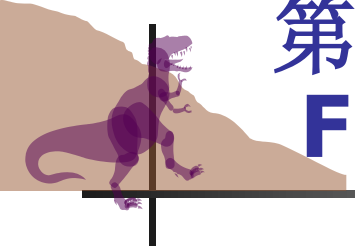
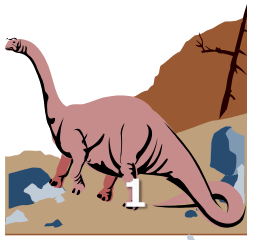


# 第15章 文件系统内部
# File-system internals

- 本章目标 CHAPTER OBJECTIVES
  - 深入研究文件系统及其实现细节

    Delve into the details of file systems and their implementation
  - 探索引导及文件系统共享

    Explore booting and file sharing



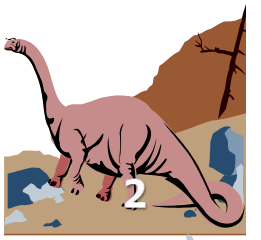
# 15.1 文件系统
# File Systems

- 磁盘可以分为多个部分，每个部分称为分区或小型磁盘

  Disk can be divided into several parts, these parts are known as partitions，or minidisks.

- 每个磁盘分区可以创建一个文件系统

  A file system can be created on each of these parts of the disk.

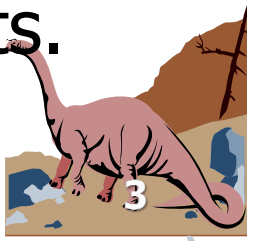


# 存储结构 2
# Storage Structure

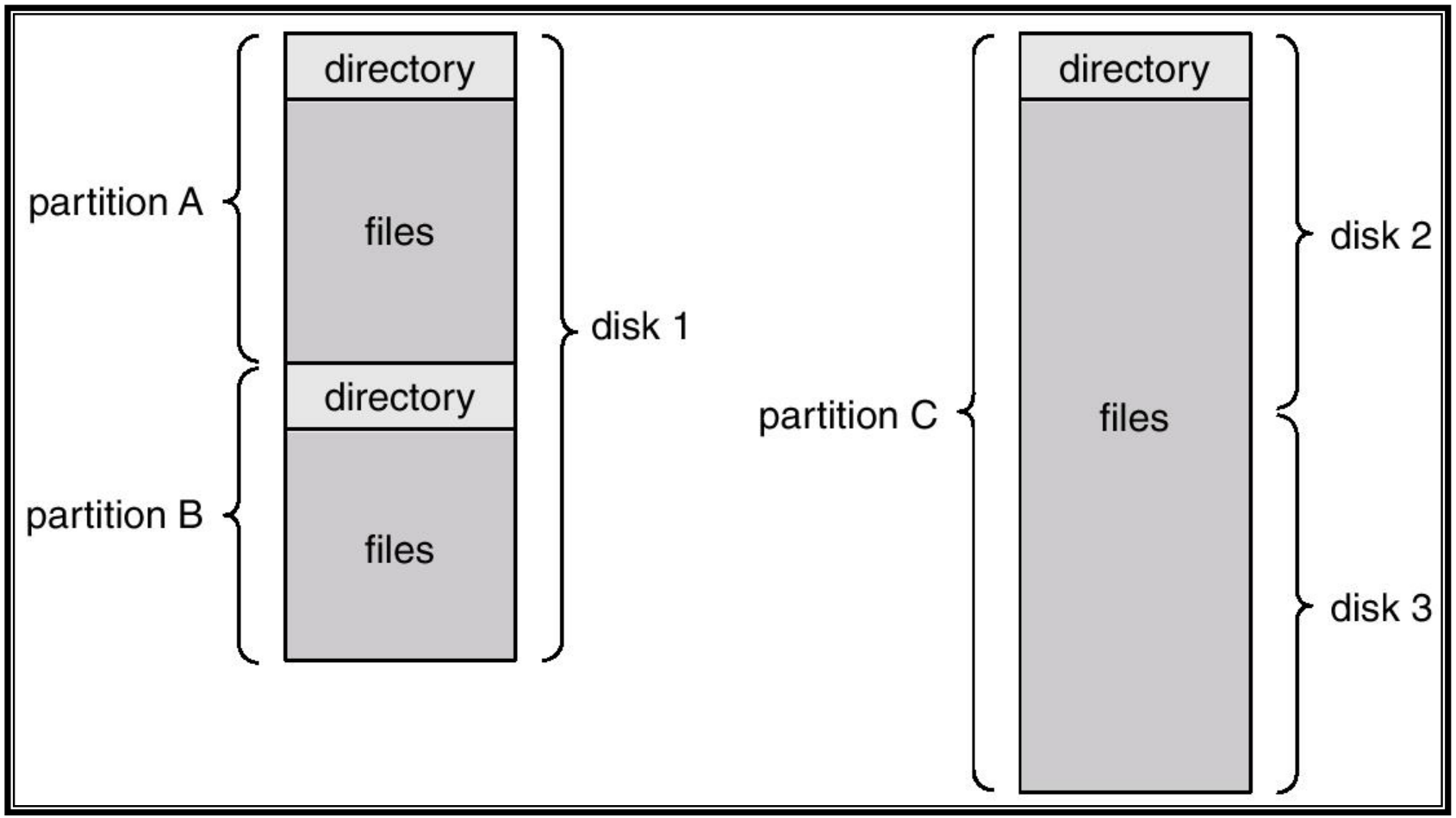
- 这些部分可以组合成更大的结构-卷。卷可以看成虚拟磁盘。

  The parts can be combined to form larger structures knows as volumes . Each volume can be thought of as a virtual disk.
- 包含文件系统的每个卷还必须包含系统上的文件信息。这些信息保存在设备目录（简称目录）或卷表中。

  Each volume that contains a file system must also contain information about the files in the system . This information is kept in entries in a device directory or volume table of contents.



# 典型的文件系统组织
# A Typical File-system Organization



# 15.2 文件系统安装
# File-System Mounting

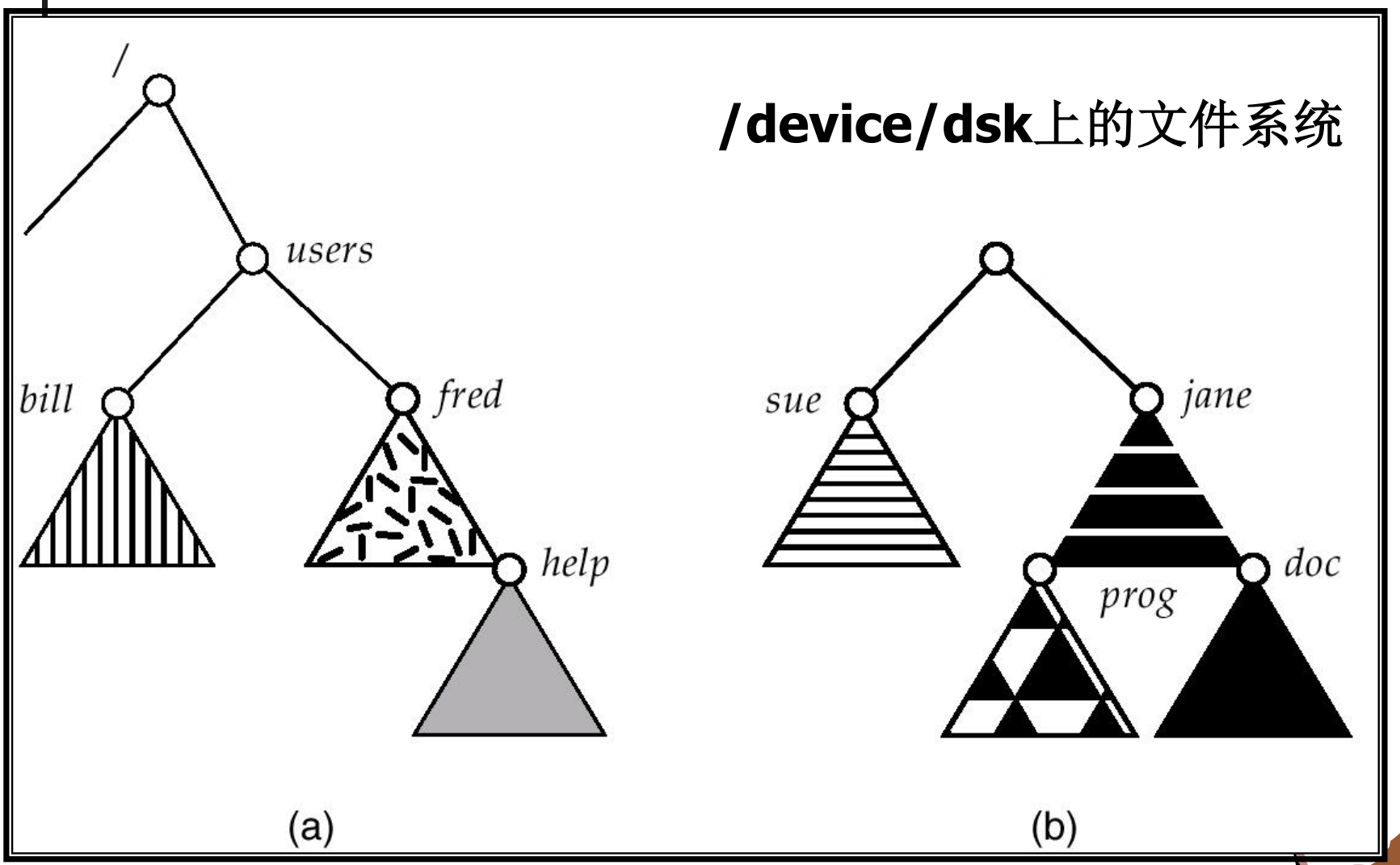
- 文件系统在访问之前必须已安装好。

  A file system must be **mounted** before it can be accessed

- 尚未安装的文件系统必须安装到安装点上

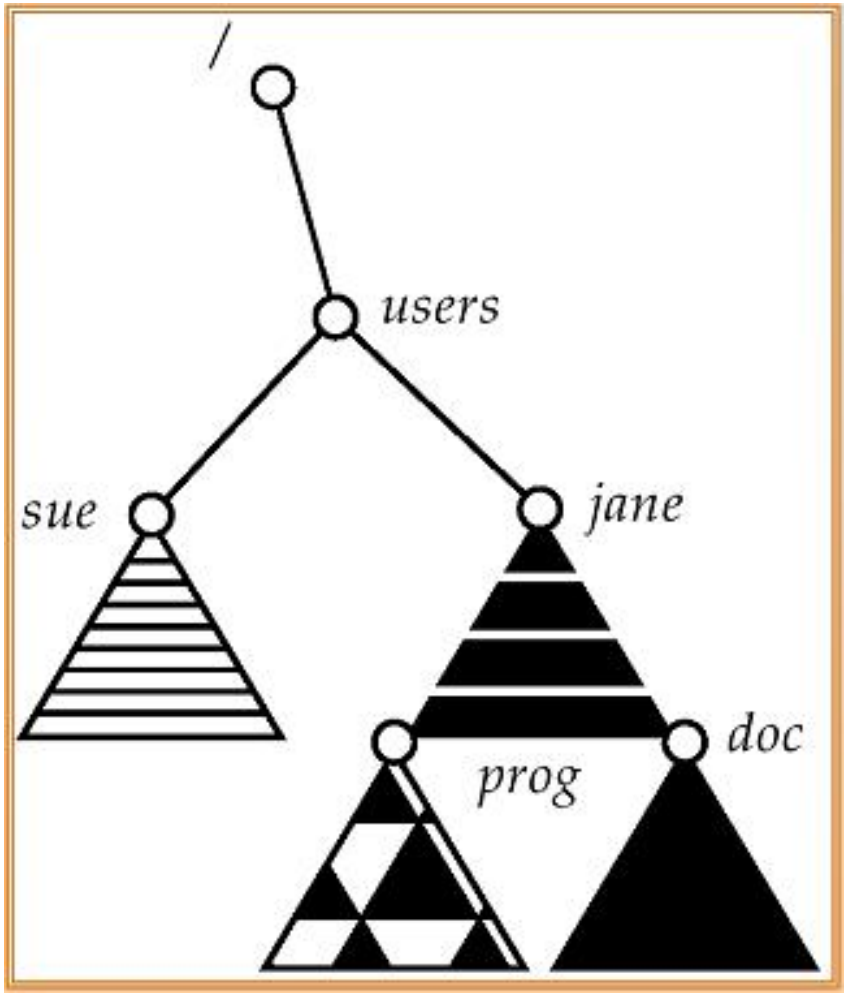
  A unmounted file system is mounted at a mount point.

# (a)已有文件系统 (b)未安装的分区



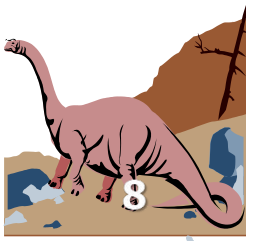
# 安装点 Mount point

**/将device/dsk上的文件系统安装到/usrs**



# 15.4 文件共享 File Sharing

- 文件共享是指不同用户可以共同使用某文件。
- 文件共享的动机是：
  - 用户合作
  - 减少磁盘空间的开销
  - 减少文件的不一致性
- 共享语义：是文件系统对共享文件或目录冲突访问的处理方法。不同共享语义定义了对缓存一致性问题的不同解决方案。



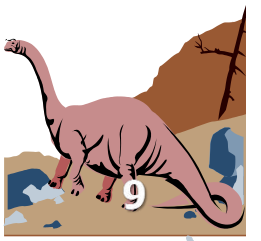
# 一致性语义
# Consistency semantics

- 一致性语义：描述多用户同时访问共享文件时的语义

Consistency semantics: specifies the semantics of multiple users accessing a shared file simultaneously.

- 这些语义规定了一个用户所修改的数据何时对另一个用户可见

In particular, these semantics should specify when modifications of data by one user are observable by other users.



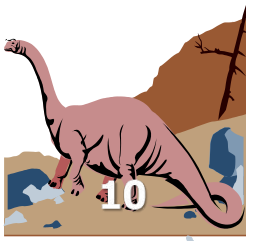
# UNIX 语义 UNIX semantics

- 一个用户对已经打开的文件进行写操作，可以被同时打开同一文件的其他用户所见

  Writes to an open file by a user are visible immediately to other users that have this file open at the same time

- 有一种共享模式允许用户共享文件当前指针位置

  One mode of sharing allows users to share the pointer  of current location into the file

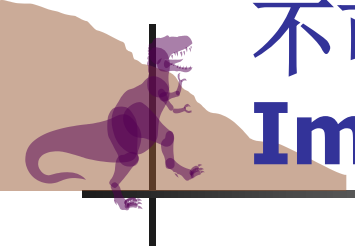


# 会话语义 Session Semantics

- AFS语义　AFS (Andrew File System) Semantics
  - 一个用户对打开文件的写不能立即被同时打开同一文件的其他用户所见。

    **Writes to an open file by a user are not visible immediately to other users that have the same file open simultaneously.**
  - 一旦文件关闭，对其所作的修改只能被以后打开的会话所见。已经打开文件的用户不能看见这些修改。

    **Once a file is closed, the changes made to it are visible only in sessions starting later. Already open instances of the file do not reflect these changes.**



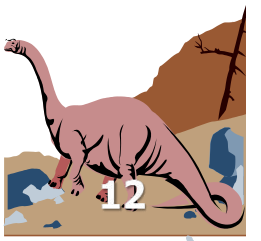
# 不可修改共享文件语义
# Immutable-shared file semantics

- 一旦一个文件被其创建者声明为共享，它就不能被修改。

  Once a file is declared as shared by its creator, it cannot be modified.

- 不可修改共享文件有两个重要特征：文件名不能重用，文件内容不可修改

  An immutable file has two key properties. Its name may not be reused and its contents may not be altered.



# 文件共享的早期实现

- 早期实现文件共享的方法有三种：
  - 绕道法
  - 链接法
  - 基本文件目录表

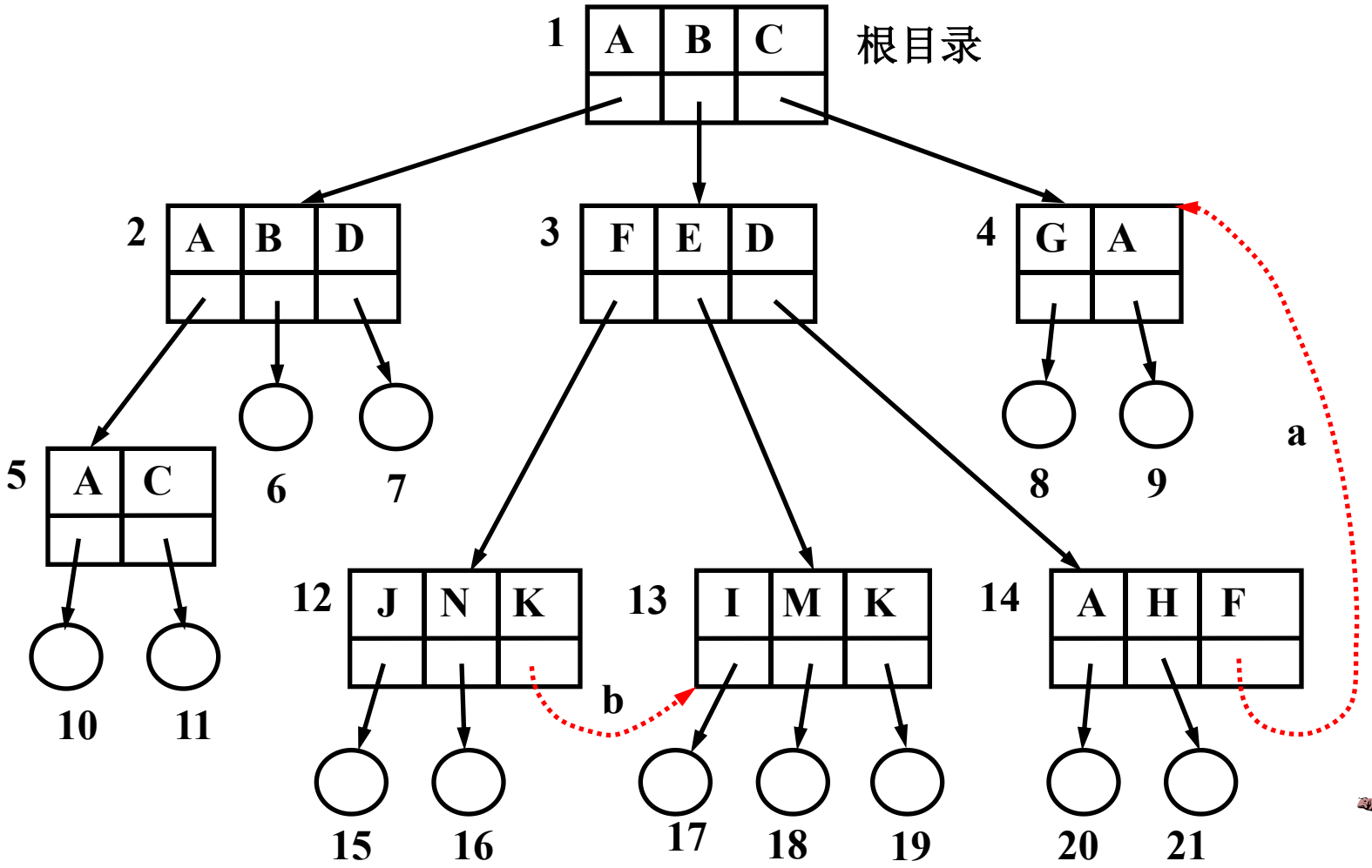
# 绕道法

- 绕道法要求每个用户在当前目录下工作，用户对所有文件的访问都相对于当前目录进行。
- 用户使用相对路径访问文件。当访问文件不在当前目录下时，用户应从当前目录出发向上返回到与所要共享文件所在路径的交叉点，再顺序向下访问到共享文件。
- 因绕道法要绕弯路访问多级目录，从而其搜索效率不高。

# 链接法

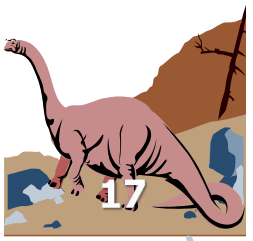
- 链接法将一个目录中的链指针直接指向被共享文件所在的目录。
- 采用链接法实现文件共享时，应在文件说明中增加“连访属性”和“用户计数”两项。前者说明文件物理地址是指向文件还是指向共享文件的目录，后者说明共享文件的用户数目。
- 若要删除一个共享文件，必须判别是否有多个用户共享该文件，若有则只做减1操作，否则才真正删除此共享文件。

# 链接示意图—虚线表示链接



# 基本文件目录表法

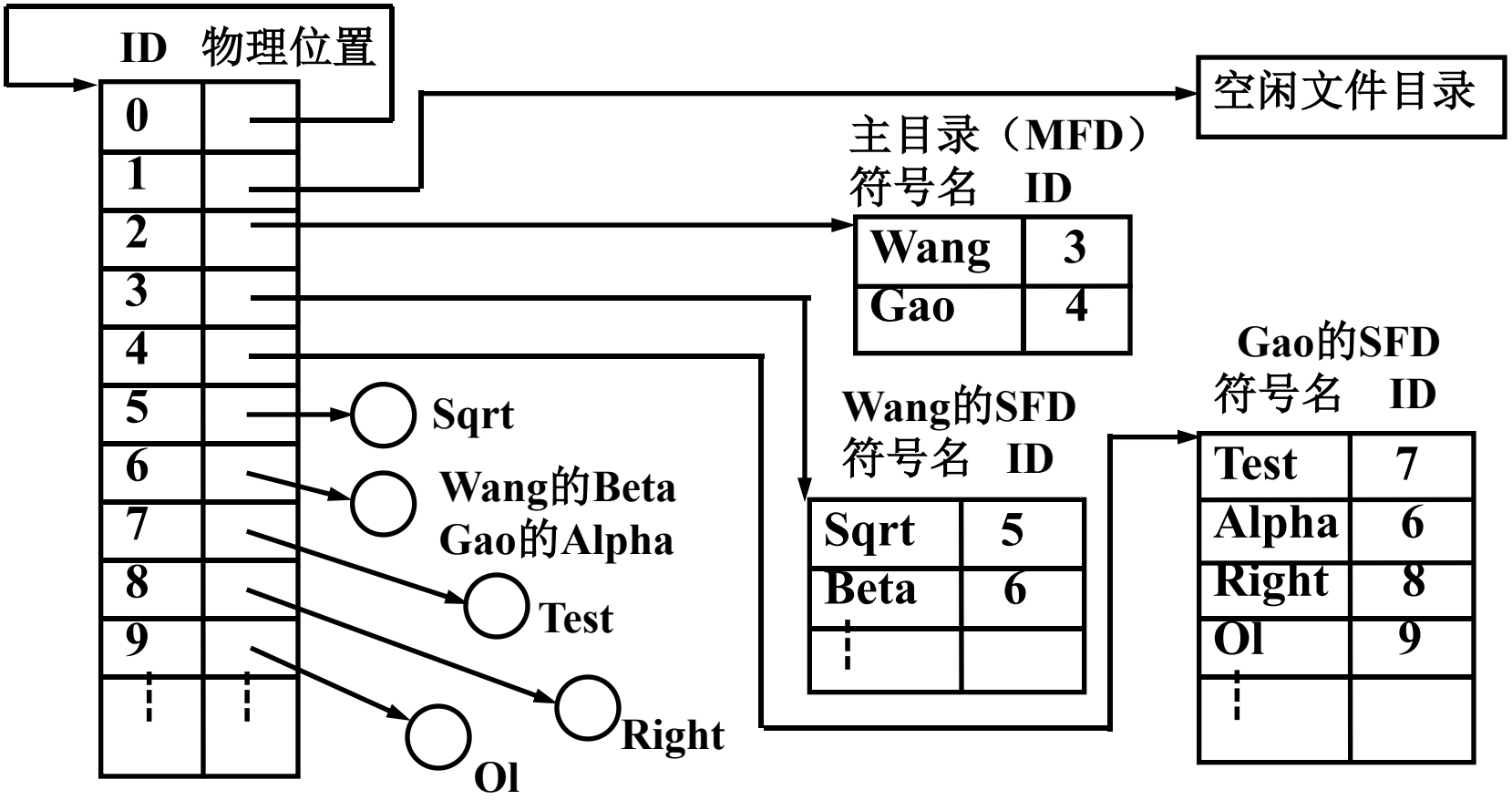
- 基本文件目录表法把所有文件目录的内容分成两部分：
  - 基本文件目录表（BFD）：由文件的属性信息及内部标识符组成。
  - 符号文件目录表（SFD）：由文件符号名和内部标识符组成。



# 特殊标识符

- 在文件系统中通常规定：
  - 0：基本文件目录标识
  - 1：空闲文件目录标识
  - 2：主目录标识符

# 基本文件目录示意图

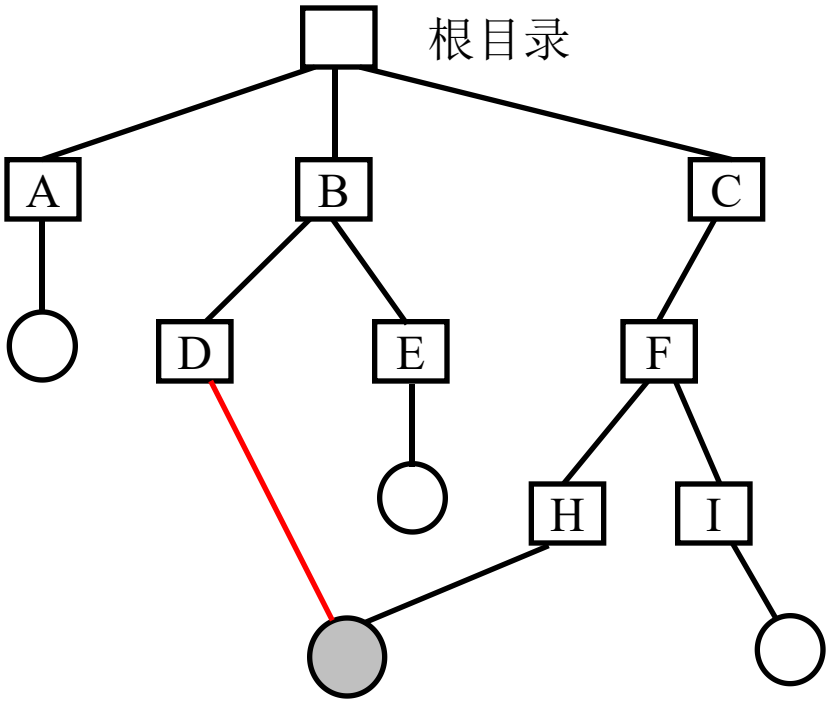

ID 物理位置
0
1
2
3
4
5
6
7
8
9
空闲文件目录
主目录（MFD）
符号名 ID
Wang 3
Gao 4
Sqrt
Wang的Beta
Gao的Alpha
Test
Right
Ol
Wang的SFD
符号名 ID
Sqrt 5
Beta 6
Gao的SFD
符号名 ID
Test 7
Alpha 6
Right 8
Ol 9
基本文件目录

# 用基本文件目录法实现文件共享

- 用基本文件目录法可以方便地实现文件共享。若要共享某个文件，只需在相应的目录文件中增加一个目录项，在其中填上符号名及被共享文件的标识符。
- 如上图中，用户Wang和Gao共享标识符为6的文件，对于系统来说，标识符6指向同一个文件；而对Wang和Gao两个用户来说，则对应于不同的文件名Beta和Alpha。

# 基于索引节点的共享方式

- 当多个用户需要共享文件时，可以将共享文件链接到多个用户的目录中，如右图所示。
- 图中H的一个文件现在也出现在D的目录下，D称为该共享文件的一个链接。



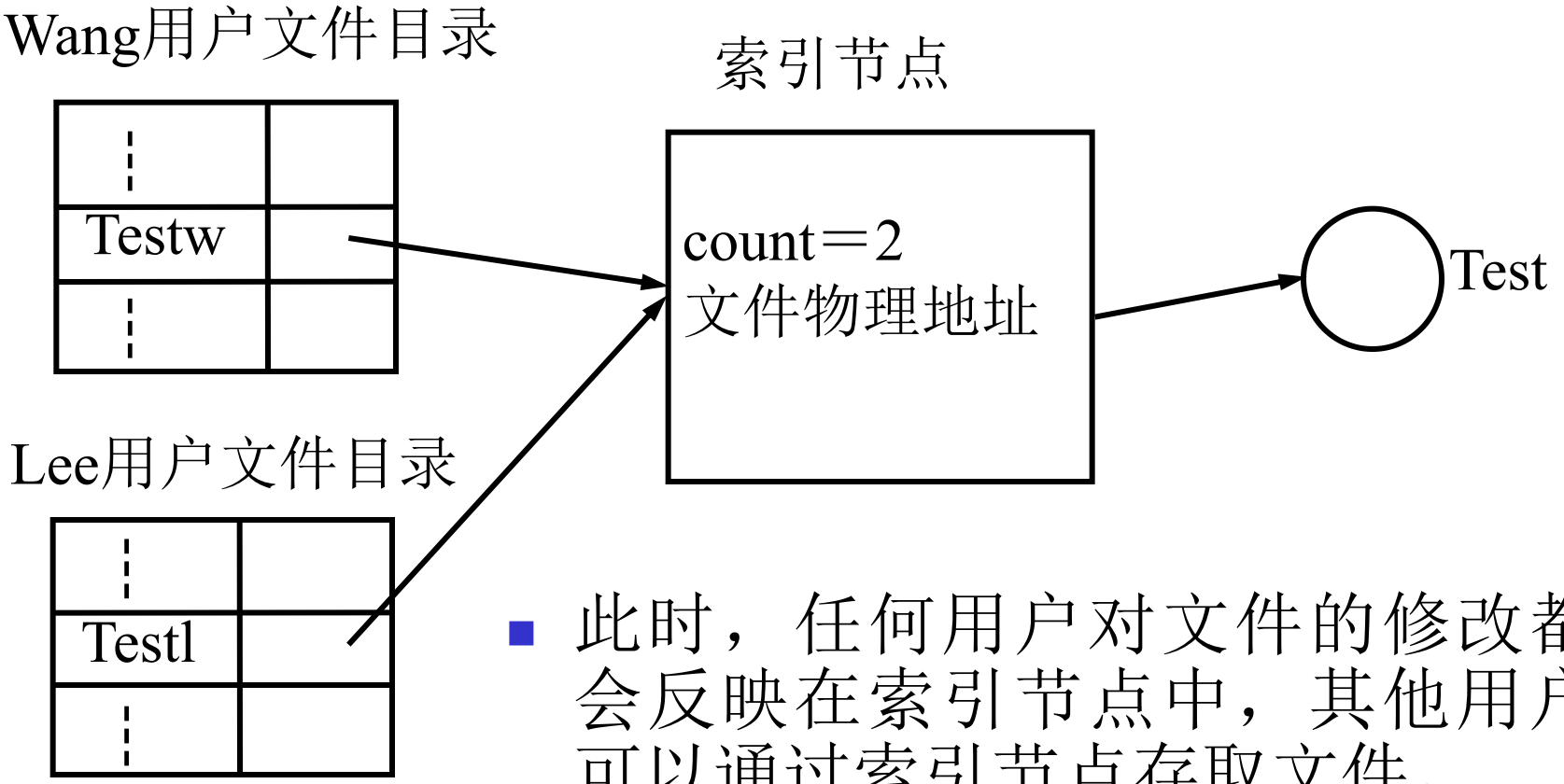
# 文件共享中存在的问题

- 用链接实现文件共享很方便，但也带来一些问题。
- 如目录中包含文件的物理地址，则在链接文件时要将文件的物理地址复制到D目录中。但若随后通过D或H往该文件中添加内容，则新数据块将只会出现在进行添加操作的目录中，这种改变对其他目录而言是不可见的，因而新增加的这部分内容不能被共享。

# 解决办法

- 为了解决这个问题，可以将文件说明中的文件名和文件属性信息分开。
- 索引节点：文件属性信息构成的数据结构，又称i节点。
- 采用这种实现方案，文件目录项仅由文件名和索引节点号构成。
- 引入索引节点后可以减少查找文件的时间开销。

# 基于索引节点的共享示意图




- 此时，任何用户对文件的修改都会反映在索引节点中，其他用户可以通过索引节点存取文件。

# 磁盘索引节点

- 每个文件有一个惟一的索引节点，主要包含：
  - 文件主标识：
  - 文件类型：正规、目录、特别
  - 文件存取权限
  - 文件物理地址
  - 文件长度
  - 文件链接计数：目录树中指向此文件的路径数。
  - 文件存取时间

# 内存索引节点

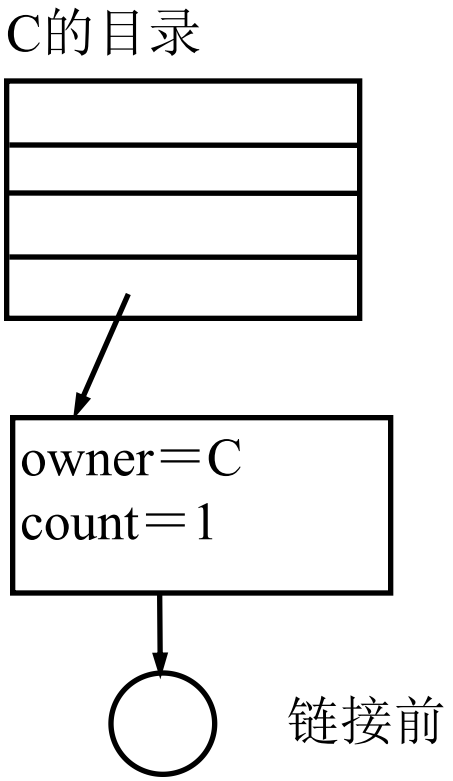
- 文件打开时，要将磁盘索引节点拷贝到内存。内存索引节点除包含磁盘索引节点内容外，还应增加：
  - 索引节点号
  - 状态：索引节点是否上锁、修改
  - 访问计数：正在使用此文件的进程数
  - 文件所属文件系统的逻辑设备号
  - 链接指针：如空闲队列、散列队列

# 索引节点中的链接计数

- 在索引节点中有一个链接计数count字段，用于表示链接到本索引节点的目录项的数目。
- 当count＝2时，表示有两个目录项链接到本文件上。

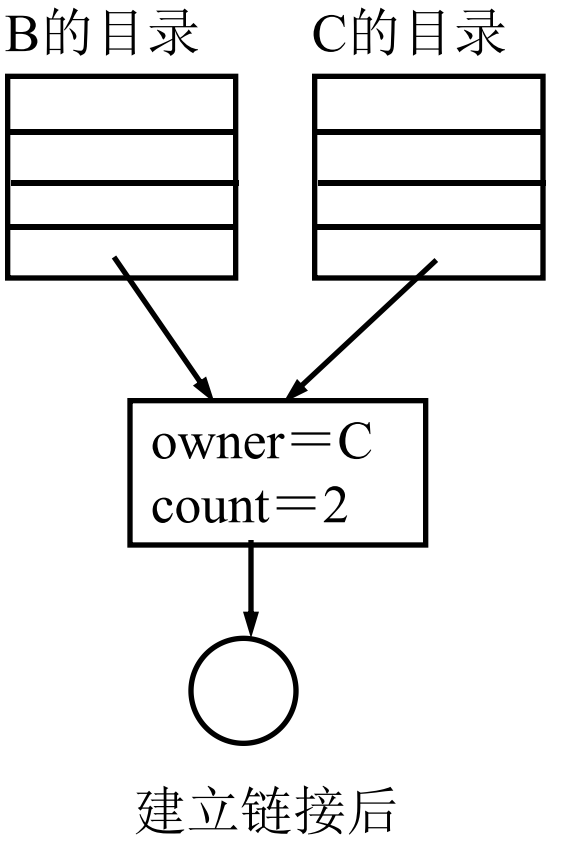
# 链接例--C创建一个新文件

- 当用户C创建一个新文件时，他是该文件的所有者，此时count值为1。



# 链接例-- B链接到C的文件

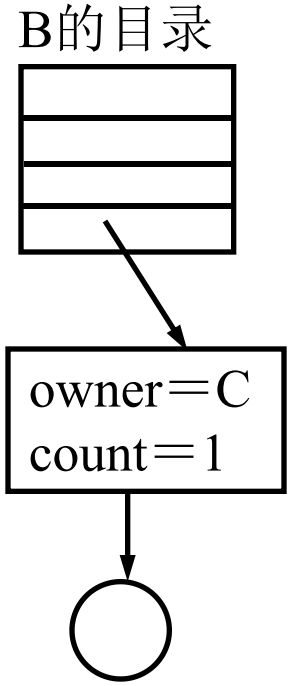
- 当用户B希望共享此文件时，应在用户B的目录中增加一个目录项，并设置指针指向该文件的索引节点，此时文件的所有者仍然是C，但索引节点的链接计数应加1（count＝2）。




建立链接后

# 链接例-- C删除文件

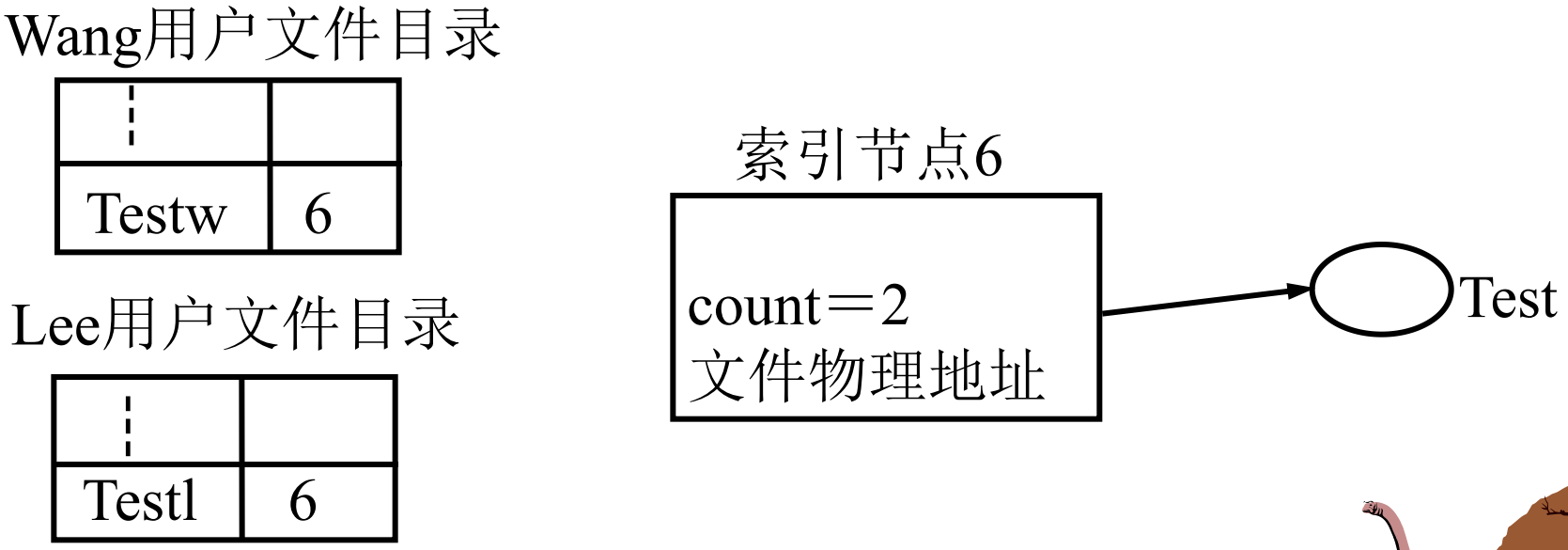
- 如果以后用户C不再需要该文件，则系统只删除C的目录项，并将count减1。
- 此时只有B拥有指向该文件的目录项，而该文件的所有者仍然是C。如果系统进行记账，C将继续为该文件付账。
- 当B不再需要它，count为0，该文作被删除。




拥有者删除文件后

# 硬链接

- 基于索引节点的文件共享方式是通过在不同目录项中设置相同索引节点号来实现的。
- 这种文件的链接方式称为硬链接。
- 硬链接的不足是无法跨越文件系统。

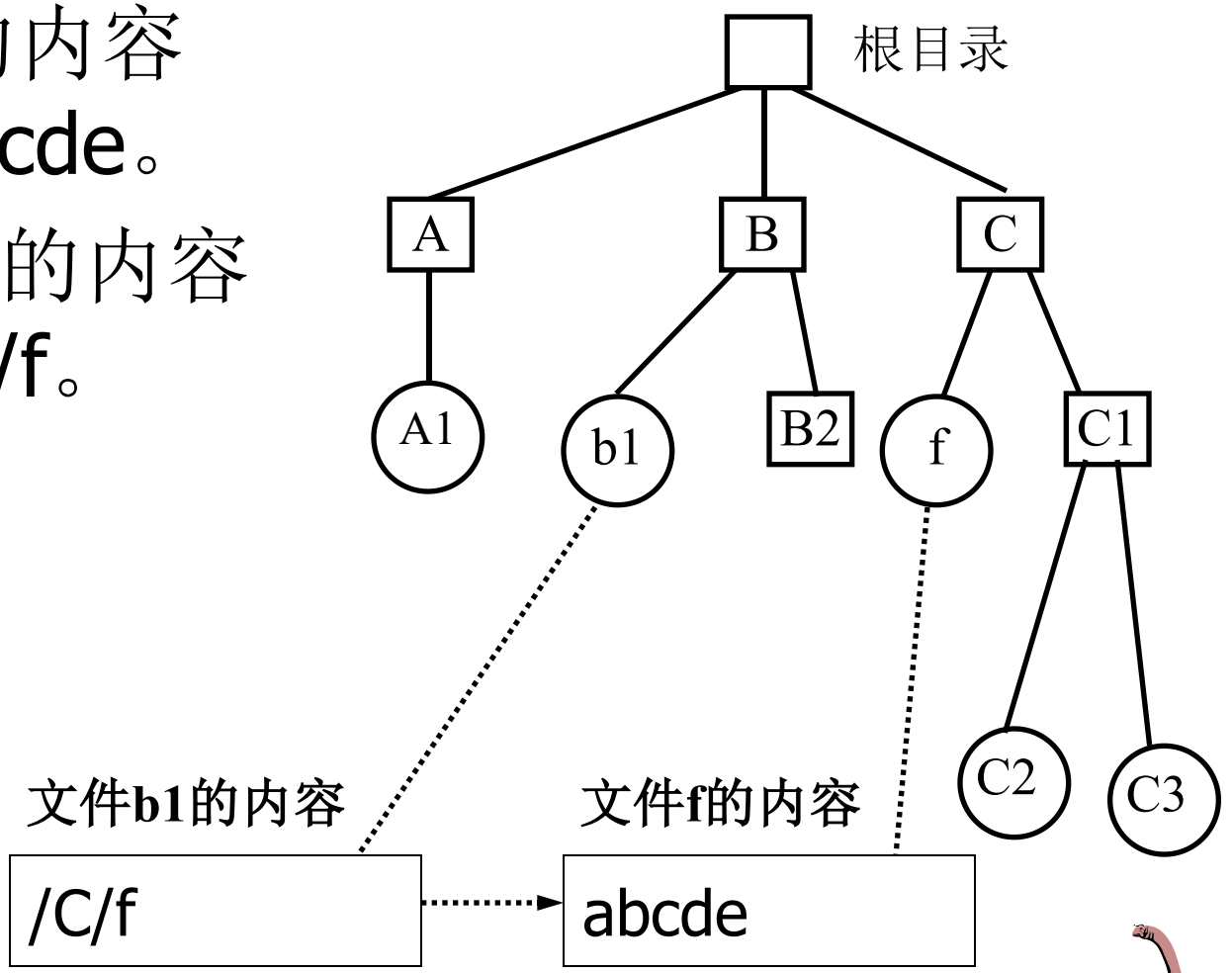


# 利用符号链接实现文件共享

- 利用符号链接也可以实现文件共享。
- 例如，B为了共享C的一个文件f，可以由系统创建一个LINK类型的新文件b1，并把新文件b1添加到B的目录中，以实现B的一个目录b1与文件f的链接。
- 新文件中只包含被链接文件f的路径名，称这种链接方式为符号链接。也称为软链接。

# 符号链接示意图

- 文件f的内容是：abcde。
- 文件b1的内容是：/C/f。

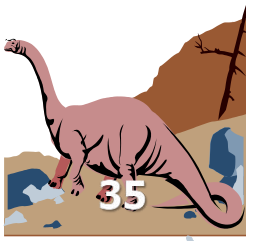


# 文件的访问

- 当用户B要访问被链接的文件f时，操作系统发现要读的文件b1是LINK类型，则由操作系统根据文件b1中的路径名去读该文件，从而实现了用户B对文件f的共享。

# 文件的删除

- 在利用符号链接实现文件共享时，仅文件所有者拥有指向其索引节点的指针，共享该文件的用户只有其路径名，而没有指向索引节点的指针。
- 当文件所有者删除文件后，其他用户若试图通过符号链接访问该文件将导致失败，因为系统找不到该文件，于是系统(可以)将符号链删除。



# 符号链接的特点

- 符号链接的不足是需要额外的开销（根据文件路径名逐个分量进行查找，需要多次访问磁盘）。另外，符号链接需要配置索引节点以及一个磁盘块用于存储路径，这也要消耗一些磁盘空间。
- 符号链接的优点是只要提供一个机器的网络地址以及文件在该机器上的驻留路径，就可以链接全球任何地方的机器上的文件。即可以跨越文件系统。

# 15.5 虚拟文件系统
# Virtual File Systems

- 虚拟文件系统提供一个面向对象的文件系统实现方法

  Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS允许不同类型的文件系统使用相同的系统调用接口

  VFS allows the same system call interface (the API) to be used for different types of file systems(API).
- API是针对VFS的接口，而非对任何特定类型的文件系统

  The API is to the VFS interface, rather than any specific type of file system.

# 虚拟文件系统实现的三个层次

- 顶层：文件系统接口 Top layer: file-system interface
  - **Open, read, write, and close and file descriptors**
- 中间层：VFS The middle layer: VFS
  - 通过定义清晰的**VFS**接口，将文件系统的通用操作与实现分开

    **To separate FS generic operations from their implementation by defining a clean VFS interface**
  - **VFS**基于称为**Vnode**的文件表示结构，该结构包含一个数值标识符以表示网络范围内的唯一文件

    **The VFS is based on a file-representation, called a vnode, that contains a numerical designator for a network-wide unique file.**

# 文件系统实现的三个层次2

- 底层 The bottom layer
  - 不同文件系统实现，如ext3、 NFS

    Various FS implementation：such as Ext3，NFS

# 虚拟文件系统示意图
# Schematic View of a Virtual File System