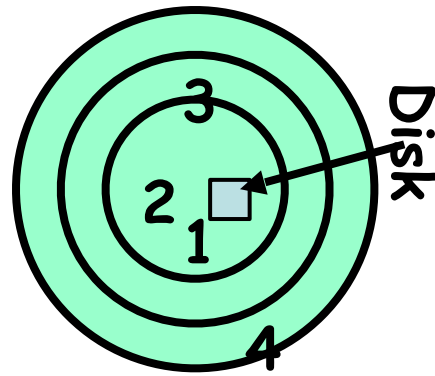


# 文件管理

# 内容

- 基本概念
- 文件结构和存取方法
- 文件目录
- 文件系统的挂载和使用
- 文件共享
- 安全性和可靠性

# 背景



**Where is my mp3?**

# 概述

- 文件系统出现的原因
  - 用户直接操作和管理辅助存储器上信息，繁琐复杂、易于出错、可靠性差
  - 多道程序、分时系统的出现要求以方便、可靠的方式共享大容量辅助存储器
- 文件
  - 文件是由文件名字标识的一组信息的集合。
- 实现按名存取的文件系统的优点：
  - 将用户从复杂的物理存储地址管理中解放出来
  - 可对文件提供各种安全、保密和保护措施
  - 实现文件的共享（同名共享、异名共享）

# 文件系统

- **文件系统：**管理文件的软件即存取和管理信息的机构。它是操作系统中负责存取和管理信息的模块，它用统一的方式管理用户和系统信息的存储、检索、更新、共享和保护，并为用户提供一整套方便有效的文件使用和操作方法。它是管理文件所需要的数据结构（如目录、索引表等）的总体。
- **用户观点：**文件系统呈现在用户面前的是一个文件有什么组成，如何命名，如何保护文件，可以进行何种操作等等
- **操作系统观点：**文件目录怎样实现，怎样管理存储空间，文件存储位置，磁盘实际运作方式(与设备管理的接口)等等。

# 文件系统

## 文件系统面向用户的功能：

- (1) 文件的按名存取；
- (2) 文件目录建立和维护；
- (3) 实现逻辑文件到物理文件的转换；
- (4) 文件存储空间的分配和管理；
- (5) 提供合适的文件存取方法；
- (6) 实现文件的共享、保护和保密；
- (7) 提供一组可供用户使用的文件操作。

## • 文件系统的特点

- (1) 友好的用户接口，用户只对文件进行操作，而不管文件结构和存放的物理位置。
- (2) 对文件按名存取，对用户透明。
- (3) 某些文件可以被多个用户或进程共享。
- (4) 可存储大量信息。

# 文件

- 文件是存贮在某种介质上的（如磁盘、磁带等）并具有文件名的一组有序信息的集合。文件提供了一种把信息保存在存储介质上，而且便于以后存取的方法，用户不必关心实现细节。
- 文件内容的意义：由文件的建立者和使用者解释。
- 文件不但反映了用户概念中的逻辑结构，而且和存放它的辅助存储器的存储结构紧密相关。
- 一个文件必须从逻辑文件和物理文件两个侧面来了解它

# 文件操作

- 基本操作
  - 创建(create):分配存储空间,在目录中创建条目
  - 改写(write)
  - 读取(read)
  - 重定位文件- 文件搜索
  - 删除(delete)
  - 截短(truncate): 保留文件属性, 但长度变为0
- 其它操作
  - 重命名(rename)
  - 拷贝(copy)
  - 扩展(append)



# 文件类型

- **分类的目的：**对不同文件进行管理,提高系统效率；提高用户界面友好性。
- **按文件性质和用途分类：**
  - (1) 系统文件：有关OS及有关系统所组成文件。
  - (2) 用户文件：用户编制的文件。
  - (3) 库文件：标准子程序及常用应用程序组成文件，允许用户使用但不能修改。
- **按信息保存期限分类：**
  - (1) 临时文件；
  - (2) 永久文件；
  - (3) 档案文件。
- **按文件的保护方式分类：**
  - (1) 只读文件；
  - (2) 读写文件；
  - (3) 可执行文件；
  - (4) 不保护文件。
- **按文件的逻辑结构分类：**
  - (1) 流式文件；
  - (2) 记录式文件。

# 文件类型

- 按文件的物理结构分类：

- (1) 连续文件；
- (2) 链接文件；
- (3) 索引文件。

- 按照文件的存取方式分类：

- (1) 顺序存取文件；
- (2) 随机存取文件。



- 按照设备的类型分类：

- (1) 磁盘文件；
- (2) 磁带文件；
- (3) 打印文件。

- 按照文件的内容分类：

- (1) 普通文件：包含的是用户的信息，一般为ASCII或二进制文件。
- (2) 目录文件：管理文件系统的系统文件。
- (3) 特殊文件：字符设备文件：和输入输出有关，用于模仿串行I/O设备，例如终端，打印机，网络等块设备文件：模仿磁盘。

# 文件类型

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, 	libraries of routines for programmers
print or view		ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

❑ 文件名.扩展名

❑ DOS, Windows常见扩展名:

❑ .com .exe

❑ .bat

❑ .asm .c

❑ .obj

❑ Unix, Linux常见扩展名:

❑ .c

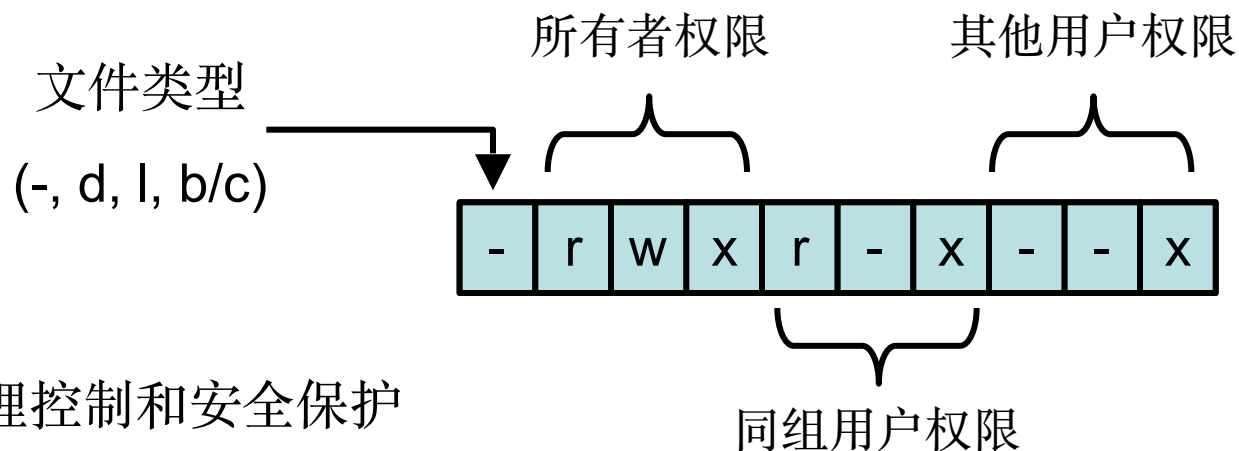
❑ .o

❑ .tar .gz

❑ Macintosh常见扩展名:

❑ store creator

# 文件属性



- 用于文件的管理控制和安全保护
- 分为：
  - 基本属性：文件名、所有者、授权者、长度等
  - 类型属性：普通文件、目录文件、系统文件、隐式文件、设备文件等
  - 保护属性：读、写、可执行、可更新、可删除、可改变保护、归档等
  - 管理属性：创建时间、最后存取时间、最后修改时间等
- **文件保护属性**：用于防止文件被破坏，称为文件保护。包括两个方面：
  - 1.防止系统崩溃所造成的文件破坏:定时转储和多副本
  - 2.防止文件主和其他用户有意或无意的非法操作所造成的文件不安全性：建立三元组（用户、对象、存取权限）

# 内容

- 文件和文件系统的概念
- 文件结构和存取方法
- 文件目录
- 文件系统的挂载和使用
- 文件共享
- 安全性和可靠性

# 与文件组织和存储相关的概念

- **卷**，是物理存储介质的单位，如一盘磁带、一张光盘等。对于存储介质和存储设备可分离的存储器，物理卷和物理设备不总是一致。
- **块**，存储介质上连续信息所组成的一个区域，也叫物理记录，是主存储器与辅助存储器进行信息交换的单位。
- **逻辑记录**，文件中按信息在逻辑上的独立含义划分的一种信息单位，应用程序处理的单位。
- **存储记录**，指附加了操作系统控制信息的逻辑记录，文件管理程序处理的单位。
- **逻辑记录和物理块的关系：**
  - 逻辑记录是按信息在逻辑上的独立含义划分的单位
  - 块是存储介质上连续信息所组成的区域。
  - 一个逻辑记录被存放到文件存储器的存储介质上时，可能占用一块或多块，也可以一个物理块包含多个逻辑记录。
- 书和章节相当于文件和逻辑记录，是逻辑概念；而册和页相当于卷和块，是物理概念。

# 文件的逻辑结构

文件的组织：

- (1) 文件的逻辑结构：是指用户思维中文件的结构。
- (2) 文件的物理结构：是指文件在存储介质上的结构（或称组织）。在当代，文件的存储介质是磁盘，包括软盘、硬盘和光盘、磁带，早期还有磁鼓。由于目前的磁带是模拟磁盘的结构，所以文件的物理结构主要是指磁盘上文件的结构。

逻辑结构：

- (1) **无结构文件**：又称流式文件，组成流式文件的基本信息单位是字节或字，其长度是文件中所含字节的数目，如大量的源程序，可执行文件，库函数等采用的就是流式结构。**无格式限制**。
- (2) **有结构文件**：指由若干个相关的记录构成的文件，又称记录式文件，如数据结构和数据库。**有格式限制**。

例如：学生登记表文件 xsdjb.dbf

姓名	学号	籍贯	通信地址	邮政编码
----	----	----	------	------

李铭	925678	武昌	武昌关山街125号	430074
----	--------	----	-----------	--------

司马乐	925679	北京	北京海军路88号	100034
-----	--------	----	----------	--------

# 文件逻辑结构

- 无结构- 字节(bytes)或者字(words)序列
- 简单的记录结构
  - 行
  - 定长记录
  - 变长记录
- 复杂结构
  - 格式化的文档
  - 可重定位文件
- 选取文件的逻辑结构应遵循如下原则：
  - (1) 当用户对文件信息进行修改操作时，给定的逻辑结构能尽量减少对已存储好的文件信息的变动。
  - (2) 当用户需要对文件信息进行操作时，给定的逻辑结构应使文件系统在尽可能短的时间内查找到需要查找的记录或基本信息单位。
  - (3) 应使文件信息占据最小的存储空间。
  - (4) 应是便于用户进行操作。



# 文件的物理结构

— 几种常见的文件物理结构（空间分配方案）：

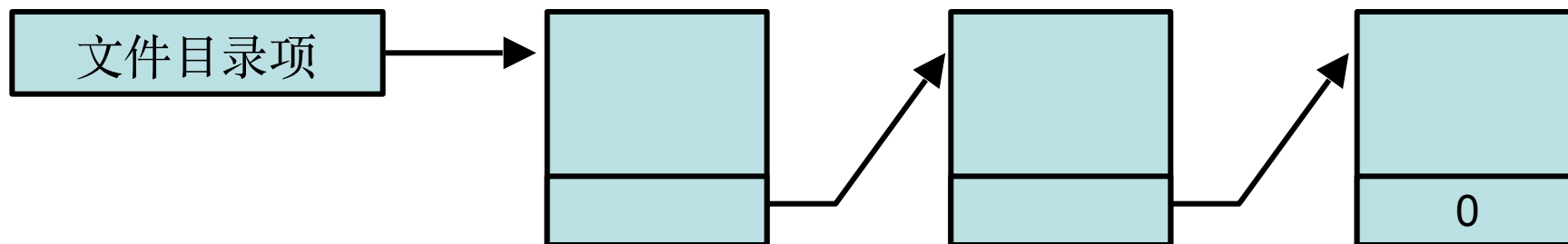
- 顺序文件，连续存储
  - 链接文件
  - 直接文件
  - 索引文件
- } 非连续存储

- 顺序文件

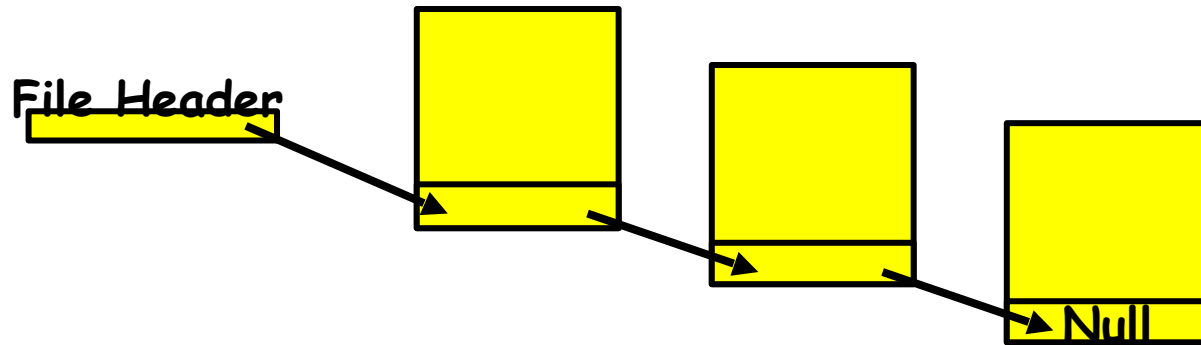
- 逻辑记录连续存储在存储介质的相邻物理块上，逻辑记录顺序和物理记录顺序完全一致
- 几种顺序文件的变种
  - 扩展顺序文件
  - 连接顺序文件
  - 划分顺序文件
- 优点：顺序文件的存取效率是所有逻辑文件中最高的；此外，也只有顺序文件才能存储在磁带上，并能有效地工作
- 缺点：创建时用户须指出文件的大小；交互应用中查找或修改单个记录时，性能可能很差；增加或删除一个记录，都比较困难

## • 链接文件

- 使用链接字或指针来表示文件中各记录之间的串联关系，又称串联文件
- 堆栈、队列、两端队列。
- 特点：逻辑记录顺序独立于物理记录顺序。**优点：**存储空间利用率高；文件创建时用户不必指出文件的大小；文件容易动态扩充和修改；顺序存取效率高。**缺点：**随机存取效率太低，如果访问文件的最后内容，实际上是要访问整个文件。类似于存储管理中的页式。

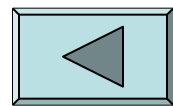
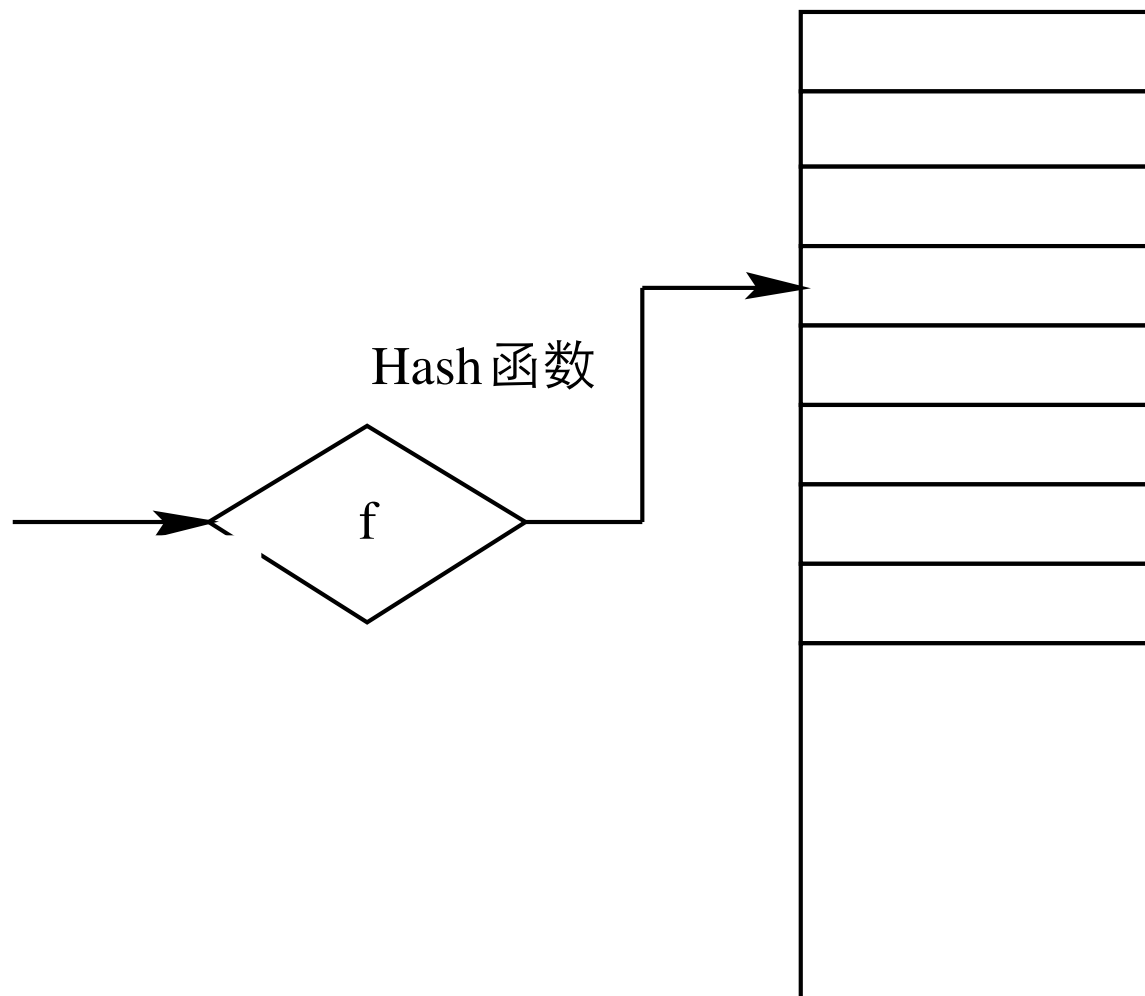


# Linked List Allocation



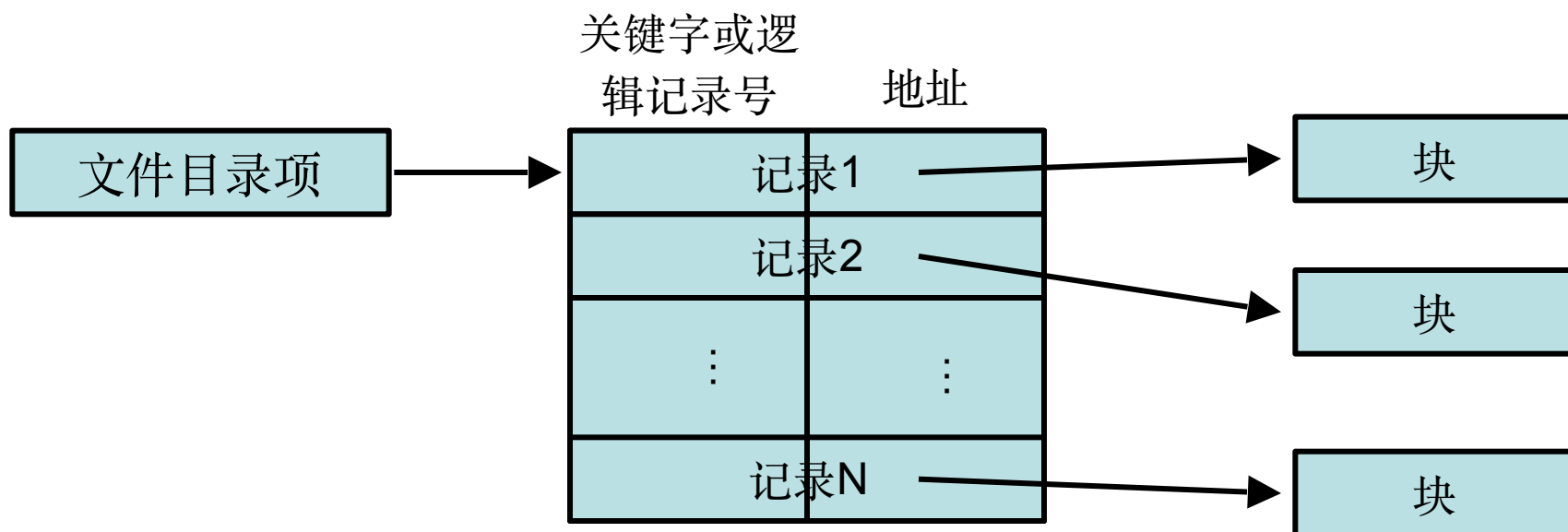
- 直接文件

- 在记录的关键字与其存储的物理地址之间建立某种对应关系（通常采用哈希散列函数），又称（哈希）散列文件
- 关键问题，对应关系的冲突问题
- 直接散列法，将记录键作为记录的存取地址
- 这种由记录键值到记录物理地址的转换被称为键值转换(Key to address transformation)。

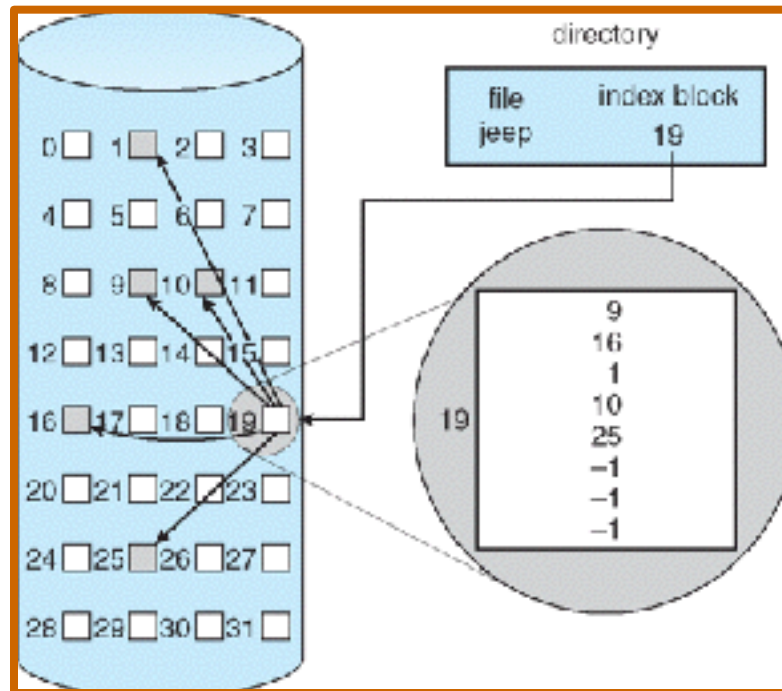


## • 索引文件

- 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张索引表，每个表目包含一个记录键（或逻辑记录号）及对应存储地址。**访问文件时，根据文件逻辑块号查找文件索引表，表中每个表目包括：逻辑块号和物理块号，找到对应的物理块号，然后，进行访问**
- **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；也能充分利用外存空间。**缺点：**寻道次数多和寻道时间长；索引表本身带来了系统开销，如内外存空间，存取时间。
- **索引结构的三种形式：**直接地址结构、索引结构、计算寻址结构

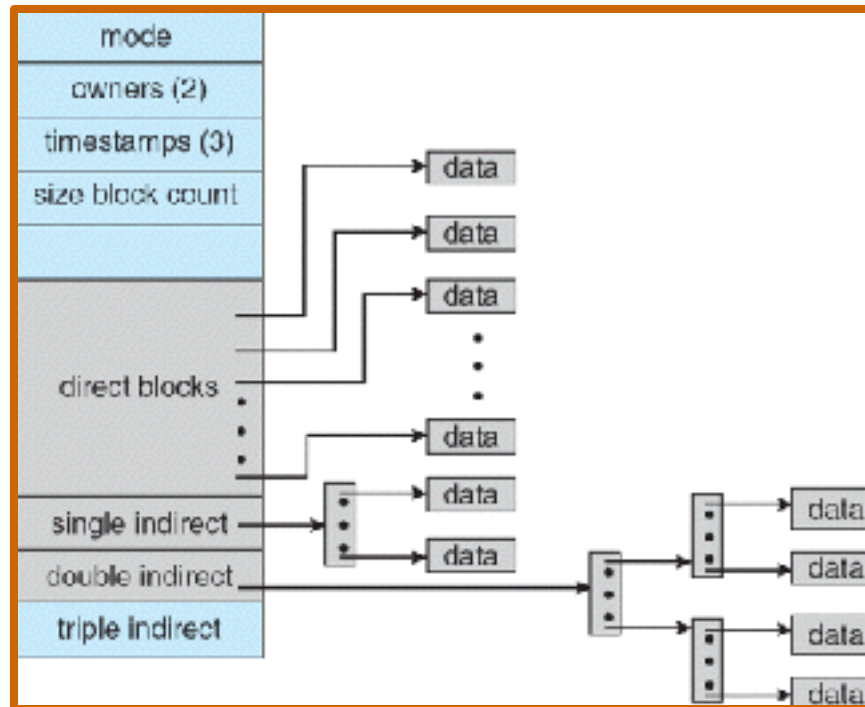


# Indexed Allocation





# Multilevel Indexed Files (UNIX 4.1)



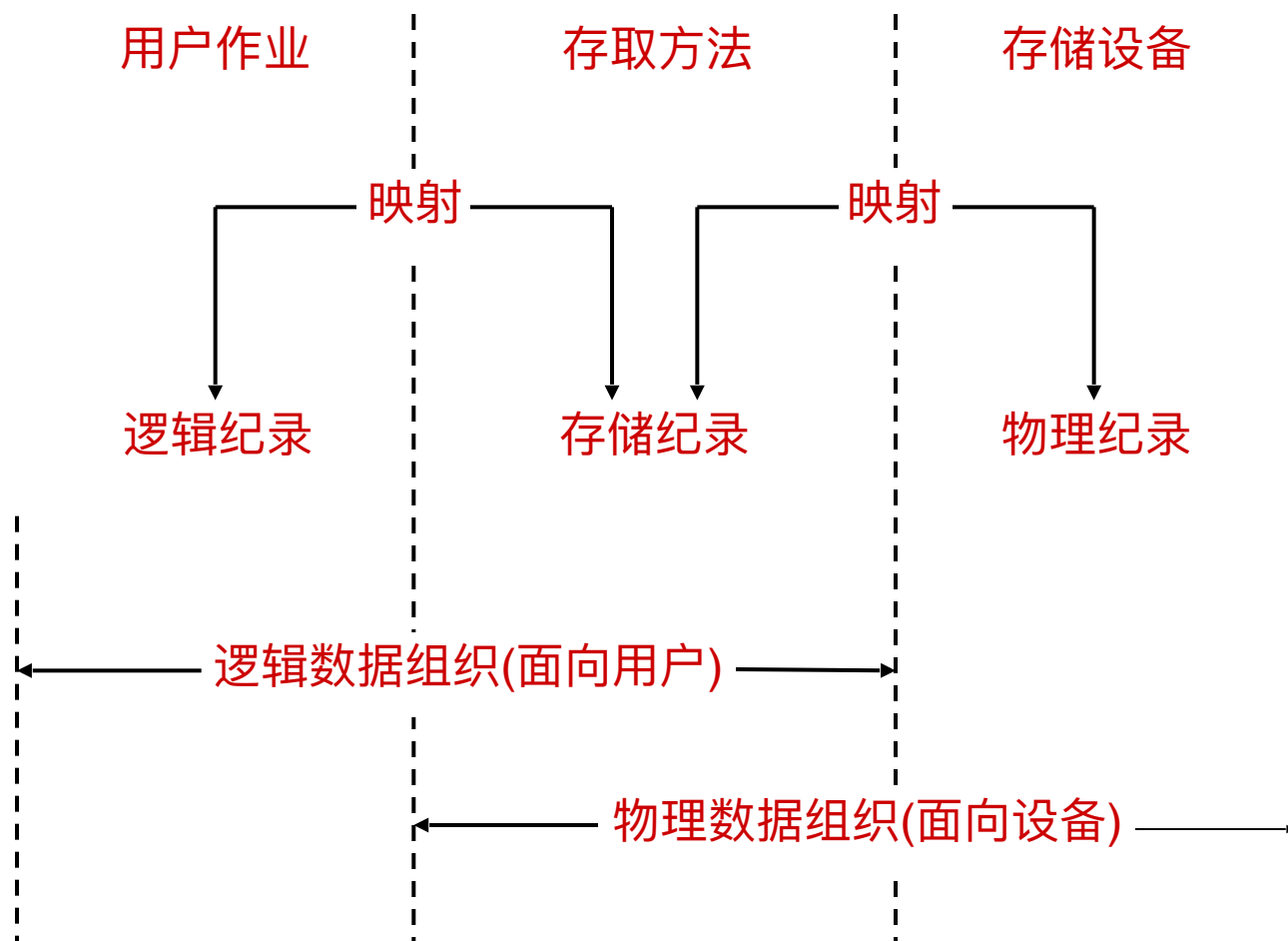
- 索引文件在文件存储器上分两个区：索引区和数据区。
- 访问索引文件需两步操作：
  - (1) 查找文件索引；
  - (2) 以相应键登记项内容作为地址而获得记录数据。
- 两类索引项：稠密索引和稀疏索引。
- 索引顺序文件：是顺序文件的扩展，其中各记录本身在介质上也是顺序排列的，它包含了直接处理和修改记录的能力。
- 索引顺序文件能象顺序文件一样进行快速顺序处理，既允许按物理存放次序（记录出现的次序）；也允许按逻辑顺序（由记录主键决定的次序）进行处理。

- **提高查找速度的办法：**二级索引，高级索引给出低级索引的位置，低级索引给出记录组的位置（若干个记录的索引本身也是一种记录）。查找时先查看高级索引表找到某键所在的索引表区地址，再搜索低级索引表找出数据记录。
- **文件结构、文件存取方式与文件存储介质的关系：**

存储介质	磁带	磁盘		
物理结构	连续结构	连续	链接	索引
存取方式	顺序存取	顺序	顺序	顺序
		随机		随机

- 构造文件物理结构的方法：

- (1) **计算法**：设计映射算法，通过对记录键的计算转换成对应的物理块地址，找到所需记录。直接寻址文件、计算寻址文件，顺序文件均属此类。
- (2) **指针法**：设置专门指针，指明相应记录的物理地址或表达各记录之间的关联。索引文件、索引顺序文件、链接文件等均属此类。



# 存取方法

- (1) **顺序存取**：是最简单的方法。它严格按照文件信息单位排列的顺序依次存取, 后一次存取总是在前一次存取的基础上进行, 所以不必给出具体的存取位置。
  - 在文件系统中, 提供文件存取操作有:  
    `n = read(fd,buffer,size);`  
    `m = write(fd,buffer,size);`  
    这两个操作总是从当前位置开始读或写, 执行顺序存取操作。
- (2) **直接存取 (随机存取)**：在存取时必须先确定进行存取时的起始位置 (如记录号、字符序号等)。
- (3) **索引存取**
  - 基于索引文件的存取方法。(实际系统中, 大多采用多级索引, 以加速记录的查找过程)

# 内容

- 文件和文件系统的概念
- 文件结构和存取方法
- 文件目录
- 文件系统的挂载和使用
- 文件共享
- 安全性和可靠性

# 文件目录

- 什么是文件目录？
  - 是文件系统建立和维护的关于系统的所有文件的清单，每个目录项对应一个文件的信息描述，该目录项又称为文件控制块(FCB)。
- **文件控制块 (FCB)**：又称文件目录项，操作系统为每一个文件开辟一个存储区，在它的里面记录着该文件的有关信息，我们把该存储区称为“文件控制块 (FCB)”找到一个文件的FCB，也就得到了这个文件的有关信息，就能够对它进行所需要的操作。它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。文件控制块也是文件存在的标志。
- 文件控制块的基本内容：
  - 文件存取控制信息，如文件名、用户名、文件主存取权限等
  - 文件结构信息，文件逻辑结构、文件的物理结构等
  - 文件使用信息，已打开该文件的进程数、文件的修改情况等
  - 文件管理信息，文件建立日期、文件访问日期等

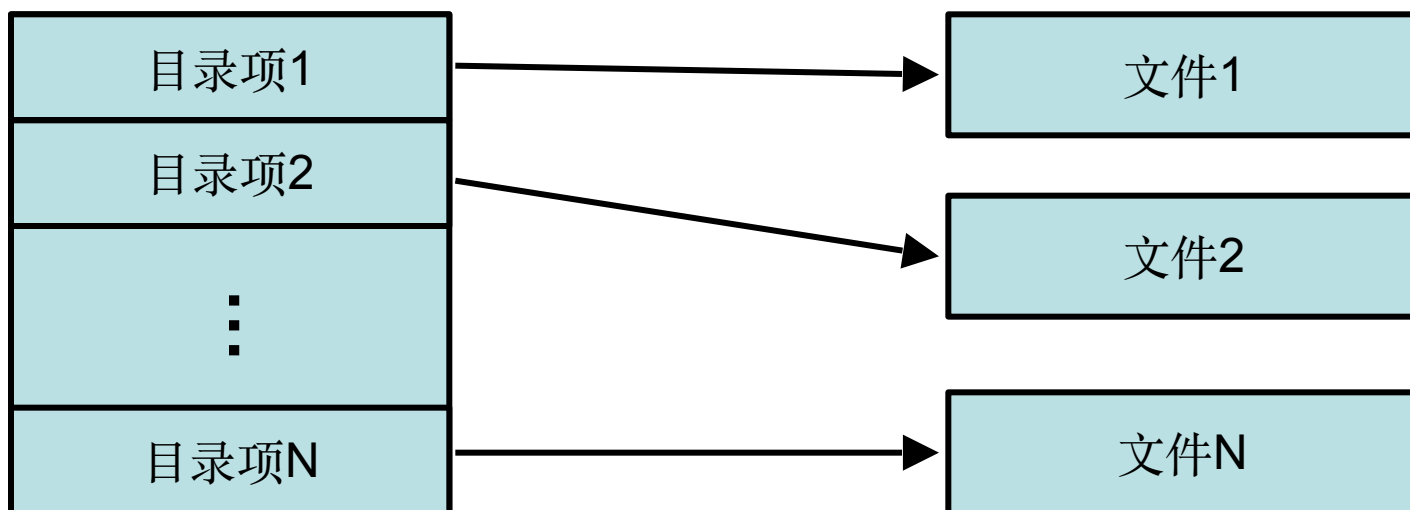


# 文件控制块(FCB)

文件名称	
文件在辅存中的起始物理地址	
逻辑记录长	逻辑记录个数
文件主的存取权限	
其他用户的存取权限	
⋮	
⋮	
文件建立的日期和时间	
上次存取的日期和时间	

# 文件目录结构

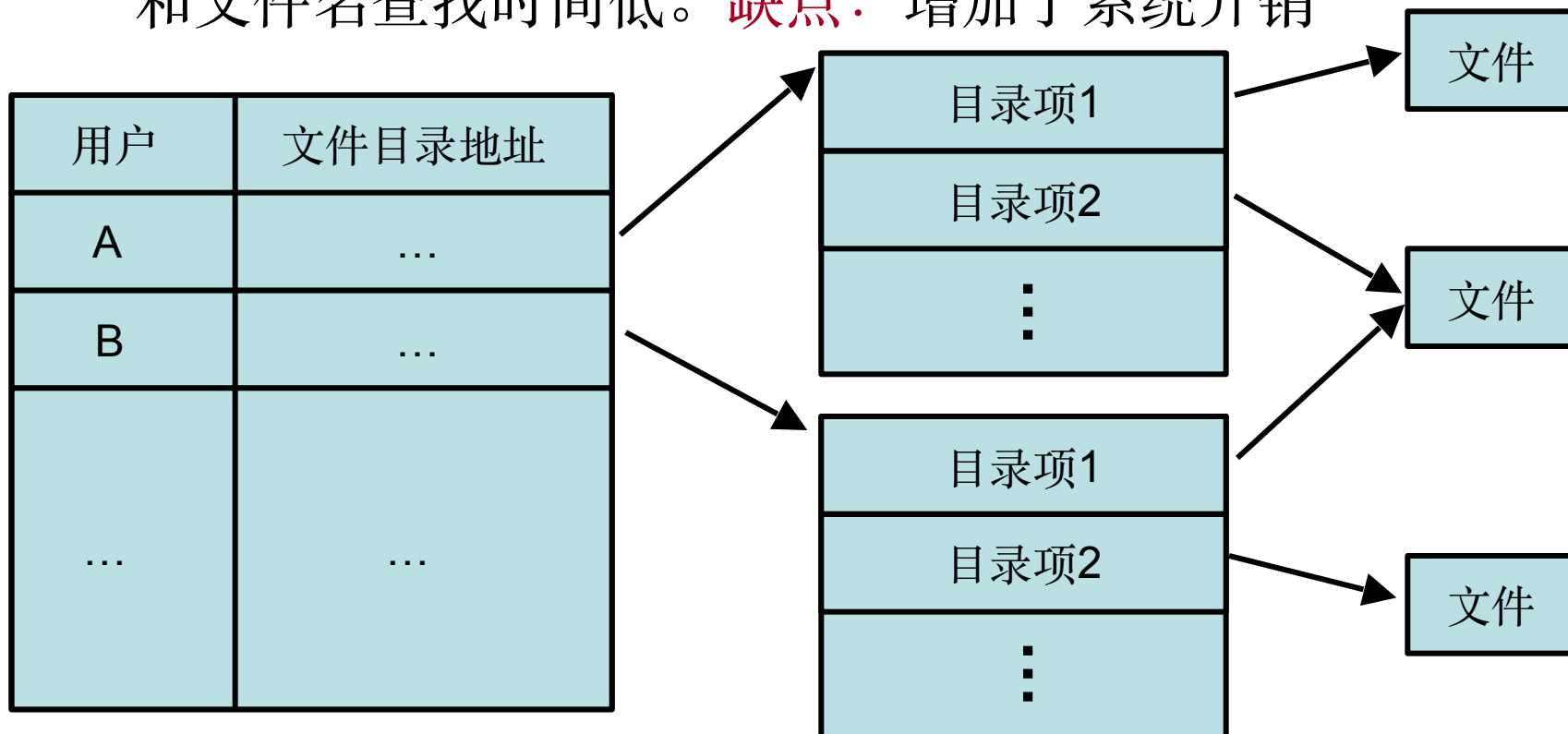
- 一级目录结构
  - 在操作系统中构造一张线性表，与每个文件有关的属性占用一个目录项就构成一级目录结构。
  - 优点: 简单，易实现
  - 缺点:
    - 重名问题（当多用户共享一个目录时）
    - 难以实现文件共享（异名共享问题）
    - 文件平均检索时间长



- 二级目录结构

- 文件目录由两级目录构成，第一级为主文件目录用于管理所有用户文件目录，第二级为用户的文件目录，用于管理每个用户下的文件。

- **优点：**解决了文件的重名问题和文件共享问题；用户名和文件名查找时间低。**缺点：**增加了系统开销



- 树形目录结构

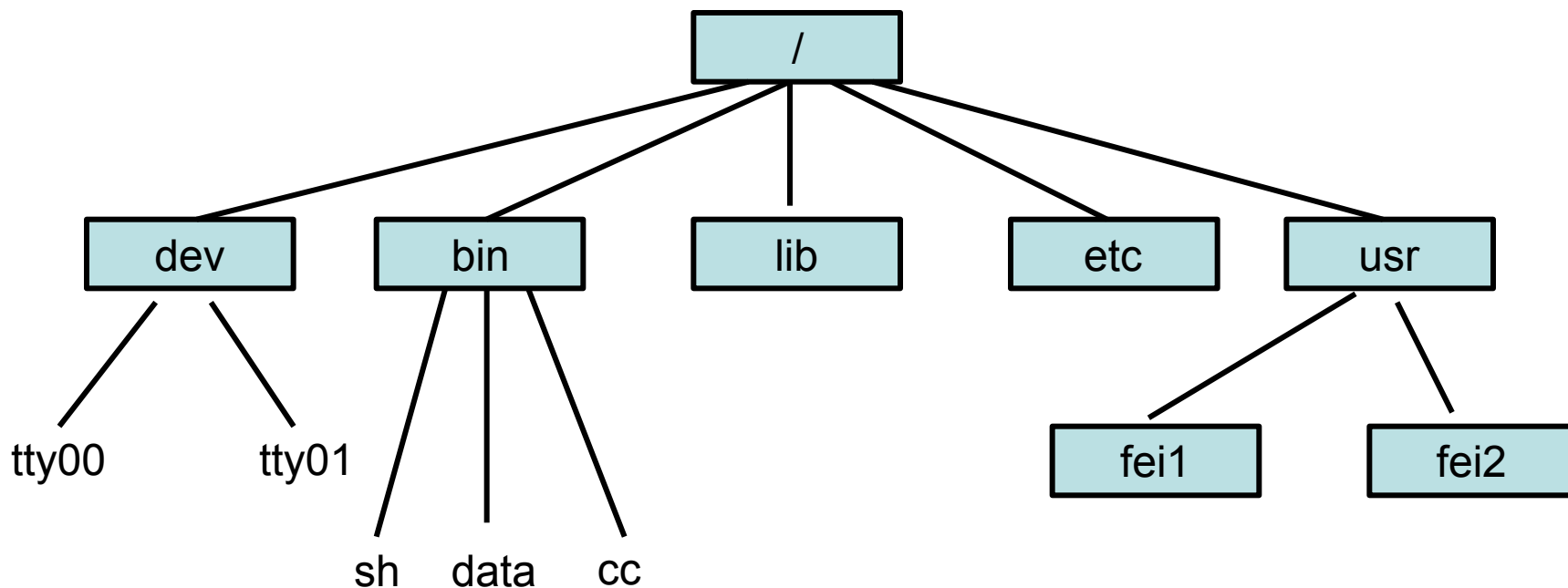
- 二级目录结构的推广成多级目录结构，该结构为一倒置的有根树，又称树形目录结构。每一级目录可以是下一级目录的说明，也可以是文件的说明，形成层次关系
- 倒置树的根称为根目录，从根向下，每一个树枝为一个子目录，而树叶则为文件。
- 优点：
  - 较好地反映现实世界数据集合之间的层次关系
  - 不同文件可以重名，只要不在同一个目录中
  - 容易以目录为单位进行文件的保护、保密和共享
- 缺点：
  - 查找一个文件按路径名逐层检查，由于每个文件都放在外存，多次访盘影响速度。

# 文件目录

- 树形目录结构

- 文件的全名，应该从根目录开始，到该文件名为止，目录路径+文件名。

- 例，/user/include/testfile.c





- **文件目录检索：**根据路径名进行检索，全路径名检索：从根开始；相对路径检索：从当前路径。
  - **绝对路径名：**一个文件的路径名是由根目录到该文件的通路上所有目录文件名和该文件的符号名组成的。DOS和WINDOWS文件路径名：E:\Program Files\OS
  - **相对路径名：**对于文件又有一种相对路径名。用户可以指定一个目录作为当前目录（也称工作目录）。从当前目录往下的文件的路径名，称为文件的相对路径名。一个文件的相对路径名与当前所处的位置有关，它不是惟一的。

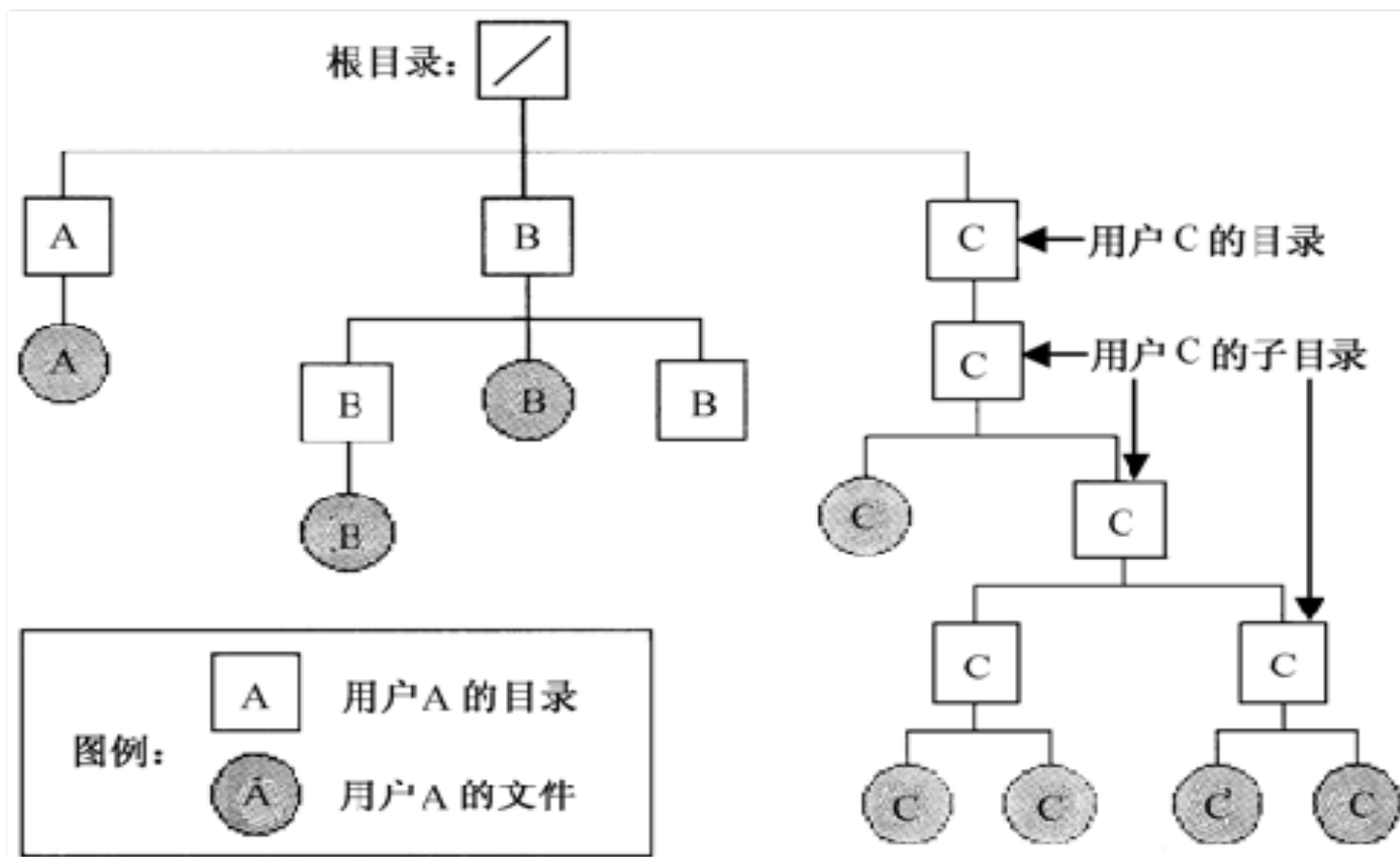


图 5-1-1 用户目录树



# 如何实现“按名存取”？

- 当用户要求存取某个文件时，系统查找目录文件，获得对应的文件目录。
- 在文件目录中，根据用户给定的文件名寻找到对应该文件的文件控制块（文件目录项）
- 通过文件控制块所记录的该文件的相关信息（如文件信息存放的相对位置或文件信息首块的物理位置）依次存取该文件的内容。

# 文件存取

- UNIX 的 “inode”
  - 每个inode对应一个唯一的inumber
  - 在Unix操作系统中的任何资源都被当作文件来管理
  - 在这个结点中存储了这个文件的大部分属性
- Question: 如何存取一个指定的文件?
  - 用户给出 inode 的编号 (inumber) .
    - Imagine: open("14553344")
  - 用户给出文件名
    - Have to map name→inumber
  - 图标
    - GUI: This is how Apple made its money. Graphical user interfaces. Point to a file and click.
- 按名存取: from user-visible names to system resources
  - In the case of files, from strings (textual names) or icons to inumbers/inodes
  - For global file systems, data may be spread over globe⇒need to translate from strings or icons to some combination of physical server location and inumber

# 内容

- 文件和文件系统的概念
- 文件结构和存取方法
- 文件目录
- 文件系统的挂载和使用
- 文件共享
- 安全性和可靠性

# 文件系统的使用

- 文件操作的实现: 文件系统提供对文件的各种操作, 这些操作方便、灵活地使用文件及文件系统, 形式分别为系统调用或命令。文件系统提供给用户程序的一组系统调用, 包括: 建立、打开、关闭、撤销、读、写和控制, 通过这些系统调用用户能获得文件系统的各种服务。
- 用户通过两类接口与文件系统联系, 获得文件系统的服务:
  - (1) 第一类是操作或控制台命令, 构成文件系统人—机接口, 如DOS命令: `dir`, `cd`等。
  - (2) 第二类是提供给用户程序使用的文件类系统调用, 构成了用户和文件系统的另一个接口。
    - 基本文件类系统调用:
      - 建立文件
      - 打开文件
      - 读/写文件
      - 文件控制
      - 关闭文件
      - 撤消文件
      - ...

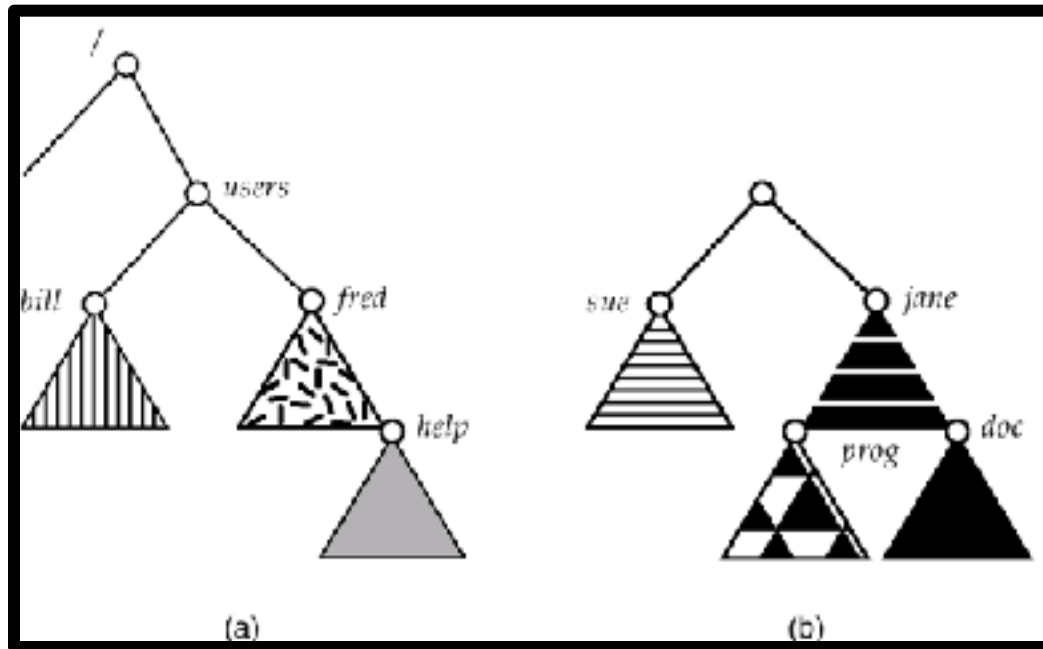
# 文件系统的挂载

- 文件卷，又称文件子系统，存放文件和目录信息，也存放文件属性、空闲区域信息。通常，存储介质的物理单位为一个卷。（硬盘的一个分区为一个卷）
  - Windows/Dos系统中，不需用户显式地进行文件卷挂载操作。
  - UNIX/Linux系统中，每个文件卷需要安装才能使用。文件系统可分为基本文件系统和可装卸地子文件系统两部分。(mount操作)

# 文件系统的挂载

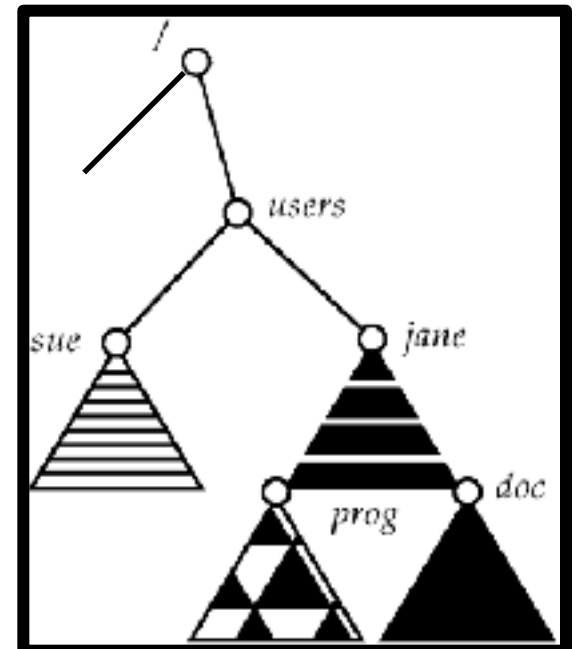
- 一个文件系统必须首先安装/挂载（mount），然后才能够被访问
  - 在多个分区上创建目录结构
- 未安装的文件系统将在安装点被安装，安装点即目录结构存放的位置
  - 例如
    - `mount /dev/hda5 /mnt/diskD`
    - `umount /dev/hda5`
    - `umount /mnt/diskD`

# 示例(Linux)



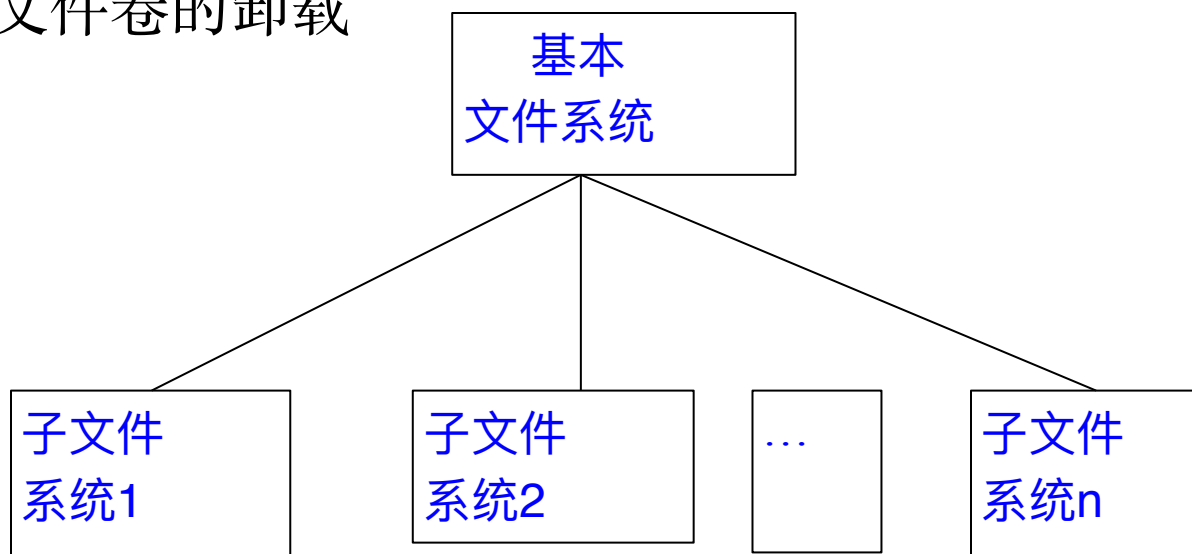
existing

unmounted partition



mount point

- 文件卷的挂载和使用：
  - (1) 文件系统结构：
  - (2) 文件卷的动态挂载
  - (3) 文件卷的动态挂载过程
  - (4) 文件卷的卸载





- 文件共享

- 文件共享是指不同用户（进程）间共同使用同一个文件。

- 共享的三种形式：

- 被多个用户使用，由存取权限控制；
      - 被多个程序使用，但各用自己的读写指针；
      - 被多个程序使用，但共享读写指针。

- 文件共享还可以节省大量的外存空间，有效减少文件复制而增加的访问外存次数，进程间通过文件交换信息

- UNIX系统中常见的共享文件方式：

- 静态共享
    - 动态共享
    - 符号链接共享

- 一致性：一个用户对数据的修改是否可以、何时可以被其它用户观察到

- **文件的静态共享：**系统调用形式为：char \* oldnamep, \* newnamep; link (oldnamep, newnamep)
  - ① 检索目录找到oldnamep所指向文件的索引节点inode编号。
  - ② 再次检索目录找到newnamep所指文件的父目录文件，并把已存在文件的索引节点inode编号与别名构成一个目录项，记入到该目录中去。
  - ③ 把已存在文件索引节点inode的连接计数i\_nlink加“1”。链接实际上是共享已存在文件的索引节点inode，完成链接的系统调用：`link("/usr/fei1/myfile.c", "/usr/fei2/myfile.c");`      `link("/usr/fei1/myfile.c", "/usr/include/testfile.c")`

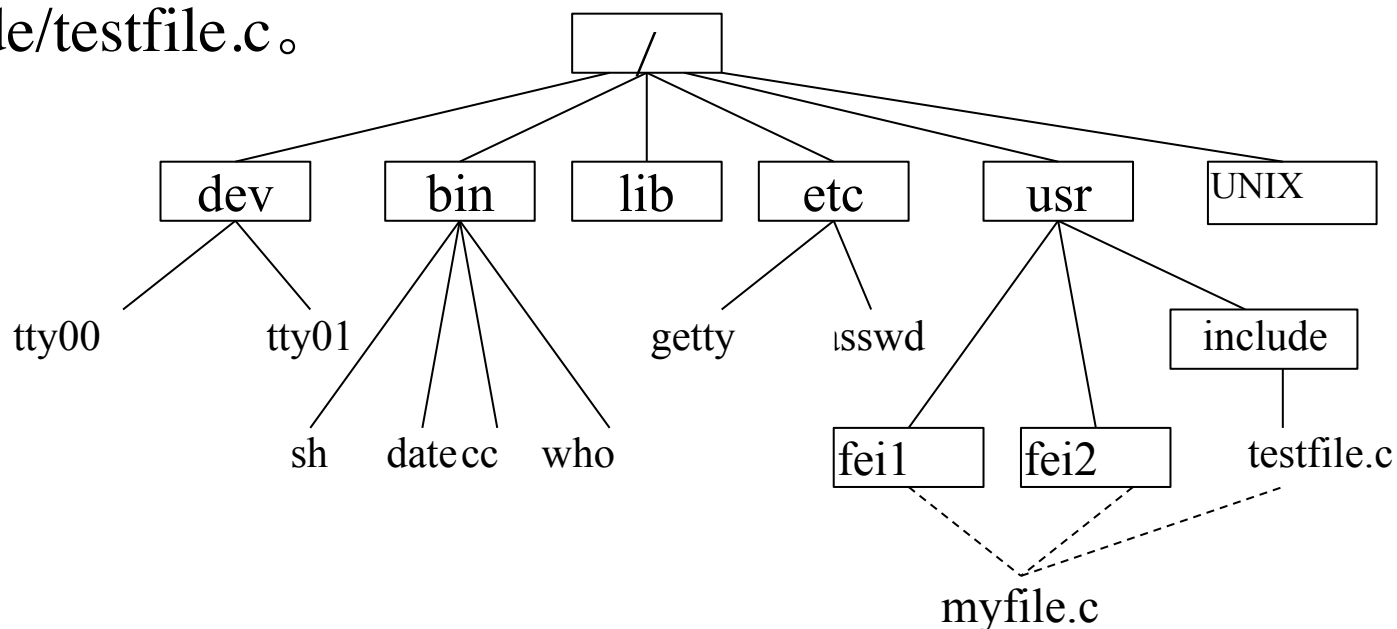
文件解除链接调用形式为：unlink (namep)。解除链接与文件删除执行的是同一系统调用代码。删除文件是从文件主角度讲的，解除文件连接是从共享文件的其他用户角度讲的。都要删去目录项，把i\_nlink减“1”，不过，只有当i\_nlink减为“0”时，才真正删除文件。

执行后，三个路径名指的是同一个文件：

`/usr/fei1/myfile.c`,

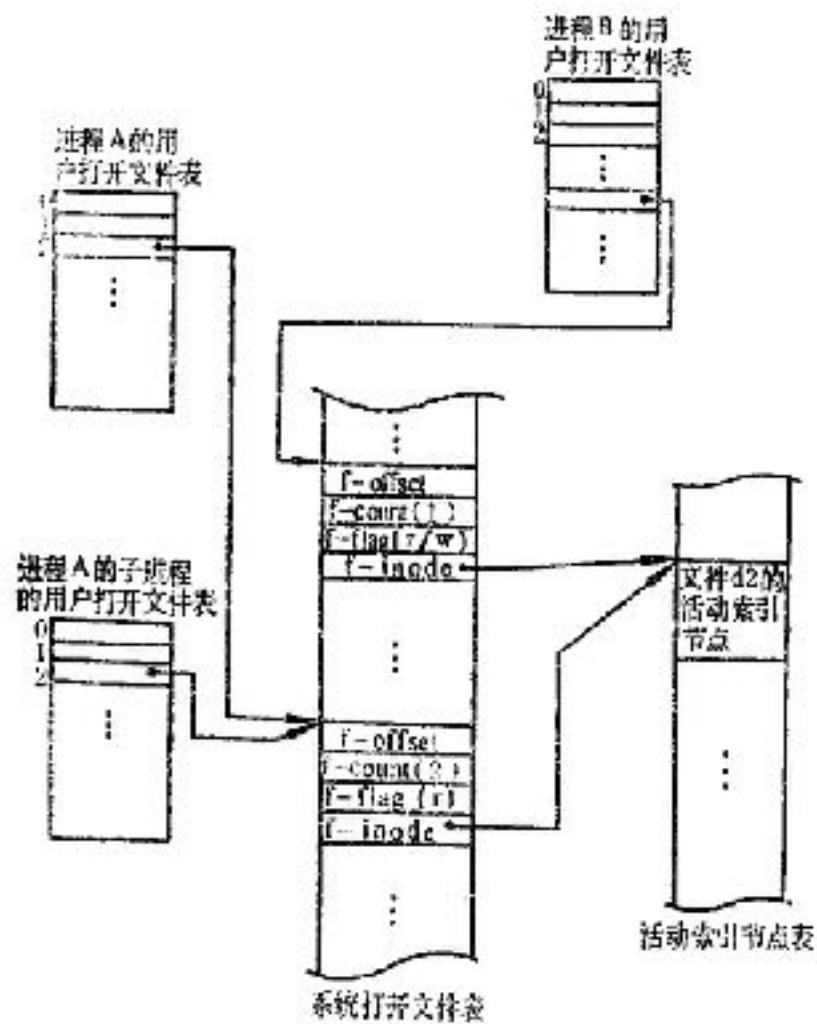
`/usr/fei2/myfile.c`,

`/usr/include/testfile.c`。



# 文件的动态共享

- 1.文件动态共享是系统中不同的用户进程或同一用户的不同进程并发地访问同一文件。
- 2.这种共享关系只有当用户进程存在时才可能出现，一旦用户的进程消亡，其共享关系也就自动消失。
- 3.UNIX文件的每次读写由一个读/写位移指针指出要读写的位置。问题是：若文件可以为多个进程所共享，那么，应让多个进程共用同一个读/写位移指针，还是各个进程具有各自的读写位移指针呢？
- 4.同一用户父、子进程协同完成任务，使用同一读/写位移指针，同步地对文件进行操作。
- 5.该位移指针宜放在相应文件的活动索引节点中。当用系统调用fork建立子进程时，父进程的user结构被复制到子进程的user结构中，使两个进程的打开文件表指向同一活动的索引节点，达到共享同一位移指针的目的。
- 6.两个以上用户共享文件，每个希望独立地读、写文件，这时不能只设置一个读写位移指针，须为每个用户进程分别设置一个读、写位移指针。
- 7.位移指针应放在每个进程用户打开文件表的表目中。这样，当一个进程读、写文件，并修改位移指针时，另一个进程的位移指针不会随之改变，从而，使两个进程能独立地访问同一文件。



# 虚拟文件系统

## — 目标：

- 同时支持多种文件系统
- 系统中可安装多个文件系统
- 为用户提供一致的接口
- 提供网络共享文件支持
- 可扩充新的文件系统

## — 基本思路：

- 对多个文件系统的共同特征进行抽象，形成一个与具体文件系统无关的虚拟层，并在此层上定义对用户的一致性接口。
- 在文件系统具体实现层使用类似开关表技术进行文件系统的转接，每个文件系统是自包含的，实现各文件系统的具体细节。

# 文件系统的保护

**文件系统的可靠性:**系统抵抗和预防各种物理性破坏和人为性破坏的能力。

1.坏块问题。

2.备份：通过转储操作，形成文件或文件系统的多个副本。

1)最简单的RAID（廉价磁盘冗余阵列）组织方式：镜像

2)最复杂的RAID组织方式：块交错校验。

3)海量转储：定期将所有文件拷贝到后援存储器。

4)增量转储：只转存储修改过的文件，即两次备份之间的修改，减少系统开销。

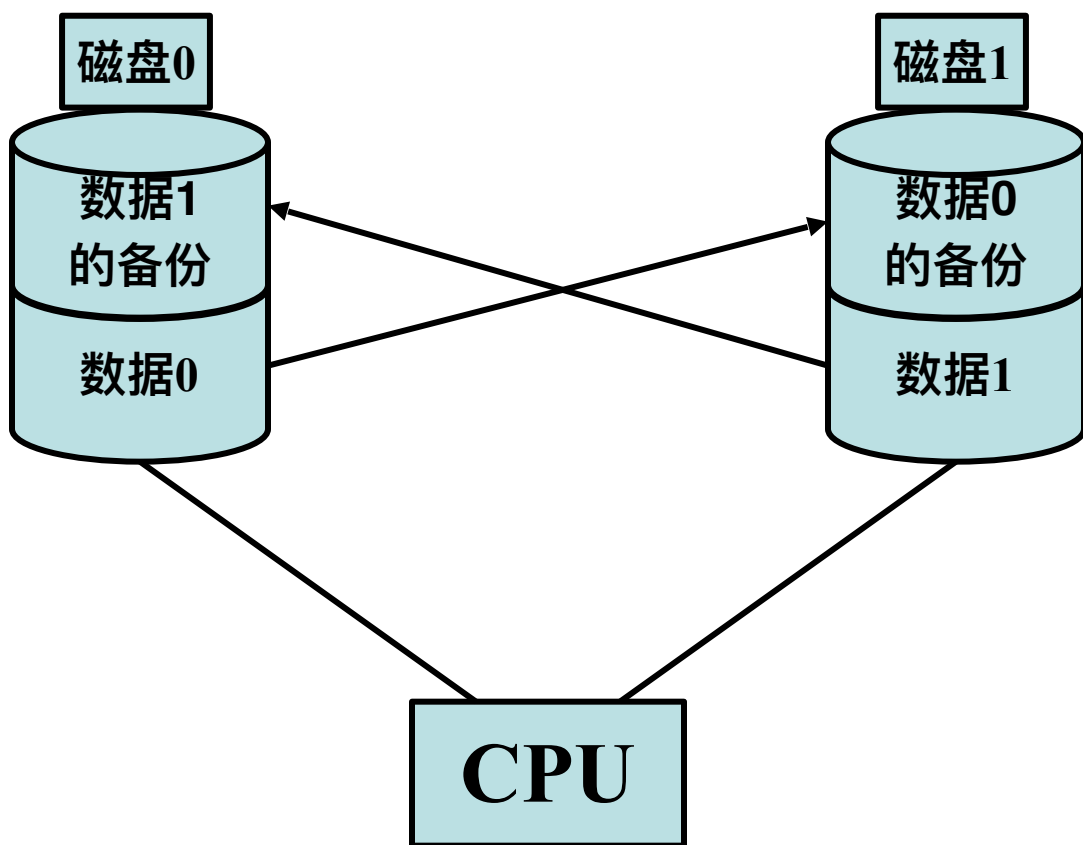
3.恢复

4.文件系统的一致性:文件写回磁盘块时,若在写回之前，系统崩溃，则文件系统出现不一致。需要设计一个实用程序，当系统再次启动时，运行该程序，检查磁盘块和目录系统。

5.一致性检查工作过程：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

- 安全性的两个重要方面：

1.数据丢失、灾难、硬件或软件故障、人的失误，可通过备份解决（存放在另一处）。

2.积极的或消极的入侵者：非技术人员的偶然窥视、入侵者的窥视、明确的偷窃企图和商业或军事间谍活动，设计安全时要考虑是那一类入侵者。

- 安全性的设计原则：

1.系统设计必须公开。

2.缺省属性应该不可访问。

3.检查当前权限。

4.给每个进程赋予一个最小的可能权限。

5.保护机制应简单一致，嵌入到系统底层。

6.采取的方案必须可接受。

## 文件的保护机制：

- 1.文件保护：用于提供安全性的特定的操作系统机制。（对拥有权限的用户，应该让其进行相应操作，否则，应禁止防止其他用户冒充对文件进行操作）。
- 2.用户验证：当用户登录时，检验其身份（用户是谁，用户拥有什么，用户知道什么）：口令、物理鉴定（磁卡，指纹，签名分析，手指长度分析等）。
- 3.存取控制：审查用户的权限和审查本次操作的合法性。
  - 1) 文件的二级存取控制：
    - 第一级：对访问者的识别，对用户分类：
      - a) 文件主（owner）。
      - b) 文件主的同组用户（group）。
      - c) 其它用户（other）。
    - 第二级：对操作权限的识别，对操作权限的分类：

- a) 读操作 (r) 。
- b) 写操作 (w) 。
- c) 执行操作 (x) 。
- d) 不能执行任何操作 (-) 。

## 2) 存取控制矩阵

文件

用户	A	B	C	.....
User1	rw	r	w	
User2	e	-	-	.....

# 访问列表和组

- 访问模式: read, write, execute
- 三类用户

a) owner access	7	⇒	RWX 1 1 1
b) group access	6	⇒	RWX 1 1 0
c) public access	1	⇒	RWX 0 0 1

- 管理员创建一个拥有唯一名字的组，组名为G，并将一些用户加入组内
- 对一个特定文件或者目录（假定名称为`game`），定义恰当的访问模式

**owner group public**  
| | |  
**chmod761 game**



# 小结

- 文件是存贮在某种介质上的（如磁盘、磁带等）并具有文件名的一组有序信息的集合
- 文件是逻辑信息组成的序列
  - 一个逻辑信息可能是字节、行、定长或者变长结构等
  - 操作系统可以支持多种文件类型，或者由应用程序来支持
- 基本文件操作: `read`, `write`...

# 小结

- 目录用于组织文件
  - 单层
  - 两层：允许多用户
  - 树结构等
- 目录记录了文件的信息
  - 文件名、磁盘上的位置、长度、类型、所有者、创建时间、上次使用时间、访问控制列表等
- 一个卷 (文件系统) 在使用前必须安装，取消安装后则不可访问

# 小结

- 文件共享依赖于操作系统提供的操作
- 访问控制、保护
  - 文件属性: 所有者(创建者), 组
  - 用户标识、组标识: uid, gid
  - 访问的类型: read, write, execute, append, delete, list...

# 文件系统实现

# 文件系统的类型

- FAT文件系统（MS-DOS文件系统、msdos）

它是MS-DOS操作系统使用的文件系统，它也能由Windows98/NT、linux、SCO UNIX等操作系统访问。文件地址以FAT表结构存放，文件目录32B，文件名为8个基本名加上一个“.”和3个字符扩展名。

- FAT32文件系统（vfat）

它是Windows98使用的扩展的DOS文件系统，它在MS-DOS文件系统基础上增加了对长文件名（最多到256B）支持。

- NTFS（NT文件系统）

它是Windows NT操作系统使用的文件系统，它具有很强的安全特性和文件系统恢复功能，可以处理巨大的存储媒体，支持多种文件系统。

- S51K/S52K（sysv）

它是AT&T UNIX S V 操作系统使用的1KB/2KB文件系统。

## 文件系统的类型

- ext2（二级扩展文件系统）

它是Linux操作系统使用的高性能磁盘文件系统，它是对Minux操作系统中使用的文件系统扩展（ext）的扩展。它支持256字符的文件名，最大可支持到4TB的文件系统大小。

- HPFS（高性能文件系统、hpfs）

它是OS/2操作系统使用的文件系统。

- CD-ROM文件系统(iso9660)

它是符合ISO9660标准的支持CD-ROM的文件系统，它有High sierra CD-ROM和Rock Ridge CD-ROM二种类型。

- UDF通用磁盘格式文件系统

UDF(Universal Disk Format)文件系统是依据光学储存技术协会（Optical Storage Technology Association, OSTA）的通用磁盘格式文件系统规格1.02版所制定的。它提供了对UDF格式媒体的只读访问（例如DVD光盘）。Windows98提供对UDF文件系统支持。

# 操作系统(Windows98/NT 、 Linux )

## 对多种文件系统的支持

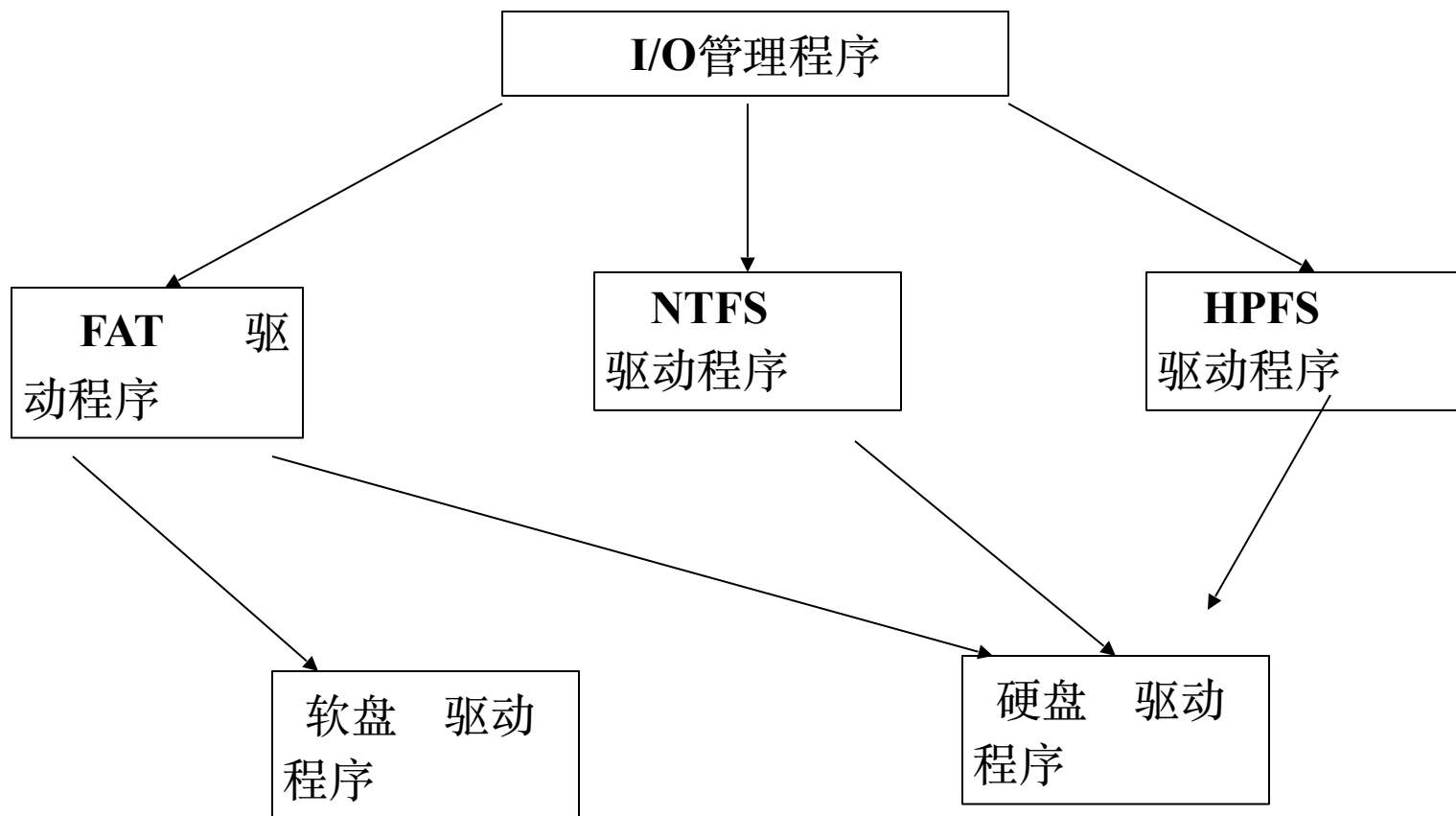
- WindowsNT多重文件系统

WindowsNT支持FAT文件系统、NTFS、HPFS、CD-ROM文件系统等多种文件系统。 Windows NT执行体内I/O系统分成I/O管理程序、文件系统驱动程序和盘驱动程序三层，不同的文件系统采用不同的文件系统驱动程序，系统用动态连接库对这些文件系统进行装入和卸出并适宜于将来的扩展， WindowsNT分层驱动程序如下图所示。

WindowsNT磁盘管理器窗口显示某台微机硬盘各分区安装的多种文件系统如下图所示，图中所示硬盘有二个分区，C盘是FAT文件系统，D盘是NTFS文件系统，E盘是CD-ROM文件系统。

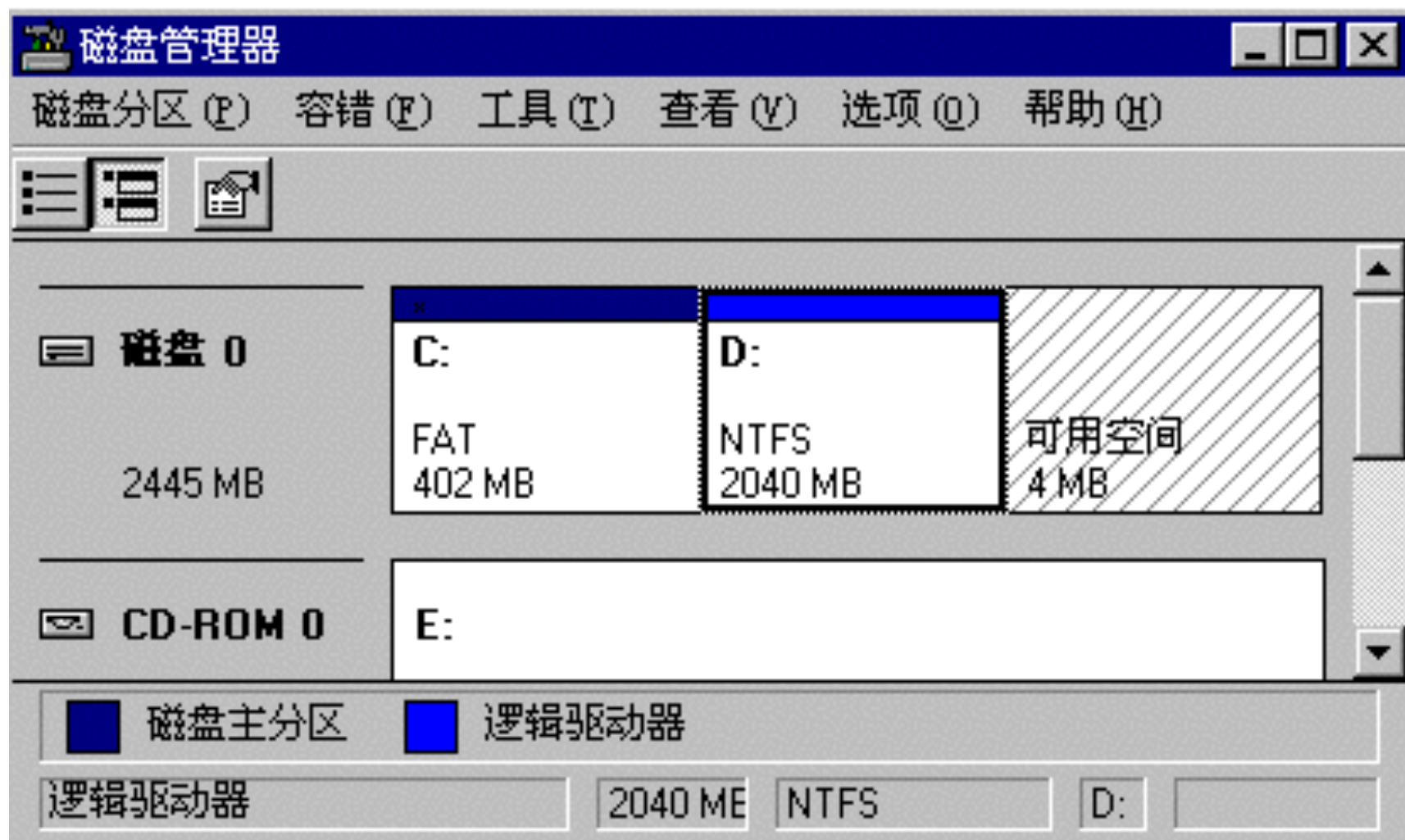
# WindowsNT多重文件系统-1

## WindowsNT分层驱动程序





## WindowsNT多重文件系统-2



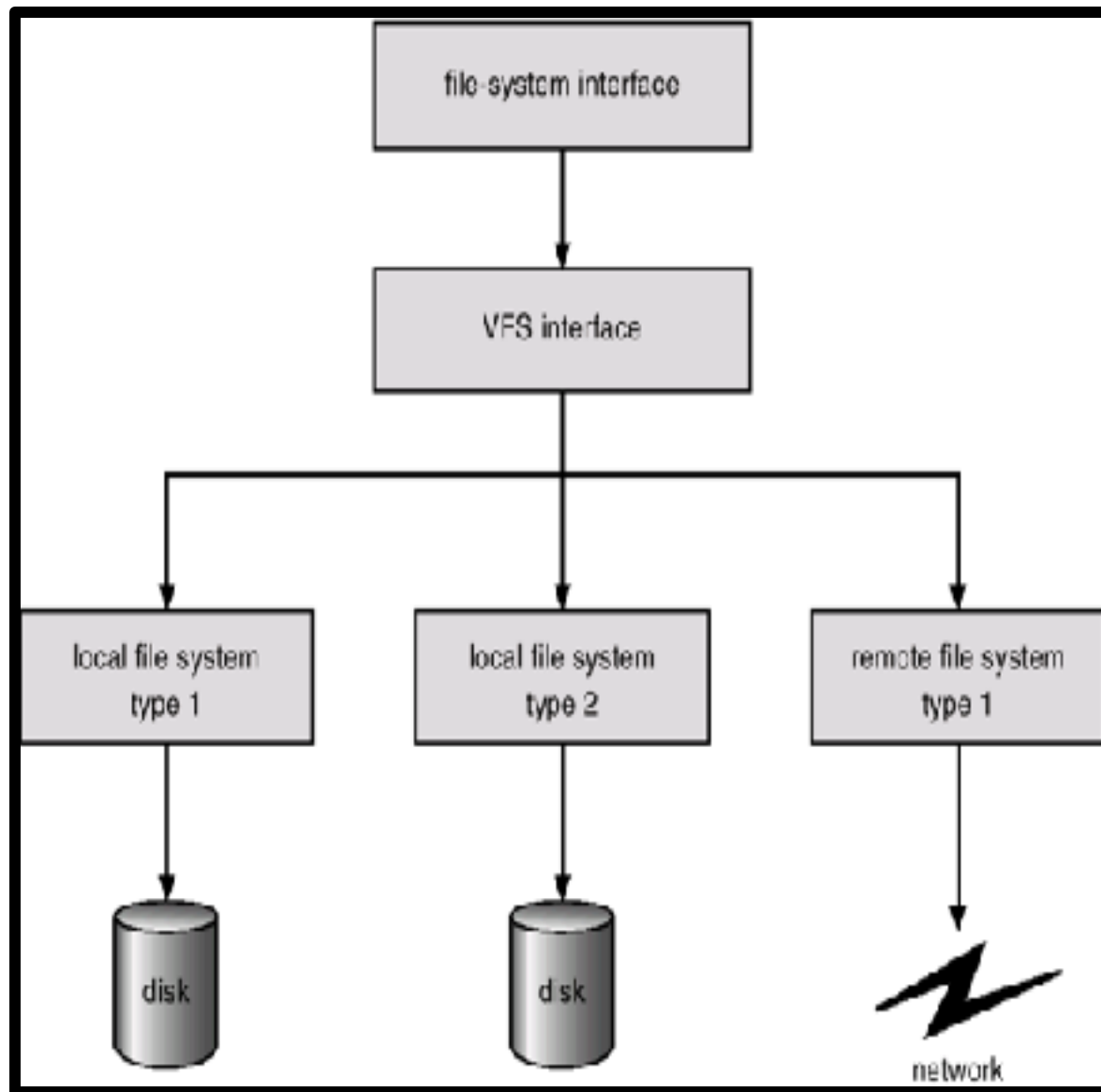
## Linux 虚拟文件系统VFS

- Linux采用虚拟文件系统VFS支持许多不同类型的文件系统，VFS是物理系统与服务之间的一个接口层，它屏蔽各类文件系统的差异，给用户和程序提供一个统一的接口。Linux支持ext、ext2、msdos、vfat、iso9660、hpfs等多种文件系统。使用命令mkfs创建各类文件系统。
- VFS是物理文件系统与服务之间的一个接口，它对Linux的每一个文件系统的所有细节进行抽象，使得不同的文件系统在Linux内核以及系统中运行的其它进程看来都是相同的，严格说来VFS并不上一种实际的文件系统，它只存在于内存中，不存在于任何外存空间，VFS在系统启动时建立，在系统关闭时消亡。
- VSF使Linux同时安装支持不同类型的文件系统成为可能。

# 虚拟文件系统

- 虚拟文件系统 (**VFS**)提供一个面向对象的文件系统实现方法，以允许不同文件系统类型可以通过同样结构来实现
- **VFS**允许不同类型的文件系统使用相同的系统调用接口
  - **API**作为**VFS**接口要好于规定的文件系统类型

# VFS示意图



第一层文件系统接口，包括**open**，**read**，**write**和**close**调用及文件描述符

第二层**VFS**层，目的：

1 将文件系统通用操作和具体实现分开

2 **VFS**是基于称为**vnode**的文件表示结构，该结构包括一个数值指定者以表示位于整个网络范围内的唯一文件。因此，**VFS**区分本地文件和远程文件

# 目录实现

- 对目录查询的技术有两种：线性检索法和Hash法
  - 线性列表—包括存储文件名、指向数据块的指针
    - 采用线性搜索来查找特定条目，容易编程
    - 耗CPU的执行时间
    - 许多操作系统采用软件缓存来存储最近访问过的目录信息，缓存命中避免不断地从磁盘读取信息
  - 哈希表—有着哈希数据结构的线性表
    - 减少目录的搜索时间
    - 冲突：两个名字映射到同样的位置
    - 固定大小和哈希函数对大小的依赖性

# 文件存储空间管理

- 确定了块的大小，还要对它们进行管理，即要记住哪些已经分配，哪些仍然空闲。常采用的磁盘存储空间管理方案有位示图、空闲块表以及空闲块链。
- 位示图：**为所要管理的磁盘设置一张位示图。位示图的大小由磁盘的总块数决定。位示图中的每个二进制位与一个磁盘块对应，该位状态为“1”，表示所对应的块已经被占用；状态为“0”，表示所对应的块仍然是空闲，可以参加分配。

	0 位	1 位	2 位	3 位	...	30 位	31 位
第 0 字	0/1	0/1	0/1	0/1	...	0/1	0/1
第 1 字	0/1	0/1	0/1	0/1	...	0/1	0/1
...					...		
...	0/1	0/1	0/1	0/1	...	0/1	0/1
...	0/1	0/1	0/1	0/1	...	0/1	0/1
第 99 字	0/1	0/1	0/1	0/1	...	0/1	0/1

# 文件存储空间管理

- **空闲区表：**用空闲区表来管理文件存储空间，做法是系统设置一张表格，表中的每一个表目记录磁盘空间中的一个连续空闲盘区的信息。

序号	起始空闲块号	连续空闲块个数	状态
1	2	5	有效
2	18	4	有效
3	59	15	有效
4	80	6	空白
...	...	...	

(a)

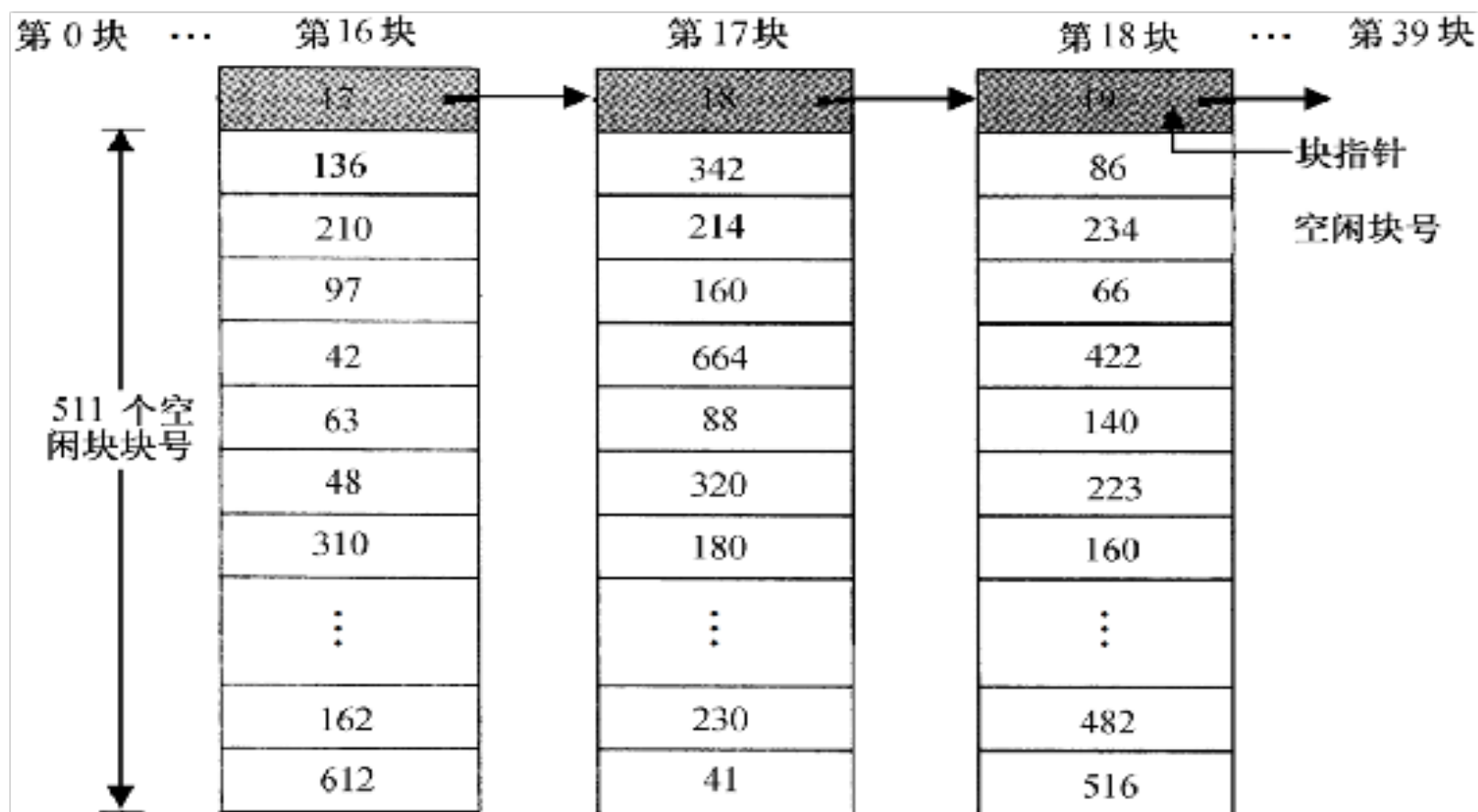
序号	起始空闲块号	连续空闲块个数	状态
1	2	5	有效
2	18	4	有效
3	65	9	有效
4	80	6	空白
...	...	...	

(b)

# 文件存储空间管理

- **空闲块链**：所谓空闲块链即在磁盘的每一个空闲块中设置一个指针，指向另一个磁盘空闲块，从而所有的空闲块形成一个链表，这就是磁盘的“空闲块链”。有时也把磁盘空闲块链称为“空白串联文件”。
- **成组链接**：一种改进的方法，在这种方法中，系统根据磁盘块数，开辟若干块来专门登记系统当前拥有的空闲块的块号，UNIX操作系统就是采用的这种方法。
- **内存中所需的表目**：
  - (1) **系统打开文件表**：放在内存，用于保存已打开文件的FCB。此外，还有文件号，共享计数，修改标志等。
  - (2) **用户打开文件表**：每个进程一个，文件描述符，打开方式，读写指针，系统打开文件表入口，在进程的PCB中，记录了用户打开文件表的位置。





用户打开文件表与系统打开文件表之间的关系：用户打开文件表指向系统打开文件表。如果多个进程共享同一个文件，则多个用户打开文件表目对应系统打开文件表的同一入口。

PCB主部	文件号	共享计数	修改标志
.....	.....	.....	.....

系统打开文件表

文件描述符	打开方式	读写指针	系统打开文件表入口
.....	.....	.....	.....

用户打开文件表

其它	系统打开文件表入口
.....	.....
.....	.....
.....	.....

用户打开文件表(P1)

其它	系统打开文件表入口
.....	.....
.....	.....
.....	.....

用户打开文件表(P2)

系统打开文件表

共享计数	其它
.....	.....
2	.....
.....	.....



# 文件系统效率和性能

- 效率取决于
  - 磁盘分配和目录管理算法
  - 保留在文件目录条目中的数据类型
- 性能
  - 磁盘缓存 – 一块独立内存，装载常用的块
  - 马上释放（free-behind）和 预先读取（read-ahead） – 优化顺序访问的技术
  - 留出一块内存作为虚拟磁盘（或 *RAM* 磁盘）
    - 只能用于临时存储
    - 通常用于个人计算机

# 小结

- 分层文件系统
  - 低层: 处理存储设备的物理属性
  - 高层: 处理符号文件名和文件逻辑属性
  - 中间层: 将逻辑文件概念映射到物理设备属性
- 文件系统有不同的结构和算法
  - **VFS** 层允许上层统一的处理每个文件系统类型
- 存储空间管理
- 性能，可靠性
  - 缓存
  - 日志

# 基于日志结构的文件系统

- 日志结构文件系统记录每一次更新为文件系统的一个 transaction
- 所有的 transaction 都被记录在日志中，一旦写入，transaction 即被认为是 committed，然而文件系统可能仍然没有更新。
- 当 committed transaction 完成，该 transaction 从日志中删除。
- 如果文件系统崩溃，在日志所有保存事务仍必须执行（恢复）