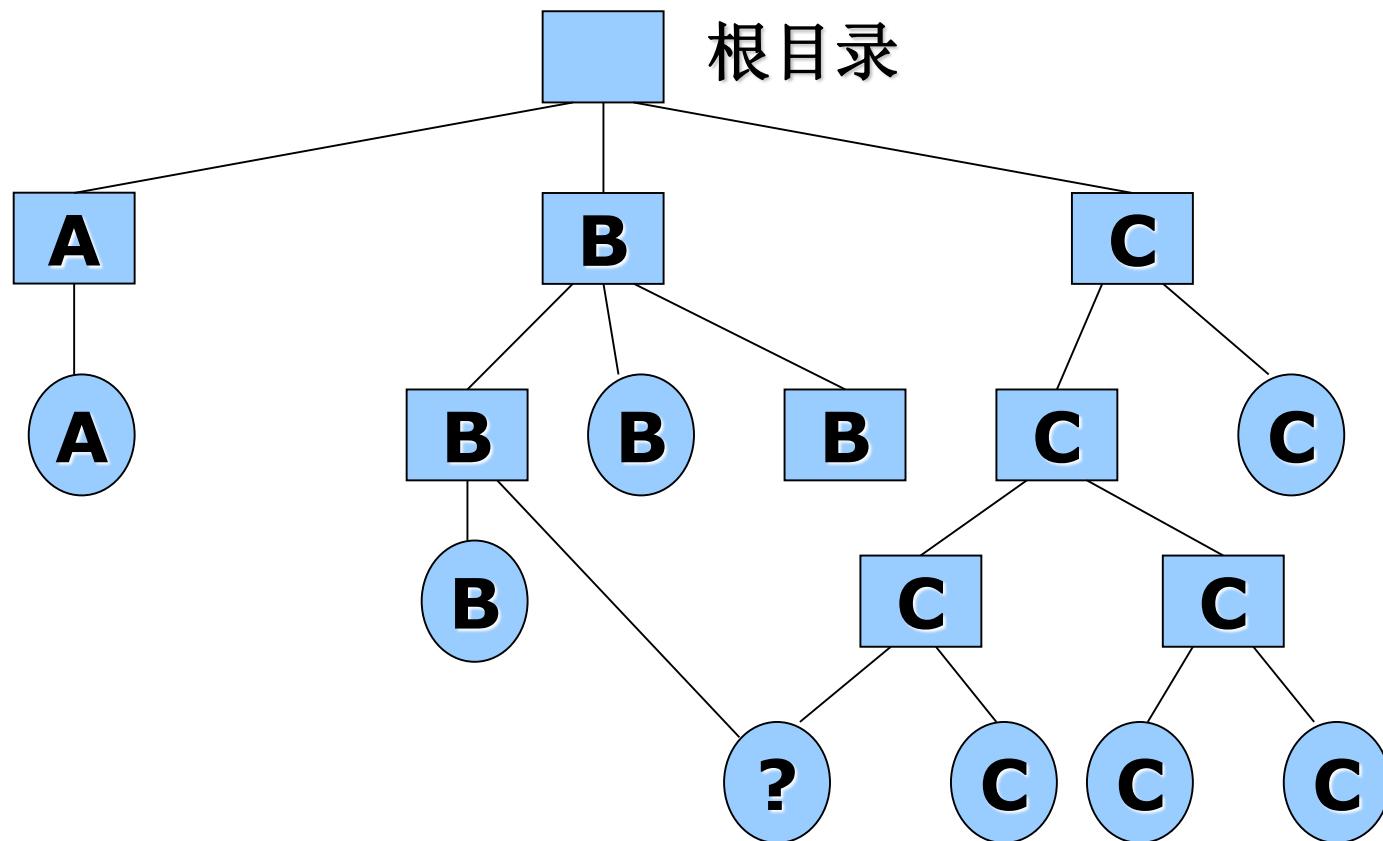
- 现代**OS**中必须提供文件共享手段，即指系统应允许多个用户（进程）共享同一份文件。这样系统中只需有一份副本。下面介绍两种文件共享方法。

# 1. 基于索引结点的共享方式

- 在树型结构的目录中，当有两个（或多个）用户要共享一个子目录或文件时，必须将共享文件或子目录连接到两个（或多个）用户的目录中，才能方便得找到该文件。此时该文件系统的目录结构已不再是树型结构，而是个有向非循环图。见下页图。
- 如果通过在文件目录中包含文件的物理地址的方法实现共享有很多缺陷。故采用基于索引结点的共享方式。



# 包含有共享文件的文件系统示意图



# 基于索引结点的共享方式示意图

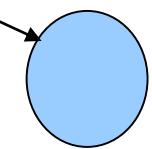
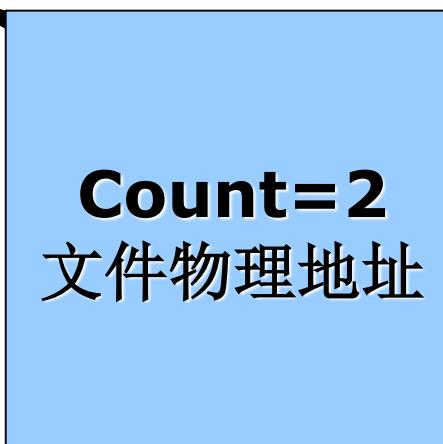
**Wang**用户文件目录

<b>Test r</b>	

**Lee**用户文件目录

<b>Test r</b>	

索引结点



**Test**

## 2. 利用符号链实现文件共享

- 为使**B**能共享**C**的一个文件**F**, 可以由系统创建一个**LINK**类型的新文件, 也取名为**F**, 并将**F**写入**B**的目录中, 以实现**B**的目录与文件**F**的链接。在新文件中只包含被链接文件**F**的路径名。这样的链接方法被称为**符号链接**。新文件中的路径名, 则被看作是**符号链**, 当**B**要访问被链接的文件**F**且正要读**LINK**类新文件时, 将被**OS**截获, **OS**根据新文件中的路径名去读该文件, 于是就实现了**B**对文件**F**的共享。