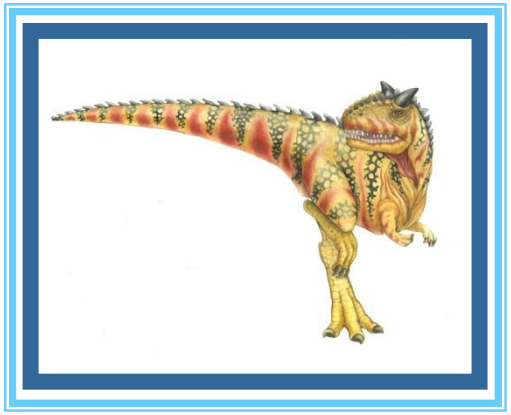
第6章 文件系统



主要内容

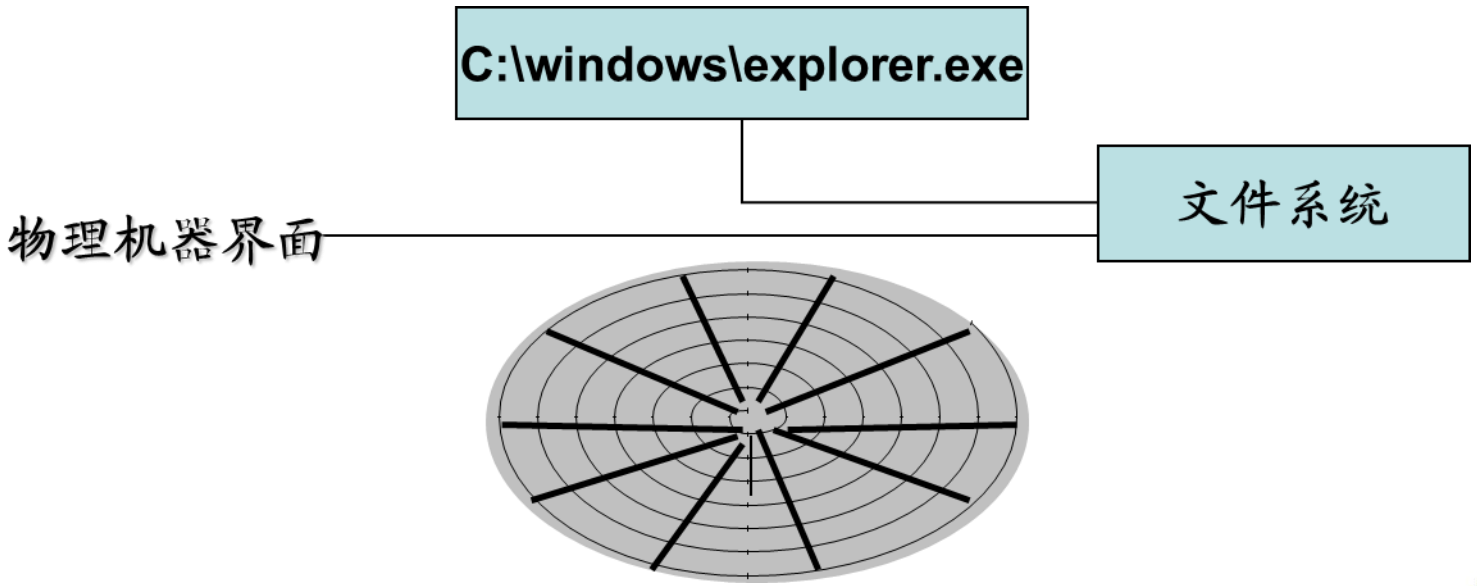
- 文件系统结构
  - 文件和文件系统
  - 文件的逻辑结构
- 目录管理
- 文件系统实现
- 外存分配方法
- 文件存储空间的管理

文件系统结构

- 文件系统的两个设计问题
  - 如何定义文件系统对用户的接口：
    - 文件及其属性
    - 文件操作
    - 目录结构
  - 创建数据结构和算法来将逻辑文件系统映射到物理外存设备上

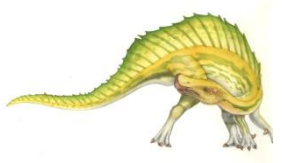
6.1 文件和文件系统

- 什么是文件
  - 是数据的一种组织形式
- 什么是文件系统
  - 文件系统是操作系统提供的对于磁盘的抽象。是介于用户与磁盘之间的界面。



6.1.1 文件类型

- 按文件的性质和用途分类
  - 系统文件
    - 操作系统及其他系统程序构成系统文件范畴。
  - 用户文件
    - 指用户在软件开发过程中产生的各种文件，如源程序、目标程序代码和计算结果等。
  - 库文件
    - 常用的标准子程序、实用子程序等组成库文件。



6.1.1 文件类型

- 按文件中的数据形式分类
  - 源文件
    - 通常由ASCII码构成
  - 目标文件
    - 二进制文件
  - 可执行文件

6.1.1 文件类型

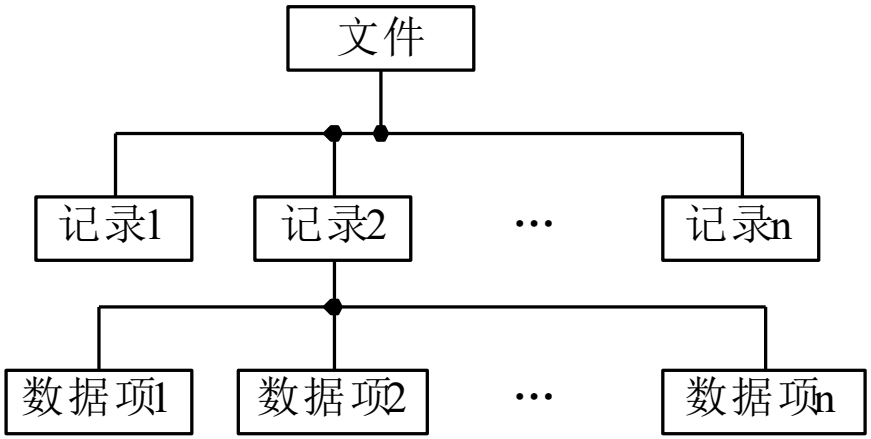
- 按文件的保护性质分类
  - 只读文件
  - 读写文件
  - 可执行文件

6.1.1 文件类型

- 按组织形式和处理方式分类
  - 普通文件
    - 存储在磁盘上的一般文件。
  - 目录文件
    - 操作系统把文件的目录项聚集在一起，形成一个文件加以管理，称为“目录文件”。
  - 特殊文件
    - 为了统一管理和方便使用设备，操作系统常以文件的观点来看待设备，也称为“设备文件”。

6.1.1 文件类型

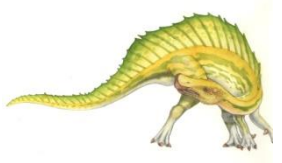
- 按文件的逻辑结构分类
  - 有结构文件
    - 由记录构成的文件，故又称为记录式文件。如：数据结构文件、数据库文件。



  - 无结构文件
    - 是指由字符流构成的文件，故又称为流式文件。如：源程序、目标代码等

6.1.1 文件类型

- 按文件的物理结构分类
  - 顺序文件
    - 是指把逻辑文件中的记录顺序的存储到连续的物理盘块中。
  - 链接文件
    - 是指文件中的各个记录可以存放在不相邻接的各个物理盘块中
    - 通过物理盘块的链接指针，将它们链接成一个链表。
  - 索引文件
    - 是指文件中的各个记录可以存储在不相邻接的各个物理盘块中
    - 为每个文件建立一张索引表，来实现记录和物理块之间的映射



文件属性

- 文件属性包括：
  - 文件名
  - 标识符：系统内部文件的唯一标识
  - 类型
  - 物理位置
  - 大小
  - 用户标识
  - 权限：读、写、执行等访问控制信息
  - 时间、日期：文件创建、上次修改和上次访问都可能有该信息。用于保护、安全和使用跟踪

文件操作

- 创建文件--create()
- 打开/关闭文件--open()/close()
- 写文件--write()
- 读文件--read()
- 在文件内重定位--lseek()
- 截短文件--truncate()

文件结构

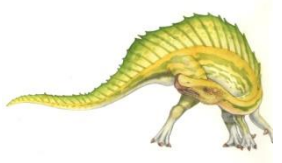
- 文件的逻辑结构
  - 从用户观点出发，所观察到的文件的组织形式，是用户可以直接处理的数据及其结构
- 文件的物理结构
  - 是指文件在外存上的存储组织形式，又称为文件的存储结构。

6.2 文件逻辑结构

- 分类
  - 无结构文件
    - 是指由字符流构成的文件，故又称为流式文件。如：源程序、目标代码等
  - 有结构文件
    - 由记录构成的文件，故又称为记录式文件。如：数据结构文件、数据库文件。

6.2.1 文件逻辑结构的类型

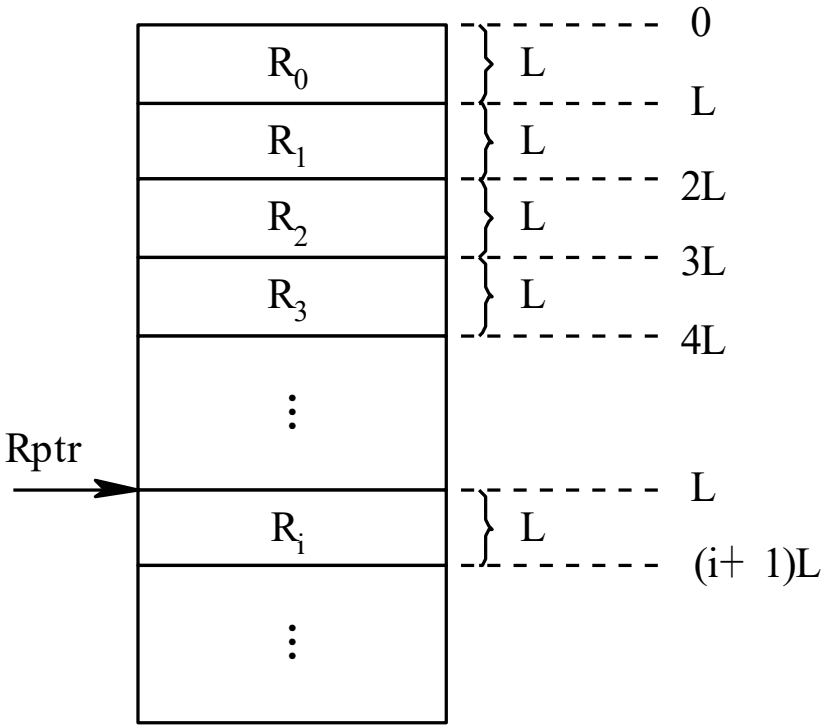
- 记录式文件的的几种形式
  1. 顺序文件
  2. 索引文件
  3. 索引顺序文件
- 目标
  - 提高检索速度,便于修改



1、顺序文件

- 定长记录顺序文件
  - 易于定位，可直接访问。
  - 第i个记录的首地址：

    Ai＝i*L

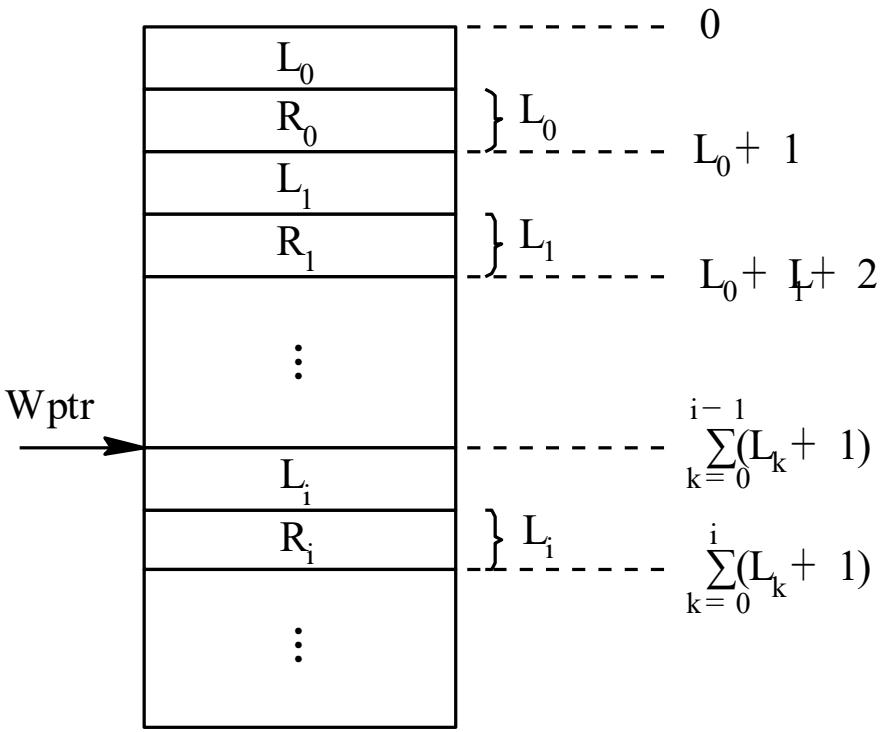


(*a*) 定长记录文件

1、顺序文件

- 变长记录顺序文件
  - 不易定位，只能顺序访问
  - 第i个记录的首地址：

$$Ai = \sum_{i=1}^{i} Li + i$$



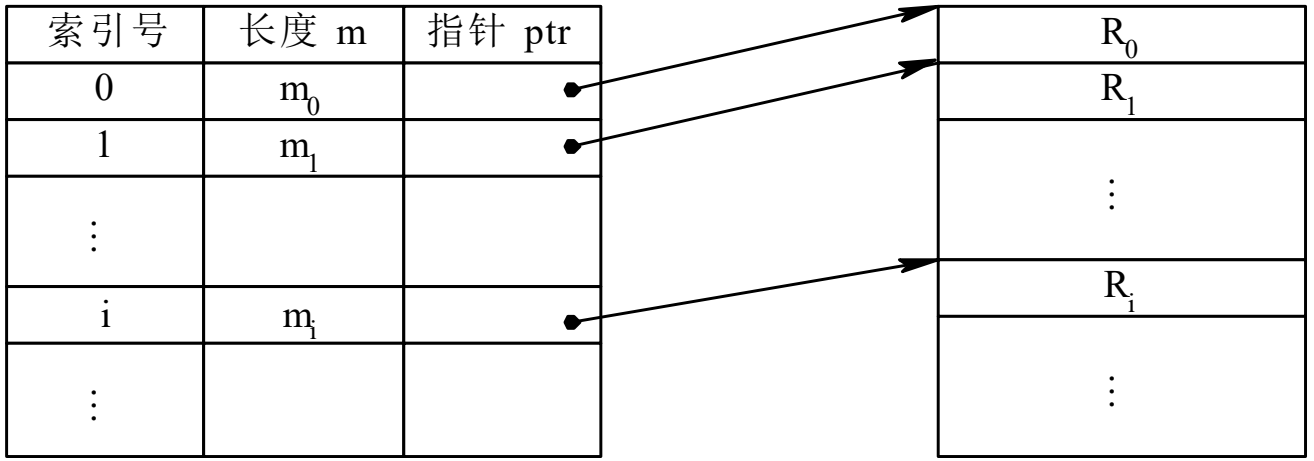
(*b*) 变长记录文件

1、顺序文件

- 优点
  - 批量存取的效率高
- 缺点
  - 查找效率低
  - 增加、删除一个记录难

2、索引文件

- 基本思想
  - 当记录为可变长度时，通常为记录建立一张索引表；索引表本身是一个定长记录文件。
  - 索引项按照记录中的某个关键字排序。

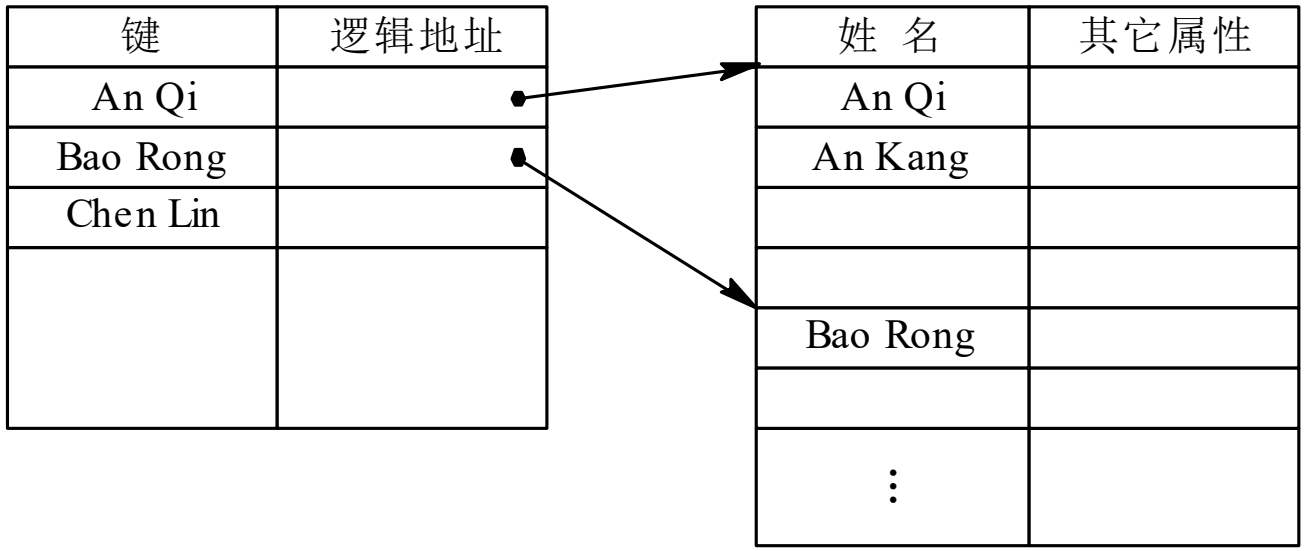


2、索引文件

- 特点
  - 可以实现直接存取。
  - 增加了存储开销—索引文件。
- 缺点
  - 对于大文件，索引本身可能太大
  - 解决方法：建立二级索引

3、索引顺序文件

- 基本思想
  - 将顺序文件中的所有记录分为若干组；为每组记录设置一个索引表项

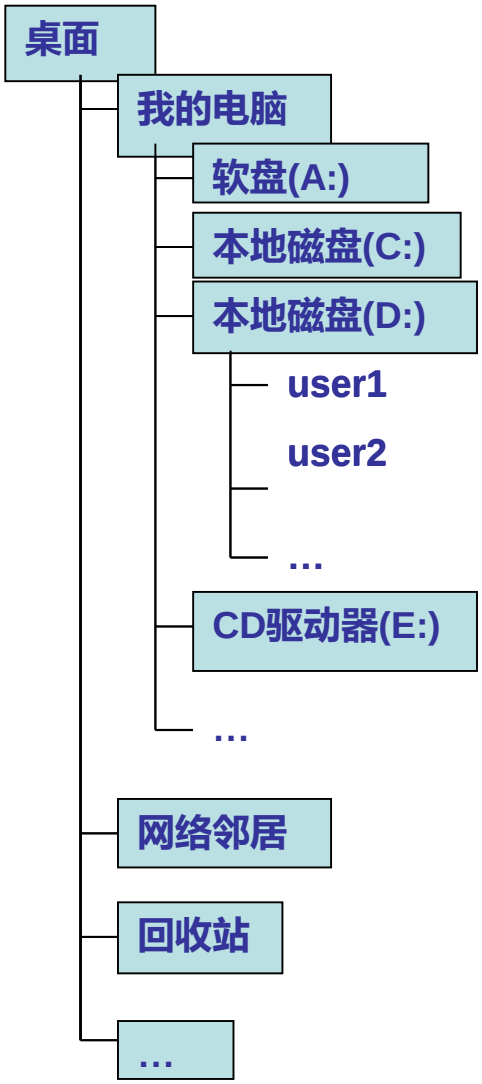


逻辑文件

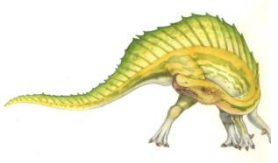
3、索引顺序文件

- 优点
  - 通过划分层次，在记录数量较大时，比顺序文件缩短检索时间。
  - 假设：有10000个记录
    - 顺序文件：平均检索5000次
    - 索引顺序文件（100个记录一组）：平均检索50+50=100次

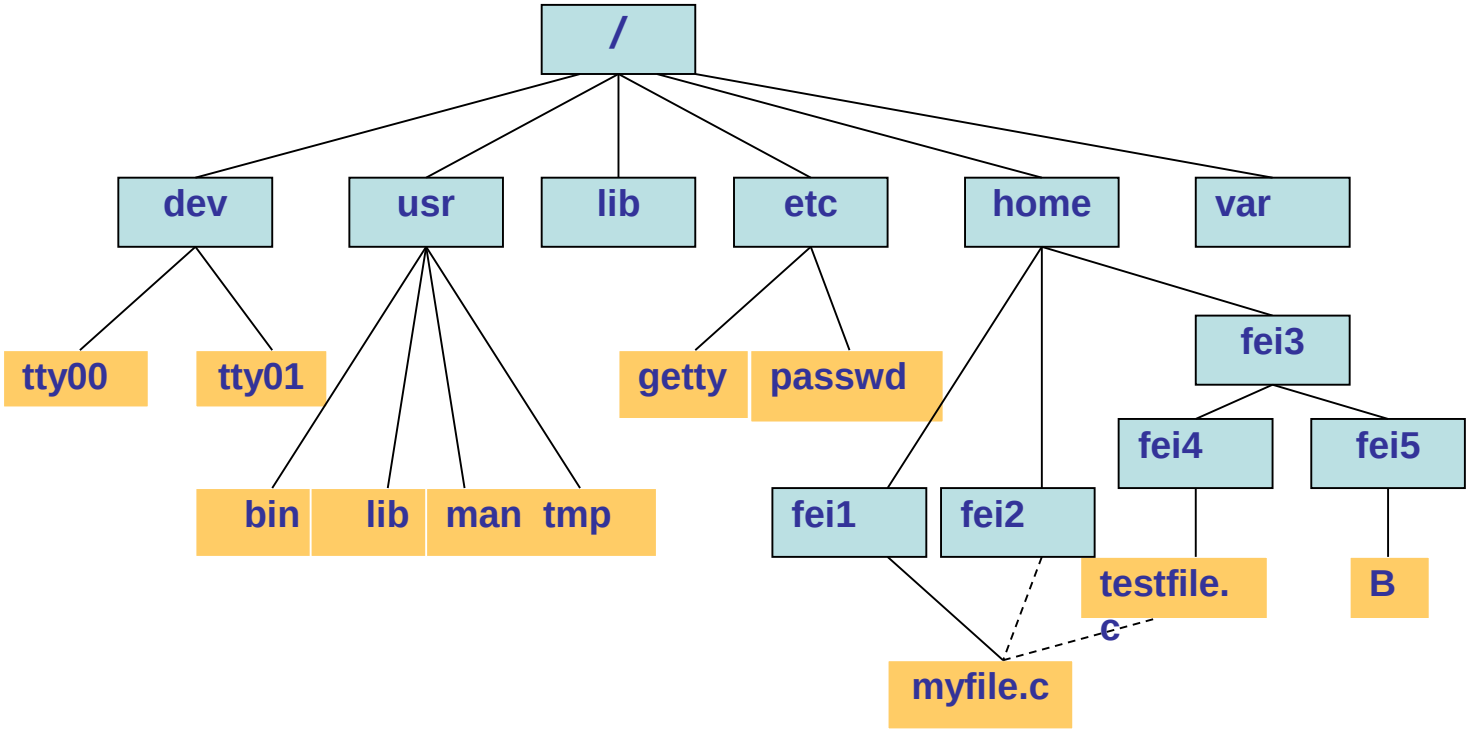
6.3 目录结构



Windows的层次目录结构



6.3 目录结构



Linux的层次目录结构

6.3 目录结构

- 功能
  - 实现按名存取--文件名到磁盘的映射
  - 提高对文件的检索速度
  - 文件共享
  - 允许文件重名

文件控制块和目录文件

- 文件控制块（FCB）
  - 记录该文件的有关信息。找到文件的FCB，就得到该文件的有关信息，就能对它进行所需的访问。
- 目录文件
  - 把系统中各个文件的文件控制块汇集在一起，就形成了系统的文件目录，每个文件控制块就是一个目录项

文件控制块和目录

- 文件控制块的内容

文件名	
文件在磁盘中的起始地址	
记录长度	记录个数
文件主及存取权限	
其他用户的存取权限	
...	
文件建立的日期和时间	
上次访问的日期和时间	

- 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

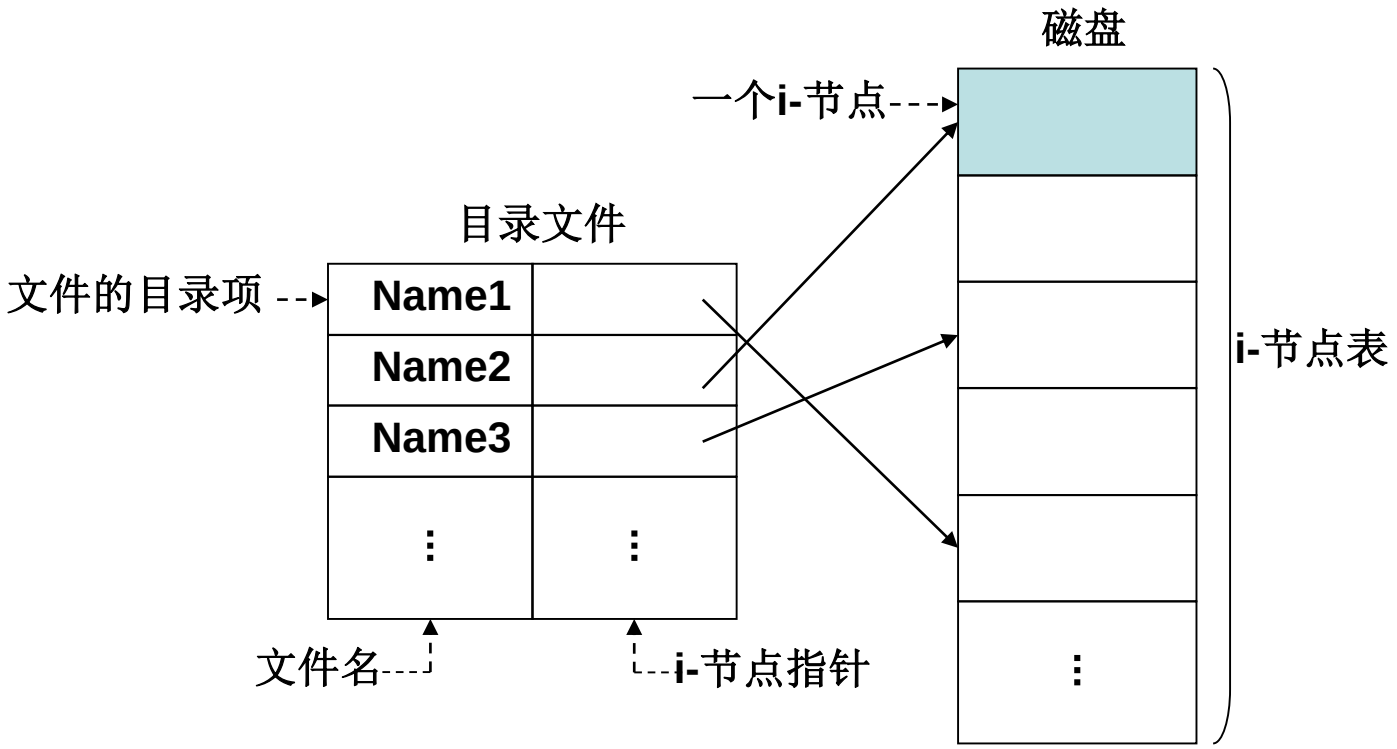
- 缺点
  - 大量的目录项进入内存，占用了内存空间
  - 增加了读盘的次数。

索引结点

- 基本思想
  - 为了加快对文件目录的搜索过程，为了提高对文件的访问速度，现代操作系统就常采用把FCB中的文件名与其他有关信息分离的办法。
    - 目录项：由文件名和相应的“i节点”指针组成
    - 索引节点(index node)： 简称“i节点”

索引结点

- 目录项、i节点指针、i节点表之间的关系



索引结点

- 优点
  - 减少了检索目录带来的内存空间的浪费。
  - 减少了读盘的次数（查询时只调入文件名部分）

练习

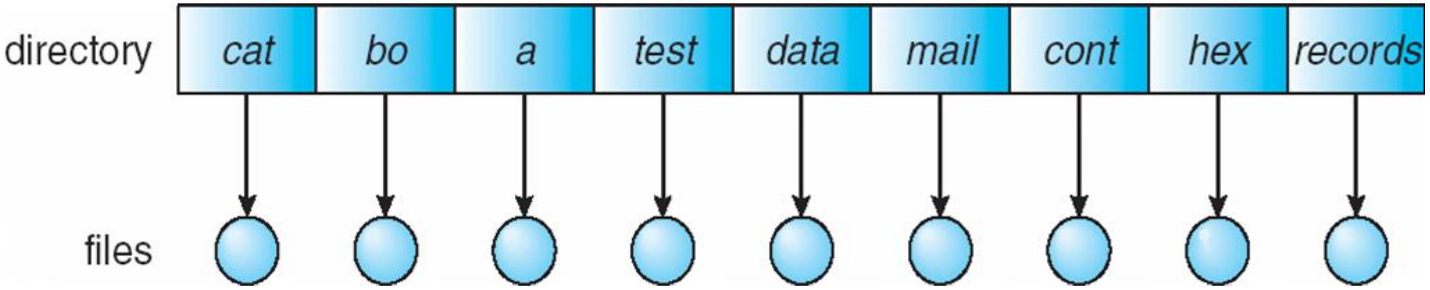
- 在某个文件系统中，每个盘块为512B，文件控制块占64B，其中，文件名占8B。如果索引节点号占2B，对于一个目录文件包含256个目录项。试比较引入索引节点前后，为找到该目录中某个文件的FCB的访盘次数。

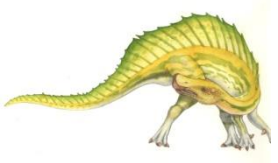
目录结构

- 常用的目录结构
  - 单级目录
  - 两级目录
  - 多级目录/树形目录

1、单级目录

- 所有文件都包含在同一目录中
  - 使用一个目录来包含系统中的所有文件。
  - 整个文件系统只建立一个目录文件
  - 每个文件占一个目录项



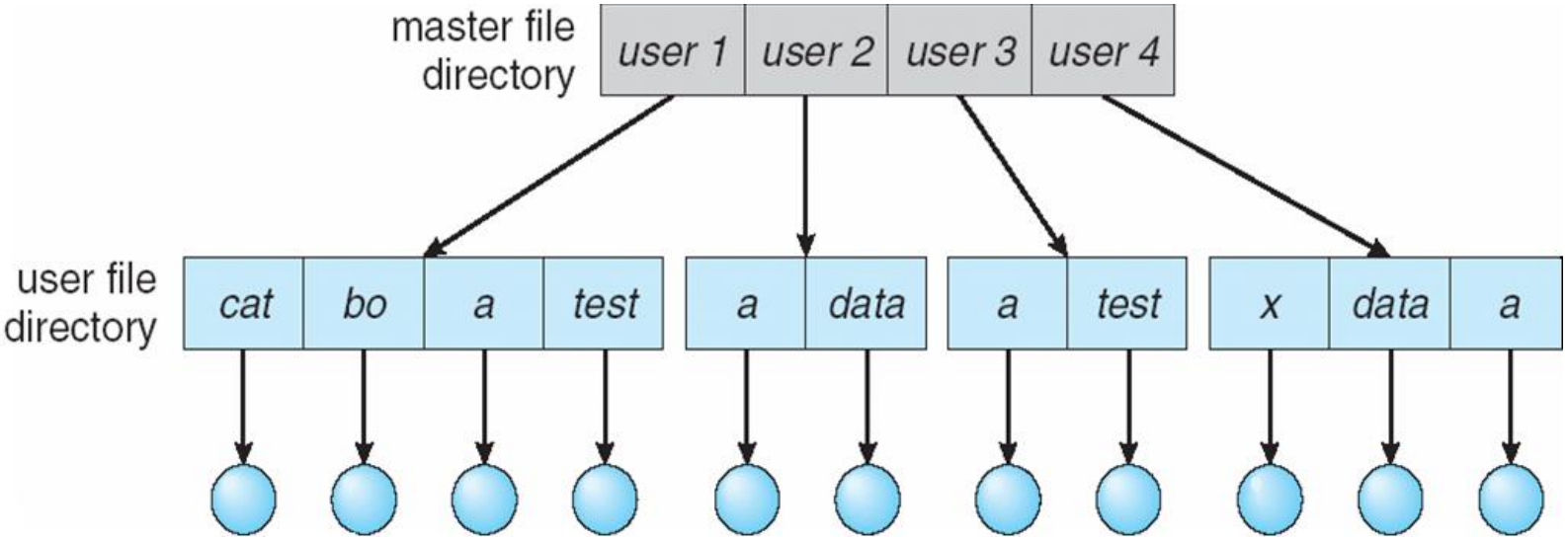


单级目录

- 特点
  - 查找速度慢
  - 不允许重名
  - 不便于共享(不能用不同名字访问同一文件)

两级目录

- 为每个用户建立独立的目录
  - 主目录（MFD）
  - 用户目录（UFD）

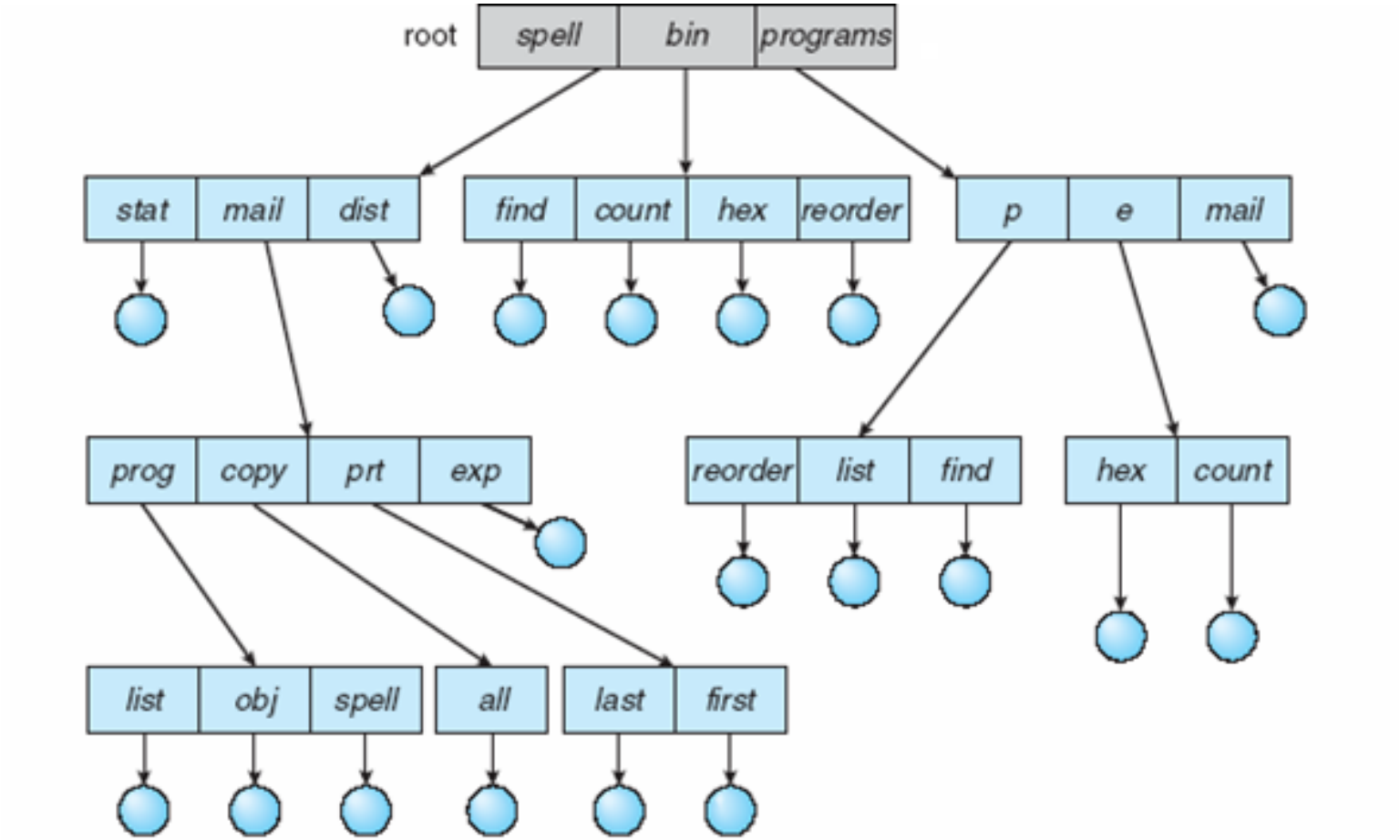


两级目录

- 特点
  - 提高了检索的速度
    - 如：主目录：n个目录项，用户目录：m个目录项
      - 采用单级目录结构：最多需检索内n*m次
      - 采用两级目录结构：最多只需检索n+m次
  - 在不同的用户目录中， 可以使用相同的文件名。
  - 不同用户可使用不同的文件名来共享文件

树形目录

- 目录采用层次结构，允许用户在自己的目录下，创建多层子目录。

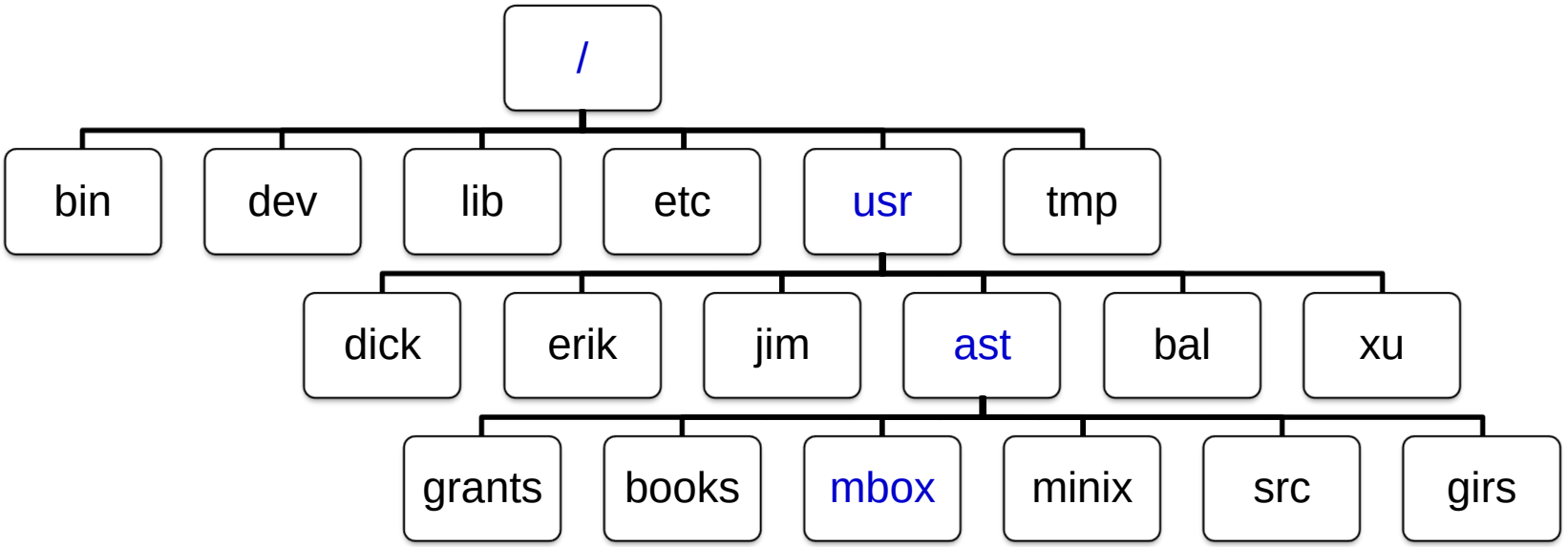


树形目录

- 特点
  - 层次清楚
  - 有利于文件的保护
  - 解决文件的重命名问题
  - 提高查找速度

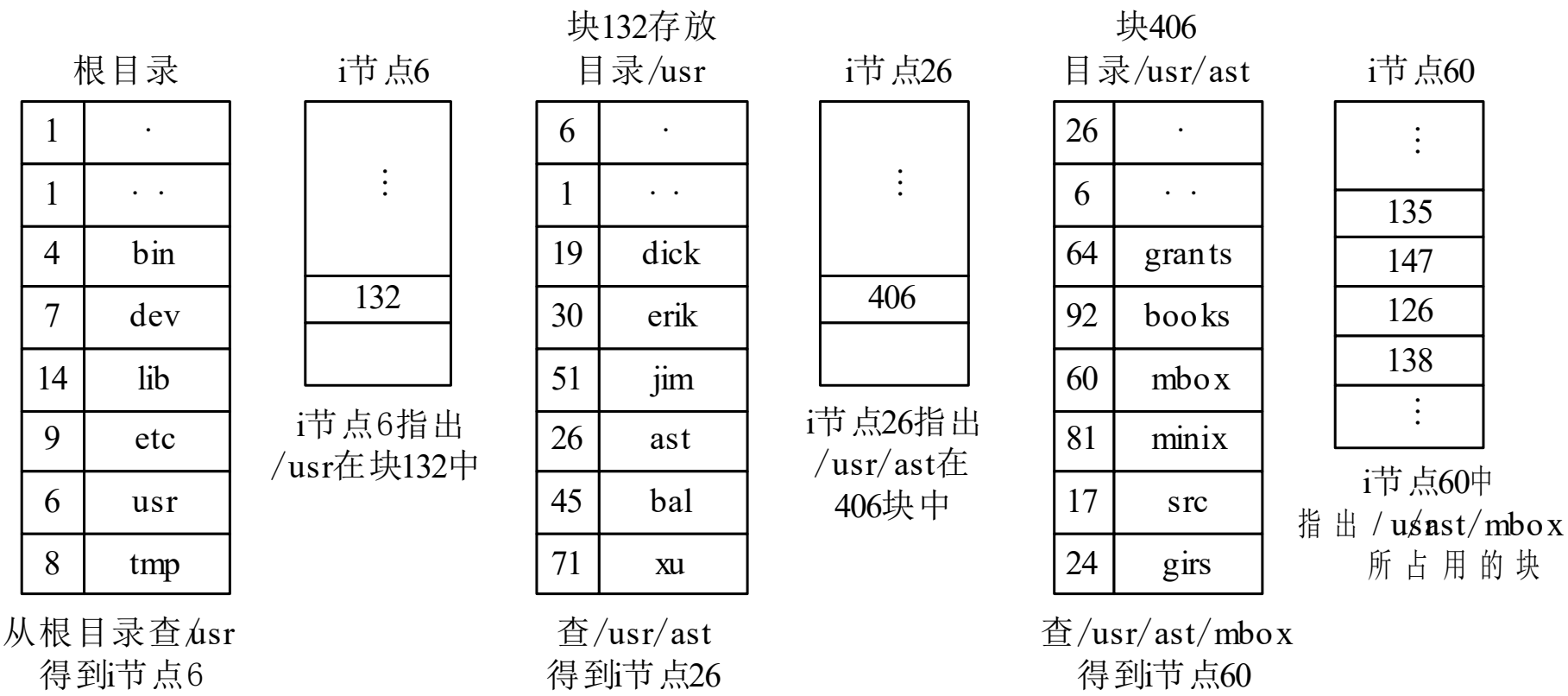
目录查询

- 例：查找 /usr /ast /mbox



目录查询

- 例：查找 /usr /ast /mbox

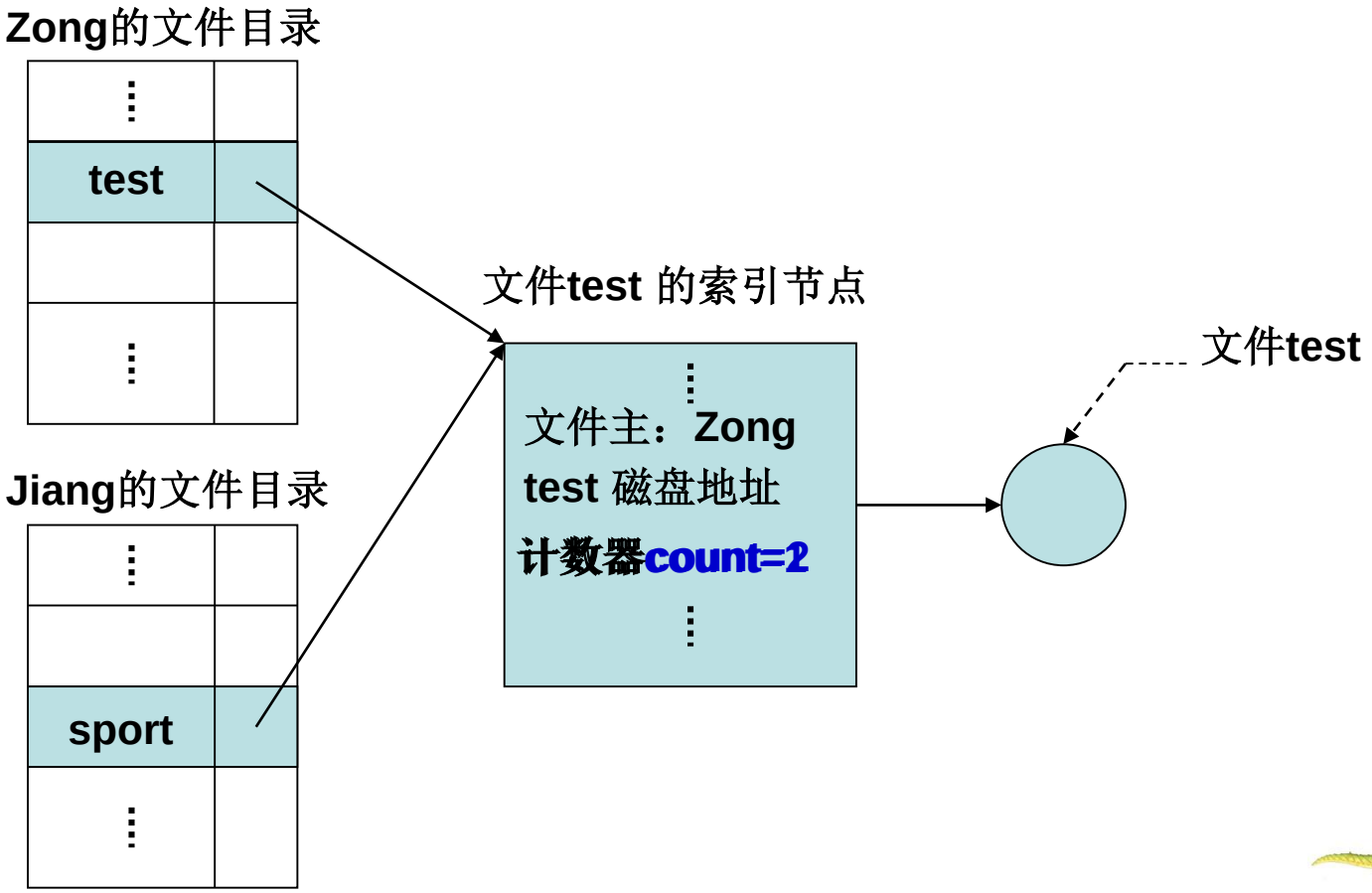


文件共享

- 引入
  - 共享文件使得在文件存储设备上只需存储一个文件副本，节省了存储空间。
- 文件共享的方法
  1. 基于索引节点的共享方式
  2. 利用符号链实现文件共享

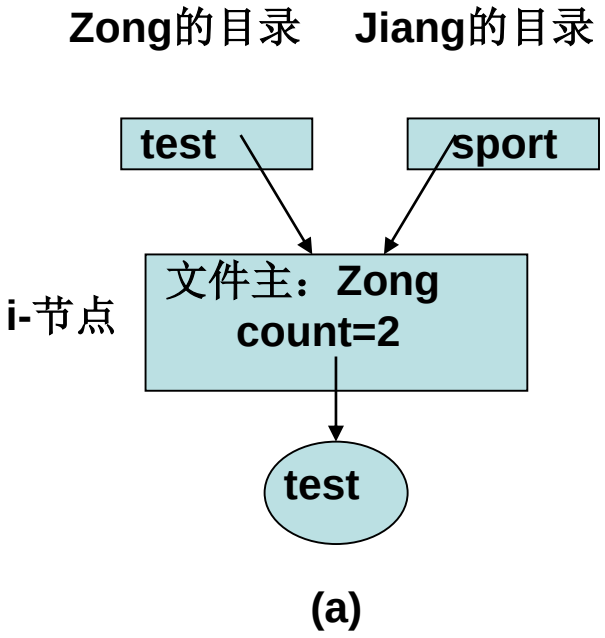
1、基于索引结点的共享方式

- 基本思想
  - 通过i节点把用户链接到共享文件。



1、基于索引结点的共享方式

- 特点
  - 当任何用户修改文件的内容时，其他用户可见。
  - 计数器count用来记录当前有多少个用户在共享使用该文件，count为0时，才能删除文件。



2、利用“符号链”实现文件共享

- 基本思想
  - 通过“符号链接”文件把用户链接到共享文件。
  - “符号链接”文件：系统定义一种新的文件类型，这种类型的文件里只包含所要共享的文件的路径名。
- 特点
  - 文件主将文件删除后，链接用户将无法再访问该文件。
  - 增加读盘的频率
  - “符号链接”文件占用磁盘空间

文件系统

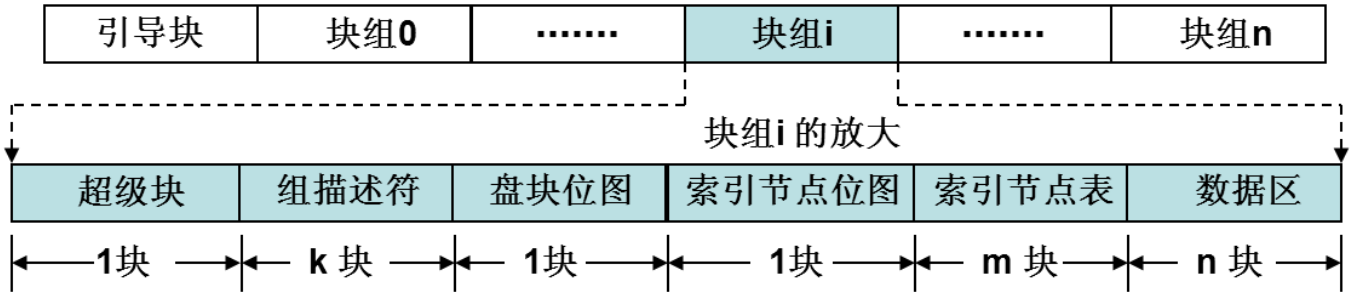
- 绝大多数操作系统都支持多个文件系统
  - CD-ROM： ISO 9660;
  - Unix：UFS, FFS;
  - Windows ： FAT, FAT32, NTFS
  - Linux ： 支持 40 多种文件系统, 标准的文件系统： extended file system (ext2 ext3 ex4)
  - FFS： distributed file systems

文件系统的实现

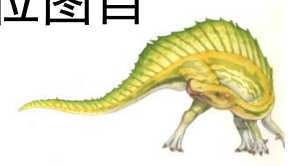
- 磁盘结构包括
  - 引导控制块（boot control block）：
    - 通常为分区的第一块。引导块（Linux）、分区引导扇区（WindowsNT）
  - 分区控制块（partition control block）：
    - 包括分区详细信息，如分区的块数、块的大小、空闲块的数量和指针、空闲FCB的数量和指针等，亦称为超级块（Linux）、主控文件表（WindowsNT）
  - 目录结构：
    - 用来组织文件
  - 文件控制块（FCB）：
    - 包括很多文件信息，如文件许可、拥有者、大小和数据块的位置等

文件系统布局

- Ext2对磁盘的组织

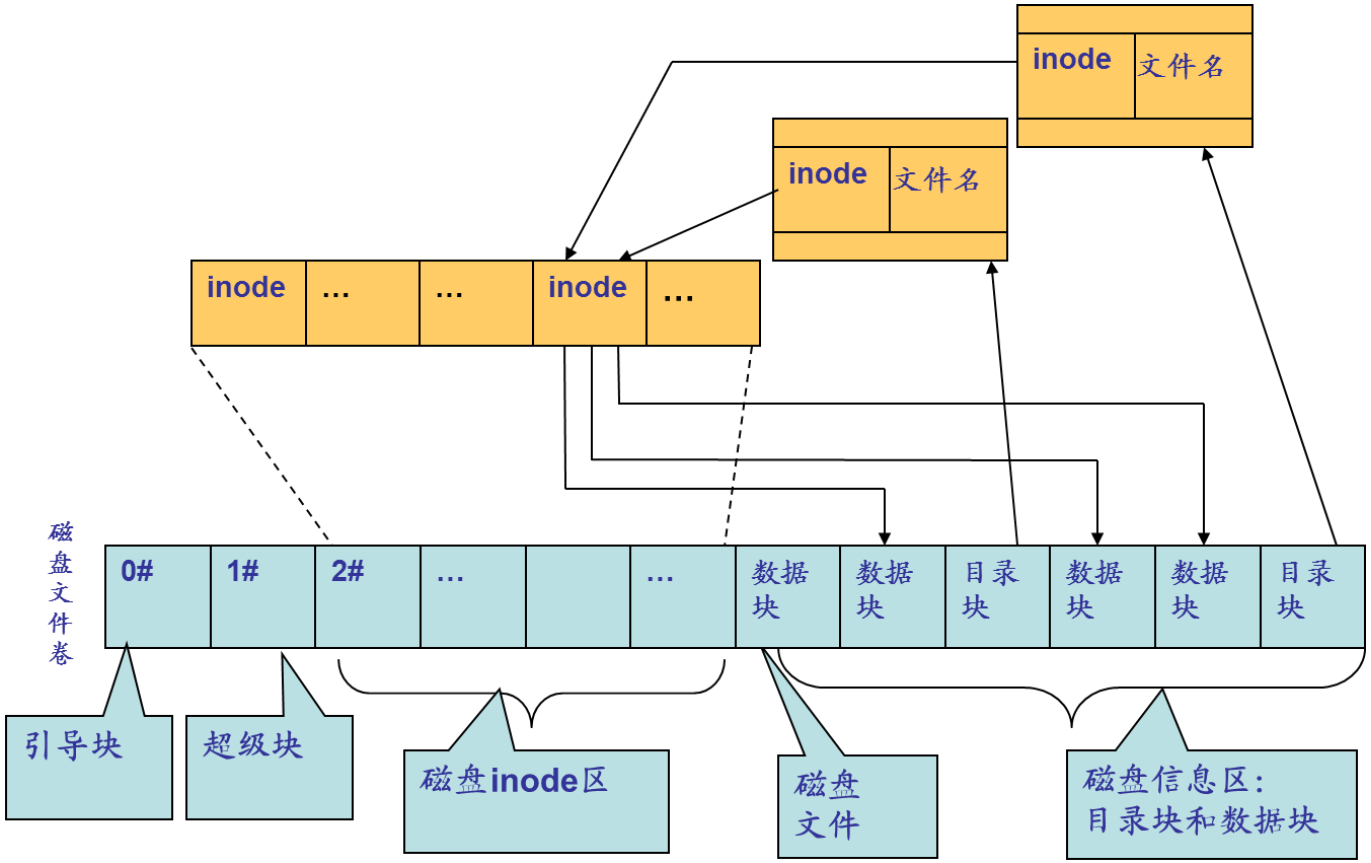


- 数据区：存放普通文件、目录文件等具体信息的地方，它占用了块组最多的盘块。
- 索引节点表：每个文件和目录文件的索引节点inode，这些索引节点的集合，构成了所谓的“索引节点表”。
- 索引节点位图：用来管理块组中的索引节点，位图中的每位，对应索引节点表里的一个表项。
- 盘块位图：管理块组中数据区里的盘块。在块组中，盘块位图自己占据一个盘块。



文件系统布局

- 目录项、inode和数据块的关系



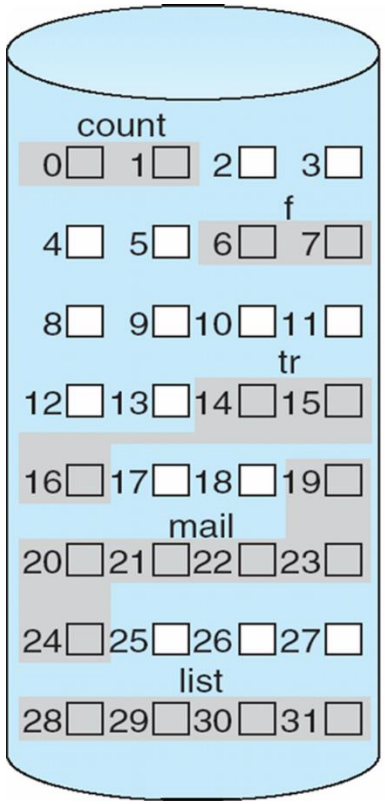
外存分配算法

- 文件在磁盘上是如何存放的（文件的物理结构）
  - 连续分配(contiguous allocation)
  - 链接分配(linked allocation)
  - 索引分配(indexed allocation)

- 目标
  - 在分配外存空间时，如何有效地利用空间
  - 提高文件的访问速度

6.4.1 连续分配

- 基本思想
  - 文件存储到一连串连续的存储块中，即顺序分配磁盘上的存储块。这样的文件为顺序文件或连续文件。

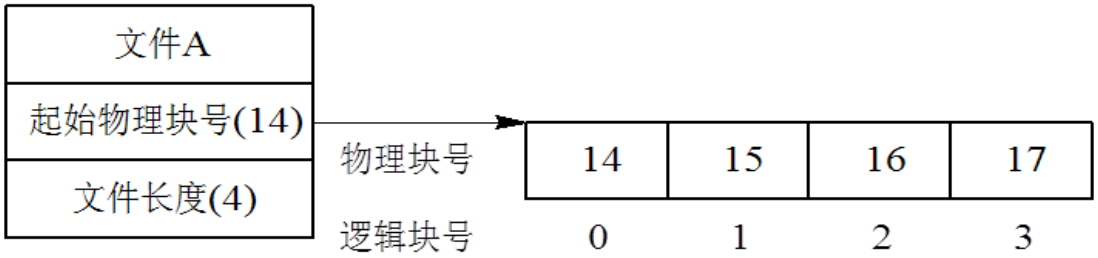


directory

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

连续分配

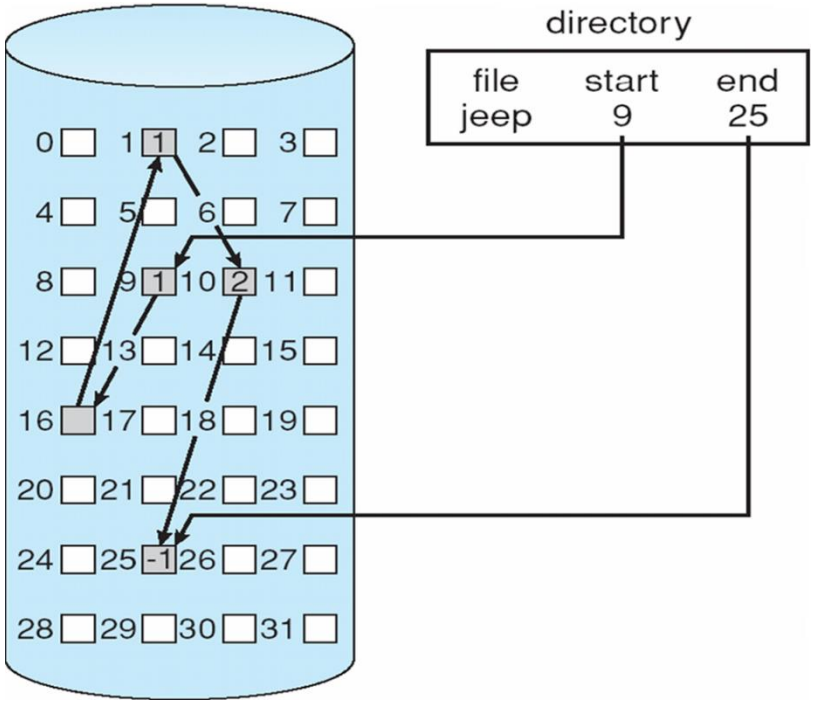
- 文件的目录项



- 特点：
  - 简单 — 只需要记录文件的起始位置（块号）及长度。
  - 访问文件很容易，所需的寻道时间也最少
- 缺点：
  - 为新文件找空间比较困难（类似于内存分配中的连续内存分配方式）
  - 磁盘碎片
  - 文件很难增长

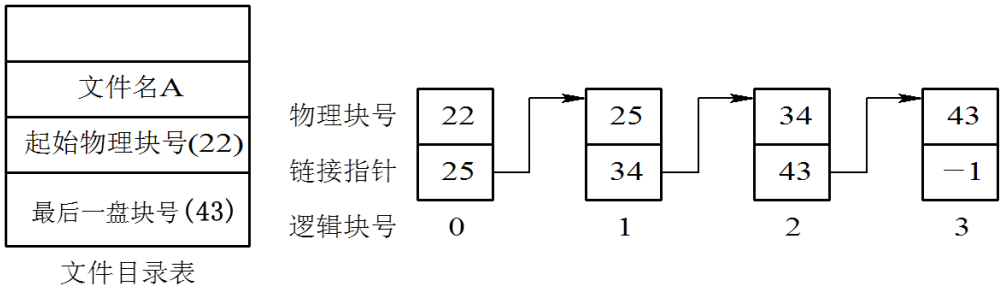
链接分配

- 基本思想
  - 文件的数据块存储到不连续的各个物理盘块中。通过链接指针将它们连接成一个链表。



链接分配

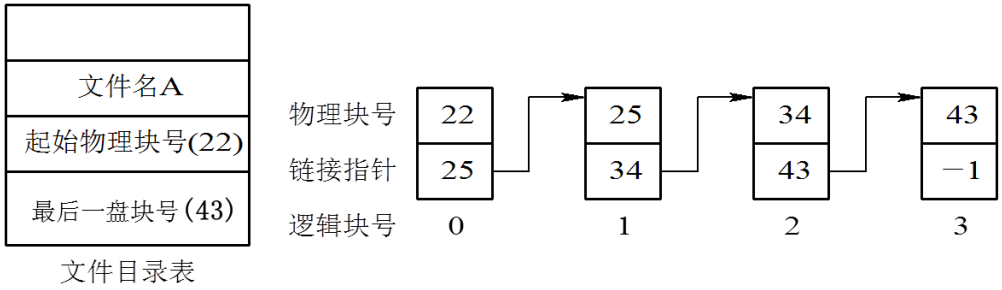
- 文件的目录项



- 优点：
  - 简单 — 只需起始位置
  - 文件创建与增长容易

隐式链接

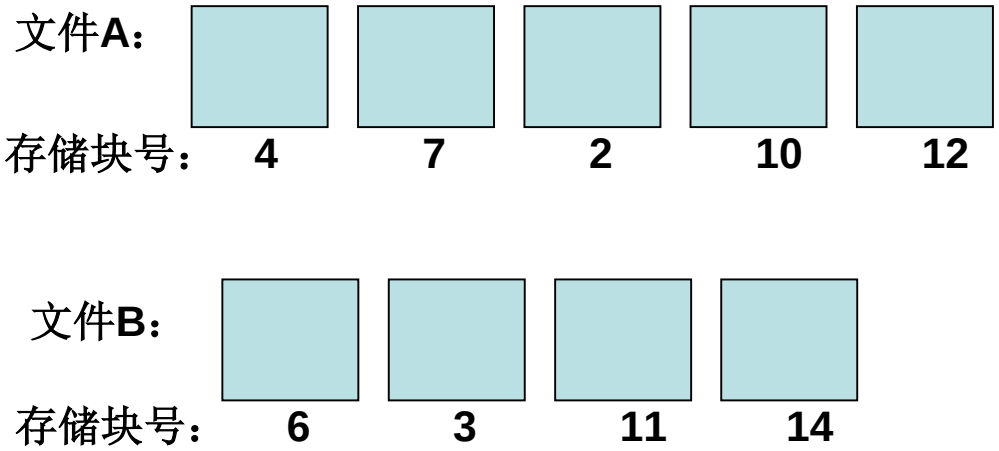
- 链接指针存放在文件的每个盘块中

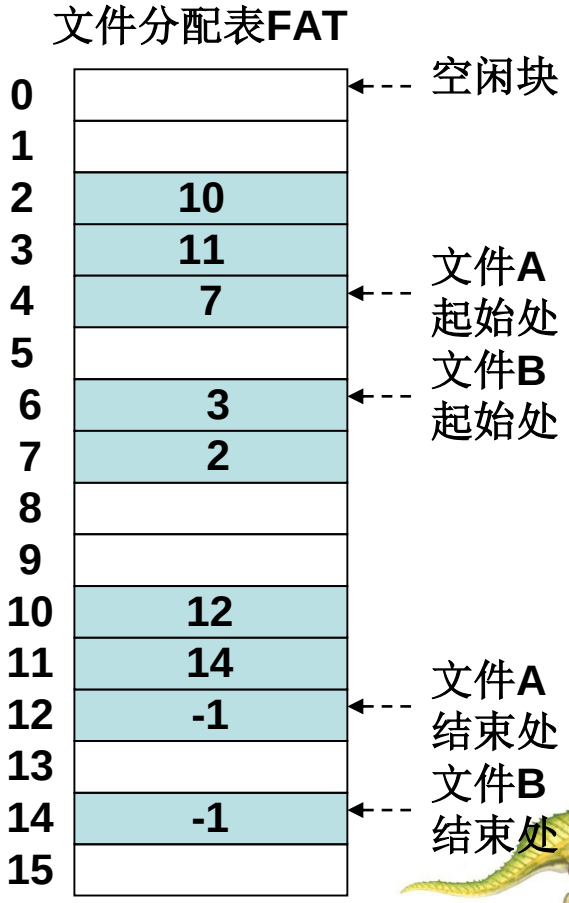


- 特点：
  - 便于扩展
  - 适用于顺序存取，不利于随机存取
- 缺点：
  - 不能随机访问
  - 块与块之间的链接指针需要占用空间
    - 簇：将多个连续块组成簇，磁盘以簇为单位进行分配
  - 存在可靠性问题
    - 指针丢失

显式链接

- 显式链接
  - 将各个盘块的链接指针存放在文件分配表FAT(File Allocation Table)中
  - 整个磁盘一张FAT





文件分配表(FAT)

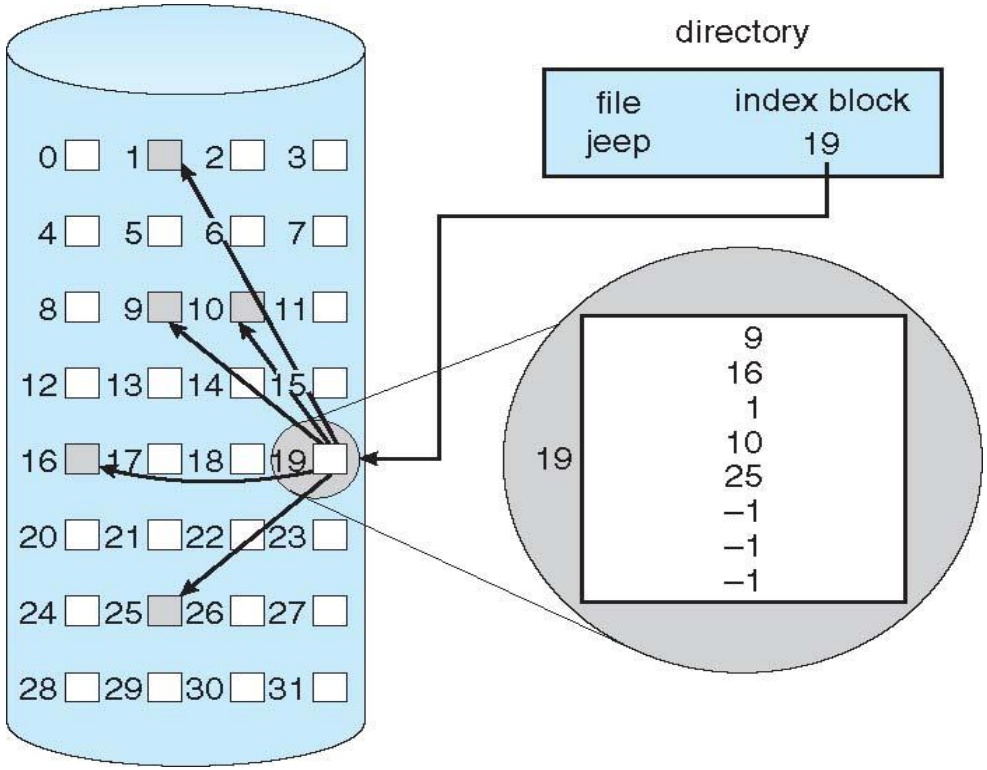
- 优点：
  - 便于随机存取
  - 查找在内存中进行，减少读盘次数
- 缺点：
  - FAT占用较大内存空间

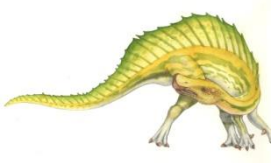
索引分配

- 基本思想
  - 文件的各个逻辑块存储到不连续的各个物理盘块中。
  - 为每个文件建立一张索引表，实现逻辑块和物理块的映射。
- 分类
  1. 单级索引分配
  2. 多级索引分配
  3. 混合索引分配

单级索引

- 基本思想
  - 为每个文件建立一张索引表，依次登记文件记录成组后存放的物理块号，称这种文件具有索引结构，或是索引文件。



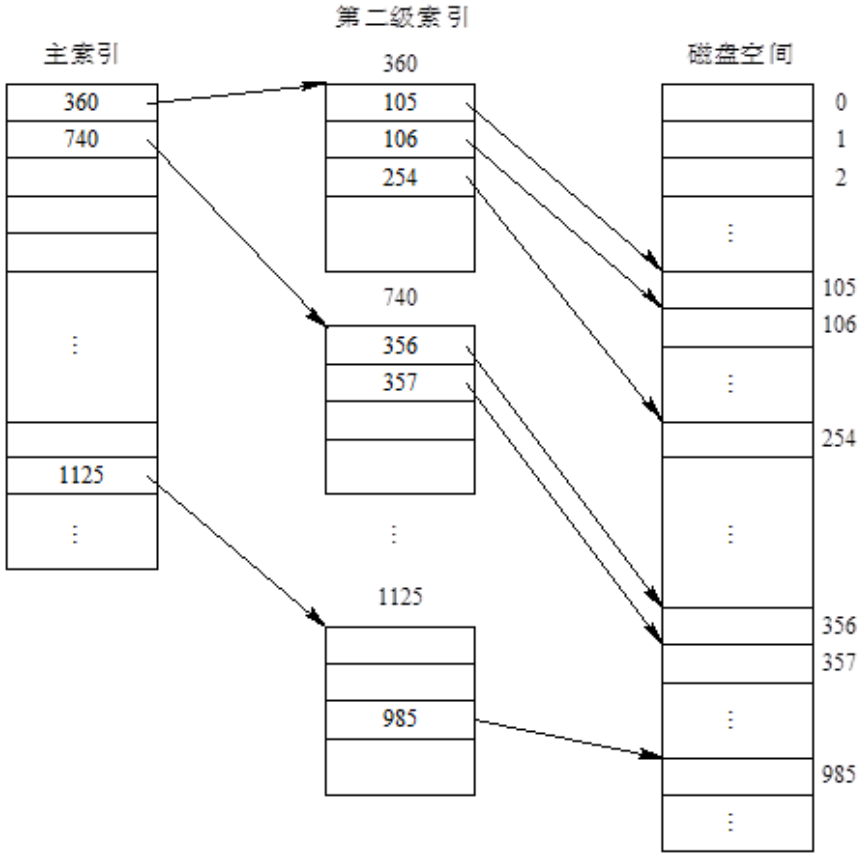


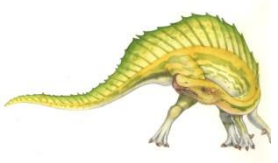
单级索引

- 优点
  - 支持直接访问
  - 不会产生外碎片；
- 缺点
  - 当文件较大时，索引块太多，查找速度减慢
    - 解决方案：建立多级索引

多级索引

- 基本思想
  - 为索引块建立索引表



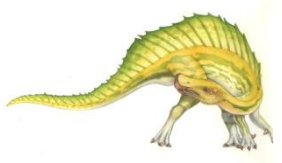


多级索引

- 设一个盘块大小为1KB，每个盘块号占4B。则：
  - 一级索引
    - 存放的文件的盘块号总数为：256
    - 文件的最大长度为：256×1KB=256KB
  - 二级索引
    - 存放的文件的盘块号总数为：256×256=64K
    - 文件的最大长度为：64K×1KB=64MB
  - 三级索引
    - 存放的文件的盘块号总数为：256×256×256=16M
    - 文件的最大长度为：16M×1KB=16GB

混合索引

- 引入
  - 单级索引分配不适用于大文件；
  - 多级索引分配不适用于小文件。
- 基本思想
  - 将多种索引分配方式相结合。
  - 通过多种索引形式访问不同规格的文件。
    - 直接地址：直接存放文件的盘块号
    - 一级索引：一次间接地址
    - 二级索引：二次间接地址
    - 三级索引：三次间接地址

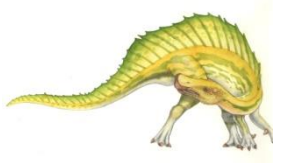


混合索引

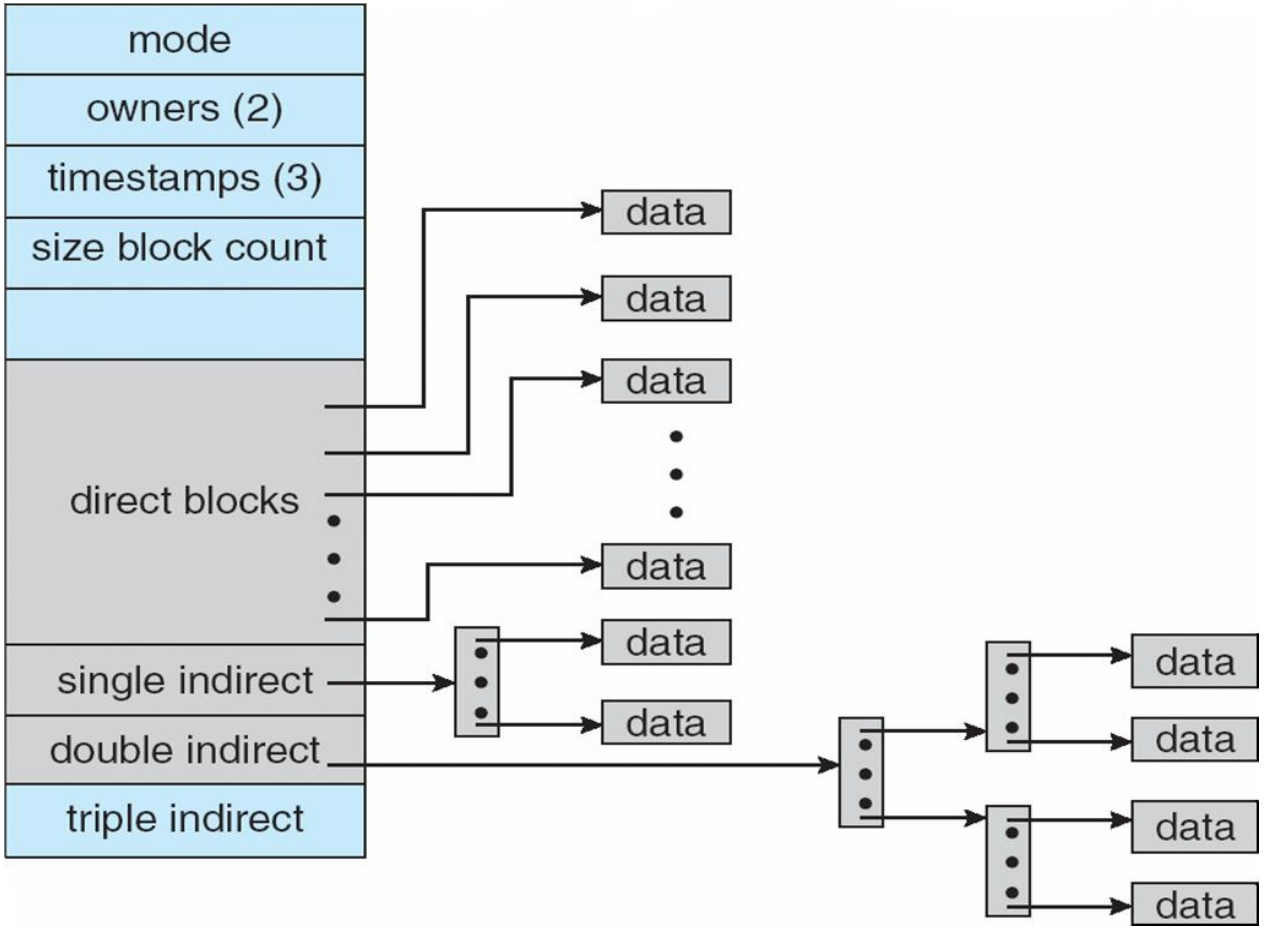
- 在UNIX系统的索引结点中共设置13个地址项：
  - 直接地址（10）：直接存放文件的盘块号
  - 一次间址（1）：一级索引
  - 二次间址（1）：二级索引
  - 三次间址（1）：三级索引

混合索引

- Ext2采用的是多级索引式结构
  - 通过该文件inode里的数组i_block[ ]，建立起文件的逻辑块号与相应物理块号间的对应关系，形成文件存储的索引表。
  - 该数组有15个元素，每个元素为一个索引项。
    - i_block[0]～i_block[11]：直接索引，直接给出文件数据存放的磁盘物理块号；
    - i_block[12]：一级间接索引；
    - i_block[13]：二级间接索引；
    - i_block[14]：三级间接索引。



混合索引


mode
owners (2)
timestamps (3)
size block count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data

混合索引

- 假设盘块大小为1KB，一索引项占4B，则
  - 小文件（<12KB)：采用直接地址立即读出。
  - 大型、中型文件（>12KB）：采用间址寻址

练习

- 在UNIX系统中，采用混合索引方式，其FCB中共有13个地址项，第0-9个地址项为直接地址；第10个地址项为一次间接地址；第11个地址项为二次间接地址；第12个地址项为三次间接地址。试写出下列文件的字节偏移量转换为物理块号和块内偏移的过程。（设盘块大小为1KB）
  - 9000
  - 14000

文件存储空间管理

- 目的
  - 记录存储空间的使用情况
  - 对空闲块进行组织和管理。

文件存储空间管理

- 几种常用的文件存储空间的管理方法
  1. 空闲表法和空闲链表法
  2. 位示图法
  3. 成组链接法

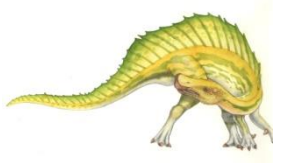
空闲表法

- 数据结构
  - 系统为外存上的所有空闲区建立一张空闲表。
  - 每个一个空闲表项对应于一块连续的存储空间；
  - 将所有空闲区按其起始盘块号递增的次序排列。

序号	第一空闲盘块号	空闲盘块数
1	2	4
2	9	3
3	15	5
4	—	—

空闲表法

- 分配过程
  - 它与内存管理中的动态分区分配方式类似
    - 首次适应算法
    - 循环首次适应算法
    - 最佳适应算法
- 回收过程
  - 类似于内存中动态分区中回收的方法（判断是否合并）



空闲链表法

- 数据结构
  - 空闲盘块链
    - 以盘块为单位连成一条链表
    - 优点：分配和回收的过程简单
    - 缺点：链表很长。
  - 空闲盘区链
    - 以盘区（一个盘区含多个盘块）为单位连成一条链表
    - 类似于内存分区分配与回收过程

成组链接法

- 空闲盘块的组织
  - 将当前文件区的所有空闲盘块分成组（如100块/组）
  - 用空闲盘块号栈存放当前可用的一组空闲盘块号
  - 每组的最后一个盘块(第100块)中记录了下一组空闲盘块的情况（空闲盘块数以及空闲盘块号）

成组链接法

- 分配过程
  - 从栈顶依次取出盘块号进行分配
  - 直到栈底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，再继续分配。
- 回收过程
  - 将回收的空闲盘块号依次记入到堆栈
  - 直到栈顶s.free(99)时，将空闲盘块栈中内容放入下一个新回收到的盘块中，将其作为栈底继续回收。

练习

- UNIX文件系统中有关盘块的分配与释放，是借助超级块中的栈进行的，假如某时刻超级块中的栈如图所示：
- 此时，某进程要释放3个物理块，其块号为150、152、160，试问：
  - 给出释放过程和释放后的堆栈的状况；
  - 此后，又有一个进程要求分配4个物理块，给出分配过程和分配后的状况。

s_nfree = 98
120#
121#
…
145#
210#

位示图法

- 位示图
  - 利用二进制的一位来表示磁盘中的一个盘块的使用情况。将所有盘块所对应的位构成的集合，称为位示图。
  - 通常可描述为一个二维数组m×n

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

位示图法

- 分配过程
  - 顺序扫描位示图，找到其值为“0”的二进制位。
  - 将二进制位（i行、j列）转换成对应的盘块号b。

    b=n*(i-1)+j　　（n：每行的位数）
  - 修改位示图：0→1
- 回收过程
  - 将回收盘块的盘块号b转换成位示图中的二进制位（行号i、列号j）

    i=(b-1) DIV n + 1

    j=(b-1) MOD n + 1
  - 修改位示图：1→0

位示图法

- 特点
  - 容易实现
  - 位示图占用空间少，可放入内存，易于访问。

练习

- 例：有一计算机系统利用所示的位示图来管理空闲盘块，盘块的大小为1KB，现要为某文件分配两个盘块，说明盘块的分配过程。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0