第11章 文件系统实现

目录

文件的用户视图和系统视图

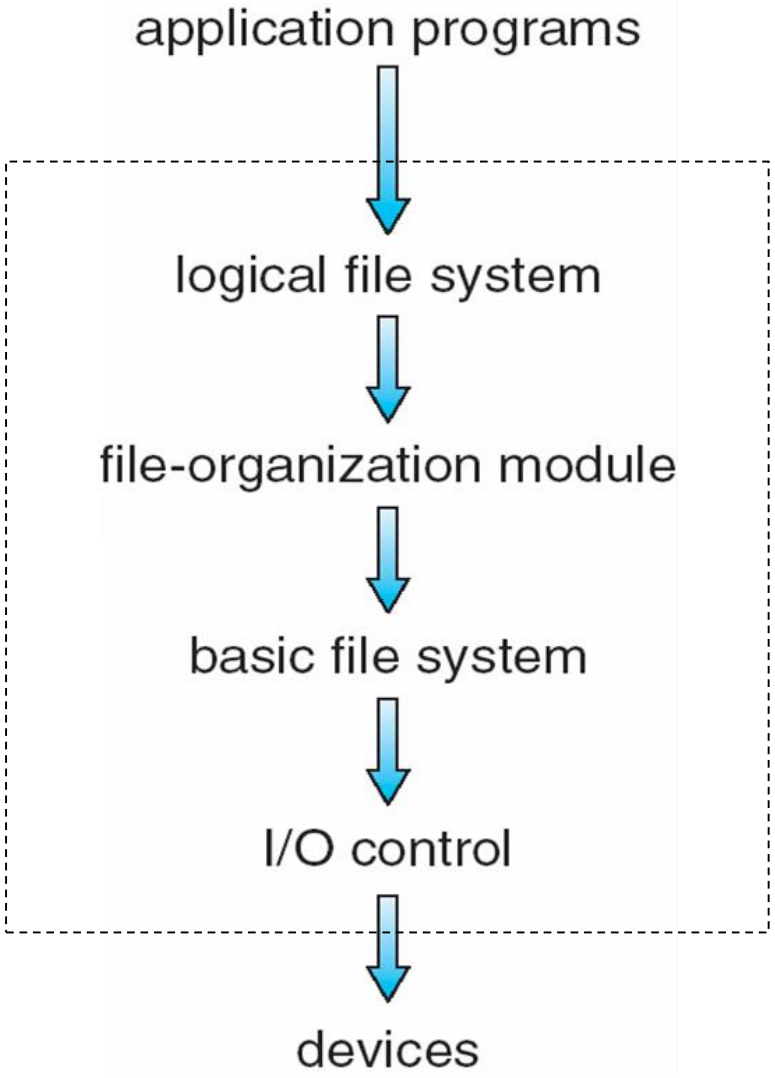
- 用户视图
  - 文件管理的数据结构
  - 通过系统调用访问文件，不关心磁盘的存储方式
- 操作系统视图
  - 数据块集合
  - 数据块是逻辑存储单元，扇区是物理存储单元

用户视图到系统视图的转换

- 读写文件
  - 获取数据块
  - 缓存数据块
  - 修改数据块
  - 写回数据块
- 文件系统中的基本操作单位是数据块
  - getc putc即使每次只访问1个字节，也需要缓存目标数据块，如4096B

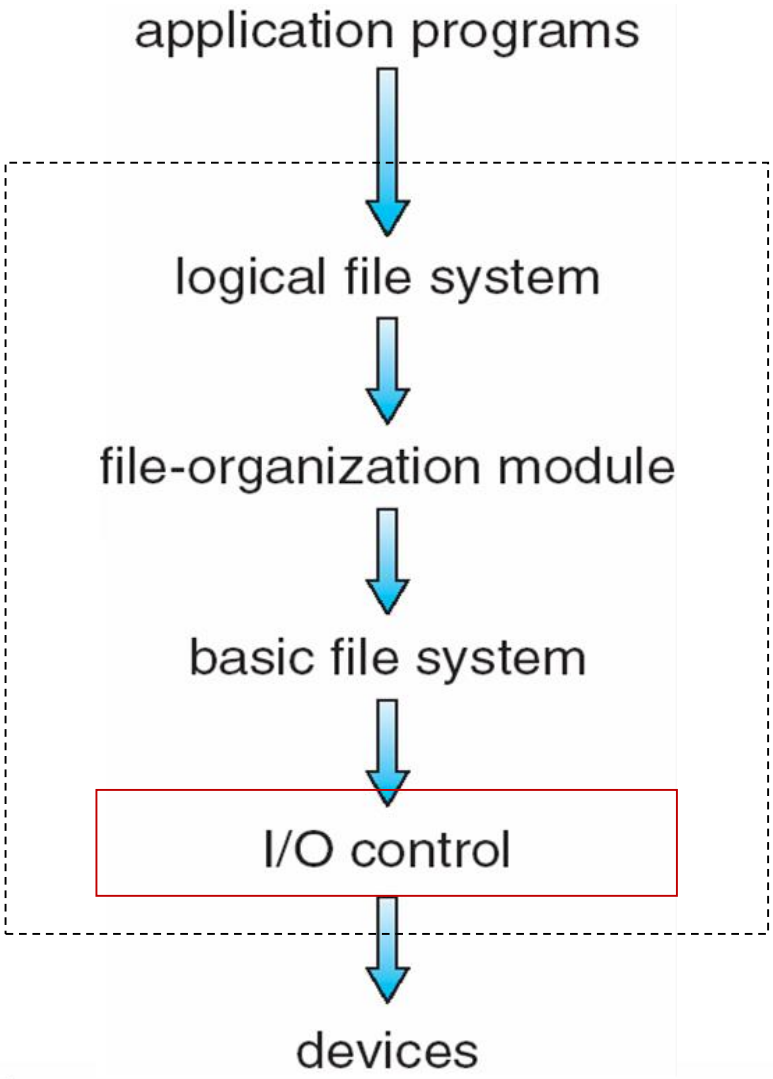
分层设计的文件系统

- 文件系统是分层组织的：
  - I/O控制层
  - 基本文件系统层
  - 文件组织模块层
  - 逻辑文件系统层



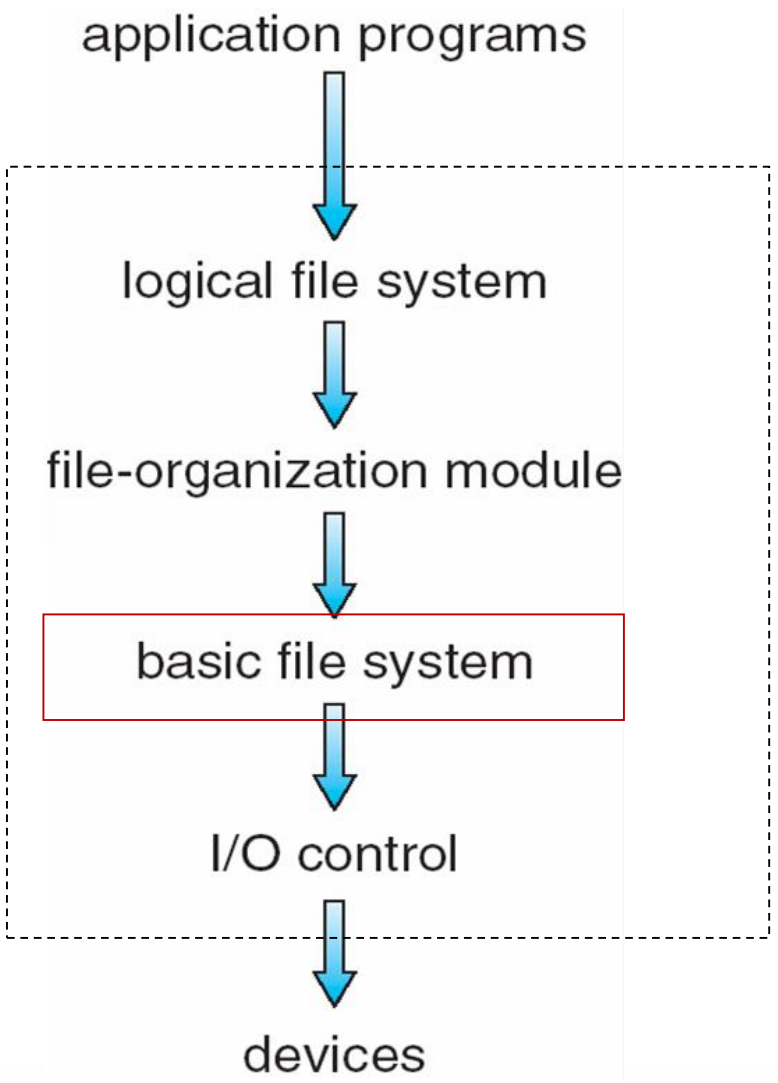
文件系统层次

- I/O控制层(I/O Control)
  - 通过设备驱动和中断处理实现对I/O设备的管理
  - 设备驱动程序会对来自于上层的指令进行翻译，并向硬件控制器发出具体的硬件相关的指令



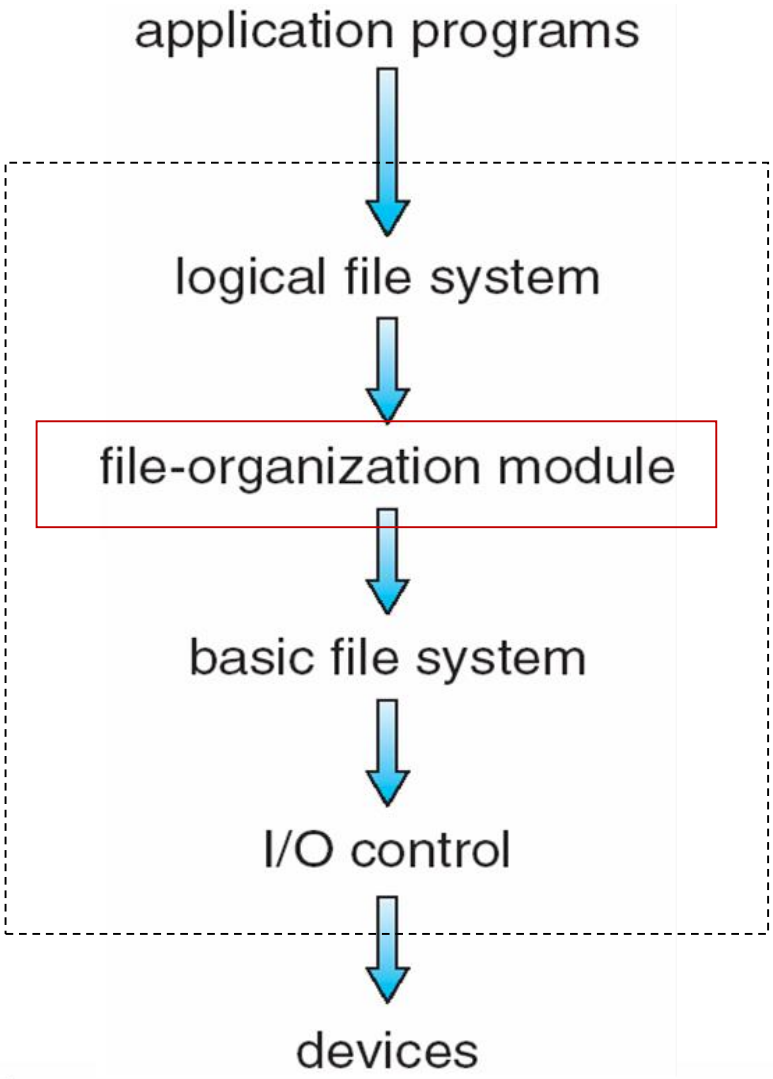
文件系统层次

- 基本文件系统(Basic File System)
  - 向设备驱动程序发送通用命令，从而读取和写入磁盘的物理块
  - 管理内存缓冲区和缓存（分配、回收、替换等）
    - Buffer：存储传输时的缓冲数据
    - Cache：存储频繁使用的文件系统元数据，如文件系统、目录、数据块缓存



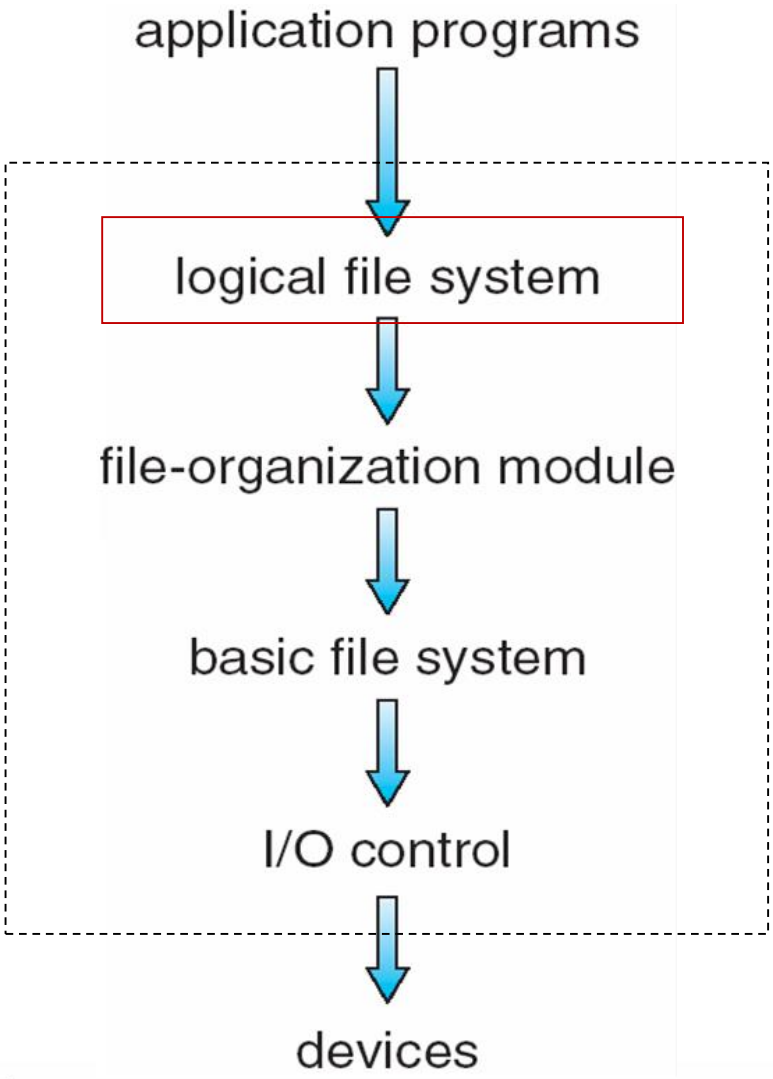
文件系统层次

- 文件组织模块(File-organization Module)
  - 该层知道文件、以及相关的逻辑块和物理块
  - 将逻辑地址转换成物理地址
  - 管理空闲空间、磁盘分配等



文件系统层次

- 逻辑文件系统(Logical File System)
  - 负责管理元数据信息，即文件系统的所有结构
  - 根据符号文件名，管理目录结构
  - 根据文件控制块，将文件名翻译成文件标识号、文件句柄和具体位置等，如inode
  - 负责保护和安全



文件系统层次

- 分层结构可有效减少复杂性和冗余，但增加了额外开销，导致性能降低。

- 典型的文件系统
  - ISO9660(CD-ROM)、UFS(UNIX)、FAT32、NTFS、ext2、ext3、GoogleFS

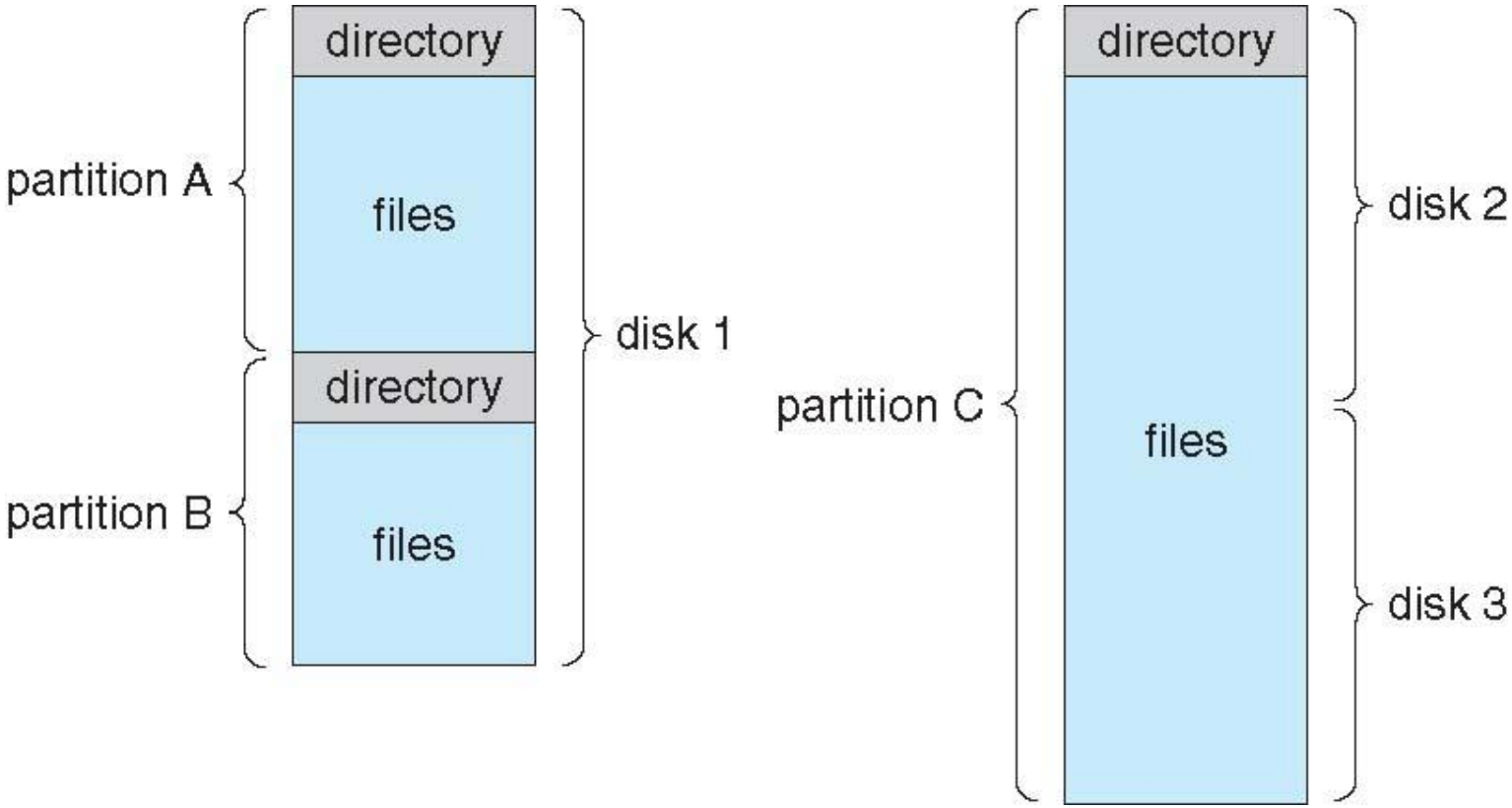
目录

1. 磁盘结构的几个常识

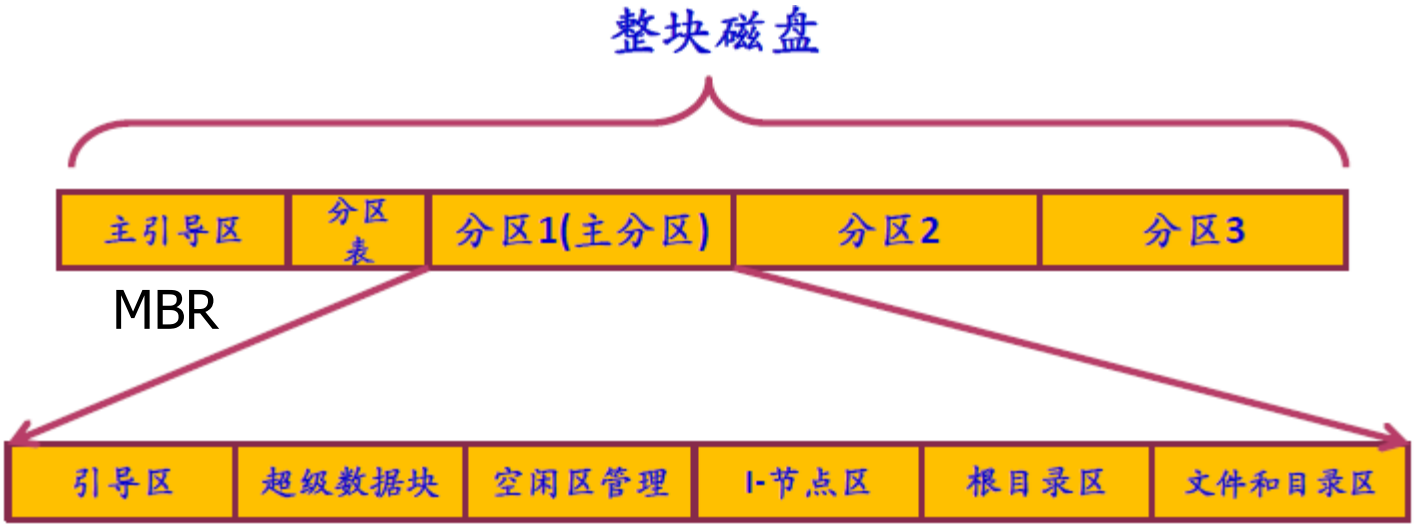
- **基本磁盘和分区 partitions**
  - 磁盘能被分割成分区（也可以叫做基本卷）
  - 通过格式化，可以在分区上建立文件系统

- **动态磁盘或卷 volume**
  - 在动态磁盘中，分区叫做卷。
  - 可以跨磁盘建立卷。
  - 卷又分简单卷、跨区卷、带区卷、镜像卷、RAID-5卷。
  - 动态磁盘具有更高的数据管理和容错能力。

典型的文件系统组成


directory
files
partition A
directory
files
partition B
disk 1
directory
files
partition C
disk 2
disk 3

磁盘上文件系统的布局



UNIX文件系统布局

Windows的FAT文件系统布局

2. 文件系统的盘上和内存结构

- 实现文件系统要使用多个磁盘和内存结构
  - 盘上结构：
    - 告知如何启动OS、总的块数、空闲块情况、目录结构、具体的文件
  - 内存结构：
    - 用于管理文件系统并通过缓存提高性能

（1）盘上结构

- 引导控制块，Boot control block
  - 包含从相应卷引导操作系统的所需信息，通常为卷的第一块，若为空，则无OS
    - UFS: boot block 引导块
    - FAT: Partition Boot Sector 引导区
    - NTFS: Partition Boot Sector 引导区

- 卷控制块，Volume Control Block
  - 包含卷/分区的详细信息：块的总量、块的大小、空闲块数量和指针、空闲FCB数量及指针
    - UFS: Superblock 超级块
    - FAT：FAT 文件分配表
    - NTFS: Master File Table 主文件表

（1）盘上结构(cont.)

- 目录结构：Directory Structure
  - 组织所有的文件
    - UFS：目录文件，包含文件名和inode号码,
    - FAT：根目录、其他目录文件
    - NTFS：存储于Master File Table中

- 文件控制块：File Control Block，FCB
  - 每个文件对应一个，包含保存文件属性信息的数据结构
    - UFS：inode区，包含inode号、权限、大小、日期等
    - FAT：根目录或其他目录中的目录项
    - NTFS：存储在Master File Table中，使用关系数据库存储，每个文件占一行

（2）内存结构

- 安装表，Mount Table
  - 存储文件系统安装卷、安装点、以及文件系统类型
- 目录结构缓存
  - 最近访问目录的信息
- 系统打开文件表，System-wide open-file table
  - **每个打开文件**的FCB副本以及其他信息
- 进程打开文件表，per-process open-file table
  - **指向与该进程相关的**系统打开文件表中已打开文件的条目的指针，以及相关信息

（2）内存结构：系统调用open()

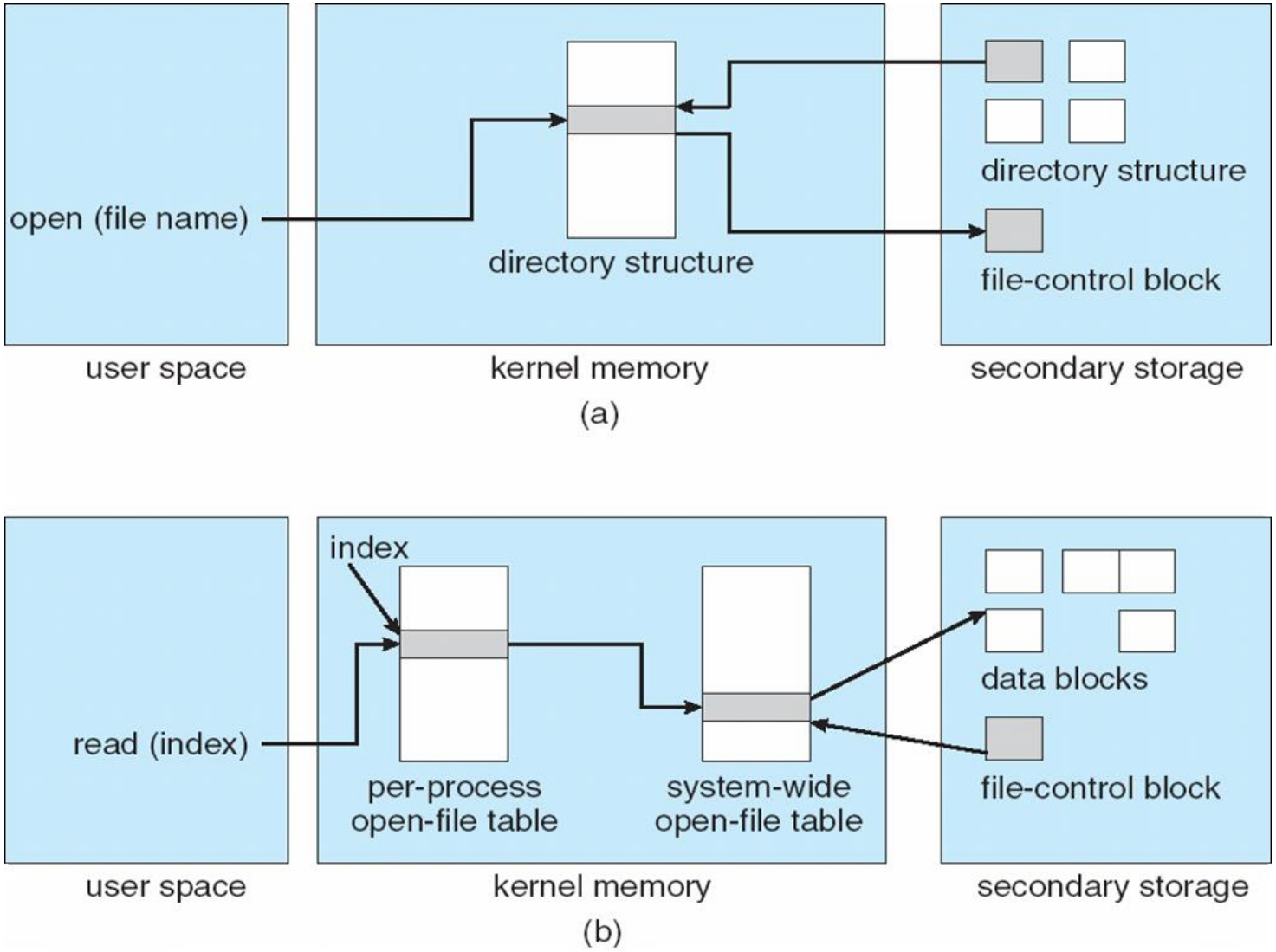
- 系统调用open()，文件名被传递到文件系统
- 查找系统打开文件表，确认这个文件是否被其他进程使用中
  - ① 如果是，就在**进程打开文件表中创建一个条目**，让其指向系统打开文件表，系统打开文件表中打开计数加一。
  - ② 如果不是，则根据给定文件名**搜索目录结构**，目录结构往往缓存在内存中，以加快速度。
    - ① 找到文件之后，就将他的FCB复制到系统打开文件表中。
    - ② 进程打开文件表中会创建新条目，指向系统打开文件表的对应条目。
- 返回进程打开文件表的对应指针，read()等就能使用了

（2）内存结构：系统调用close()

- 当文件关闭时进程打开文件表的相应条目**会被删除**，整个系统表的相应条目的**打开计数会递减**。

- 当所有打开该文件的用户关闭它时，任何更新的元数据会被复制到基于磁盘的目录结构，并且整个系统的打开文件表的条目会被删除。

(2) 内存结构


open (file name)
directory structure
directory structure
file-control block
user space
kernel memory
secondary storage
(a)
index
read (index)
per-process
open-file table
system-wide
open-file table
data blocks
file-control block
user space
kernel memory
secondary storage
(b)

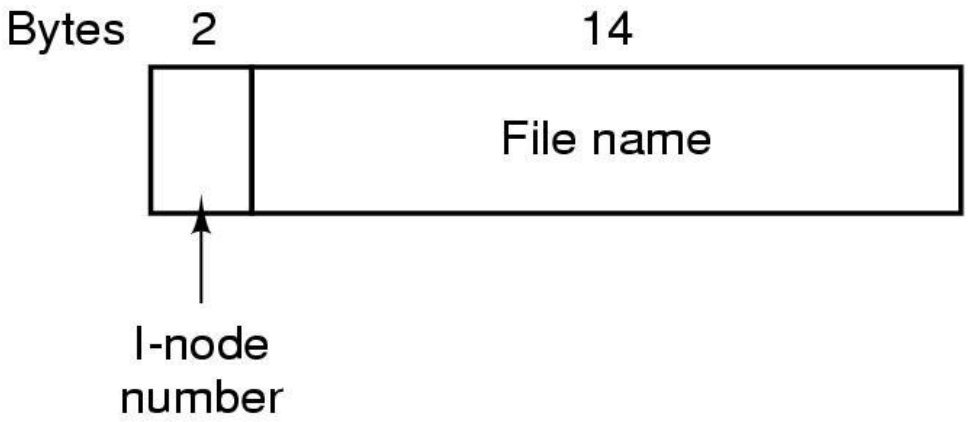
补充：索引结点

1) 索引结点的引入：为了方便检索，把FCB中的文件名和文件描述信息分开，即文件的描述信息单独形成一个称为索引节点的数据结构（简称/节点），而目录文件中的每个目录项仅由文件名和指向该文件对应索引节点的指针构成。

文件名	索引结点编号
文件名1	
文件名2	
⋮	⋮

UNIX的文件目录

UNIX中FCB: inode



目录文件的内容

文件名 14B	i-node # 2B
……	……
mytest.c	56
……	……

磁盘上的i-node表

i-node #	文件属性
56	存取权限，大小，文件主，建立/修改日期，磁盘地址 等
……	……

64B

inode的好处是什么?

- 引入索引结点
  - 按文件名检索目录文件时，只用到了文件名。当找到该文件名时，才需要它的其它描述信息。
  - 所以在把存放该目录文件的盘块从外存调入内存进行比较时，应使一个盘块中包含尽量多的文件名，以减少启动磁盘次数， 加快按名存取的速度。

举例

- 例：设物理块大小为512B，某目录下有128个文件。
  - 若采用简单FCB，设FCB占64B
    - 则每物理块能容纳 512/64=8 个FCB
    - 该目录文件需占 128/8 = 16 块
    - 查找一个文件的平均访盘次数为：（1+16）/ 2 = 8.5 次。
  - 若采用inode，目录项占16B(文件名+inode号)，inode部分64B
    - 每物理块能容纳 512/16=32个目录项
    - 每个物理块容纳 512/64=8个inode
    - 该目录文件需占 128/32 =4 块，inode部分需占 128/8=16 块
    - 查找一个文件的平均访盘次数为：（1+4）/ 2 + 1 = 3.5 次。

3. 目录实现

- 目录分配与目录管理的算法选择会影响文件系统效率、性能和可靠性
- 线性列表
  - 文件名+数据块指针，用文件名作为检索关键词
  - 管理简单，但搜索费时
    - 线性搜索时间，可采用软件缓存提高搜索效率
    - 排序列表可减少平均搜索时间，但需比较大代价维护文件的创建删除等问题，可用B+树等
- 哈希表
  - 线性列表+哈希数据结构
  - 将文件名哈希，减少目录搜索时间
  - 需要考虑碰撞问题

目录

文件存储空间的分配方法

- 分配的目标：
  - 如何为文件分配磁盘块？
  - 如何实现有效使用磁盘空间，且快速访问文件？

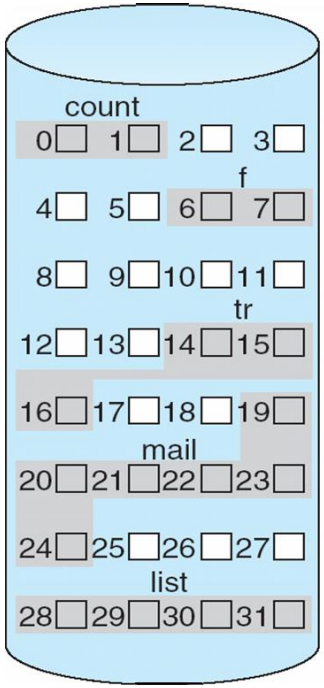
- 主要分配方法
  - 连续分配
  - 链接分配
  - 索引分配

1. 连续分配

- 连续分配，Contiguous allocation
  - 每个文件占用一组连续的盘块
  - 优点：
    - 在大多数情况下性能最优：寻道数最小
    - 访问简单：对于顺序访问和直接访问，只需要上次访问的磁盘位置和长度
  - 问题：
    - 文件信息：需要知道文件占用多大空间
    - 分配算法：首次适应，最优适应
    - 外部碎片问题：紧凑合并技术，时间代价大
    - 文件动态增长问题：扩展空间？

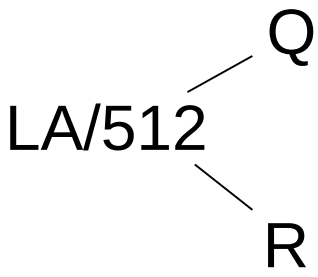
连续分配示意图

- 从逻辑地址映射到物理地址



directory

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



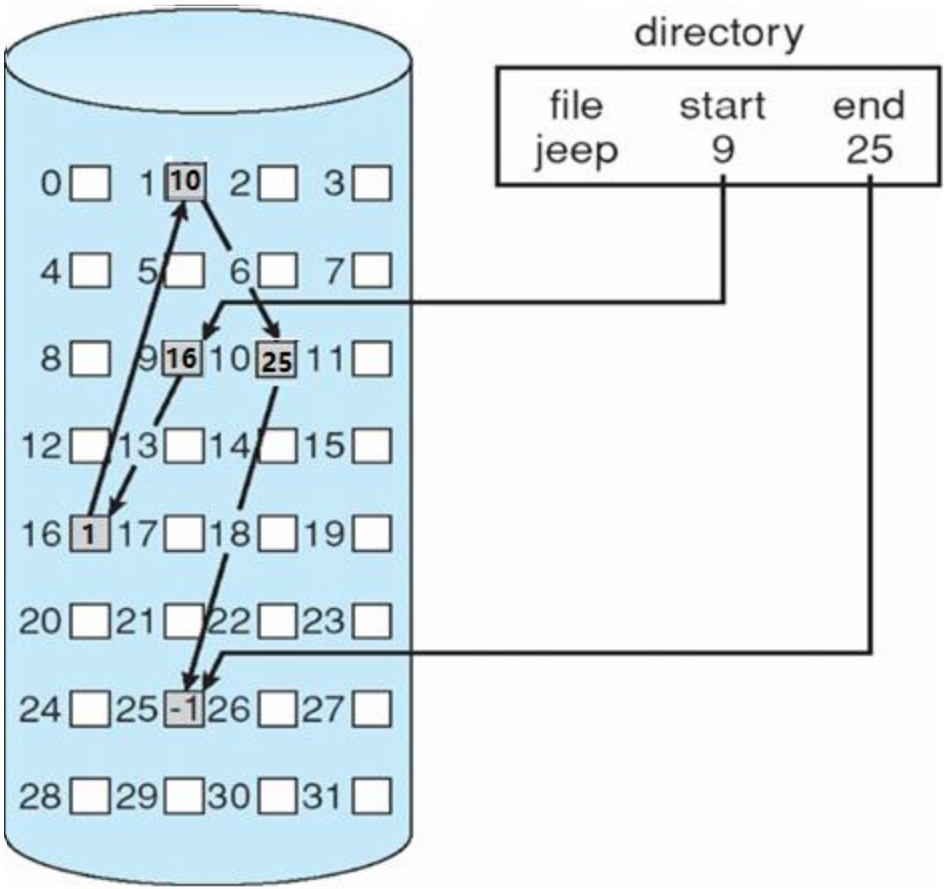
物理块号=Q+起始地址

块内偏移地址=R

2. 链接法

- 链接法：
  - 一个文件的信息存放在若干不连续的物理块中，各块之间通过指针连接，前一个物理块指向下一个物理块。
  - 每个块包含了指向下一块的指针，文件以空指针为结尾
  - 优点：
    - 提高了磁盘空间利用率；不存在外部碎片问题；有利于文件插入和删除；有利于文件动态扩充
  - 问题
    - 不支持直接访问，需遍历链表
      - 定位某个块需要许多I/O和磁盘寻道
    - 记录指针需要存储开销
      - 由按块改为采用分簇（一簇对应多块）方法，减少块数降低开销，但是引入内部碎片
    - 可靠性：指针丢失或损坏问题

链接法示意图

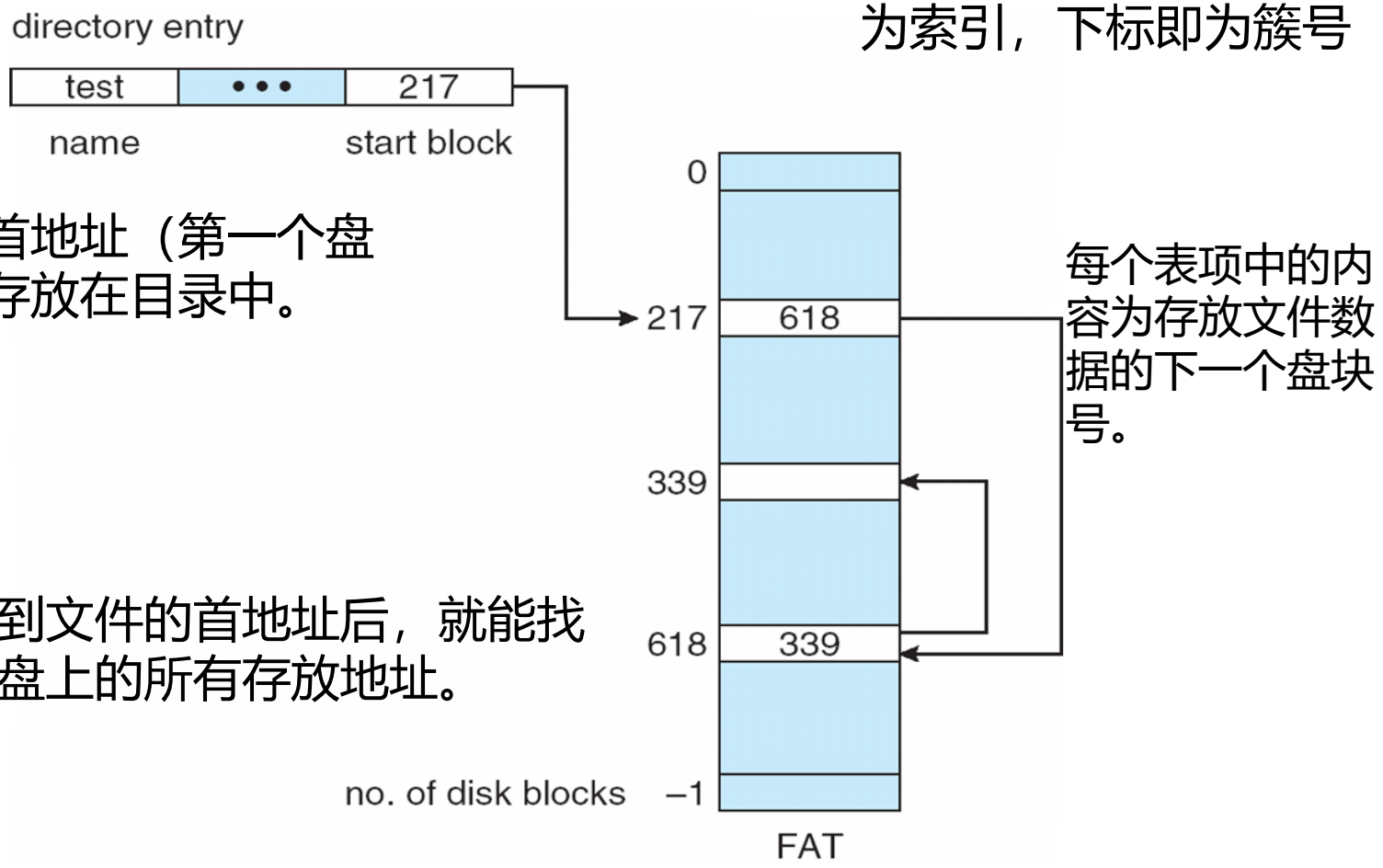

directory
file
start
end
jeep
9
25
0
1
10
2
3
4
5
6
7
8
9
16
10
25
11
12
13
14
15
16
1
17
18
19
20
21
22
23
24
25
-1
26
27
28
29
30
31

链接法实例：FAT

- FAT (File Allocation Table)
  - 文件分配表FAT是以链接方式存储文件的系统中记录磁盘分配和跟踪空白盘块的数据结构。
  - 用于MS-DOS，每个卷开头的磁盘用于存储该表
  - 表中，每个盘块都有一个FAT表项，并可按块号索引，每个表项中的内容为存放文件数据的下一个盘块号。
  - 使用方法类似链表
    - 目录项包含文件首块的块号，根据该块号索引的FAT表项包含文件的下一块块号
    - FAT如果不采用缓存，会带来大量的寻道时间，磁头必须不断移到卷开头，读入FAT，找到所需块位置，再移到块本身位置

链接法实例： FAT

FAT表以盘块（簇）号为索引，下标即为簇号



文件的首地址（第一个盘块号）存放在目录中。

每个表项中的内容为存放文件数据的下一个盘块号。

从目录中找到文件的首地址后，就能找到文件在磁盘上的所有存放地址。

文件分配表例

- 假定磁盘块的大小为1KB，对于1.2MB的软盘，其文件分配表FAT需要占用多少存储空间？若硬盘容量为200MB时，FAT需要占用多少空间?
  - 软盘大小为1.2MB，磁盘块的大小为1KB，所以该软盘共有盘块：1.2M/1K=1.2K（个）
  - 又　1K＜1.2K＜2K，
  - 故1.2K个盘块号要用11位二进制表示，为了方便存取，每个盘块号用12位二进制描述，即文件分配表的每个表目为1.5个字节。
  - FAT要占用的存储空间总数为：
  - 　1.5×1.2K=1.8KB

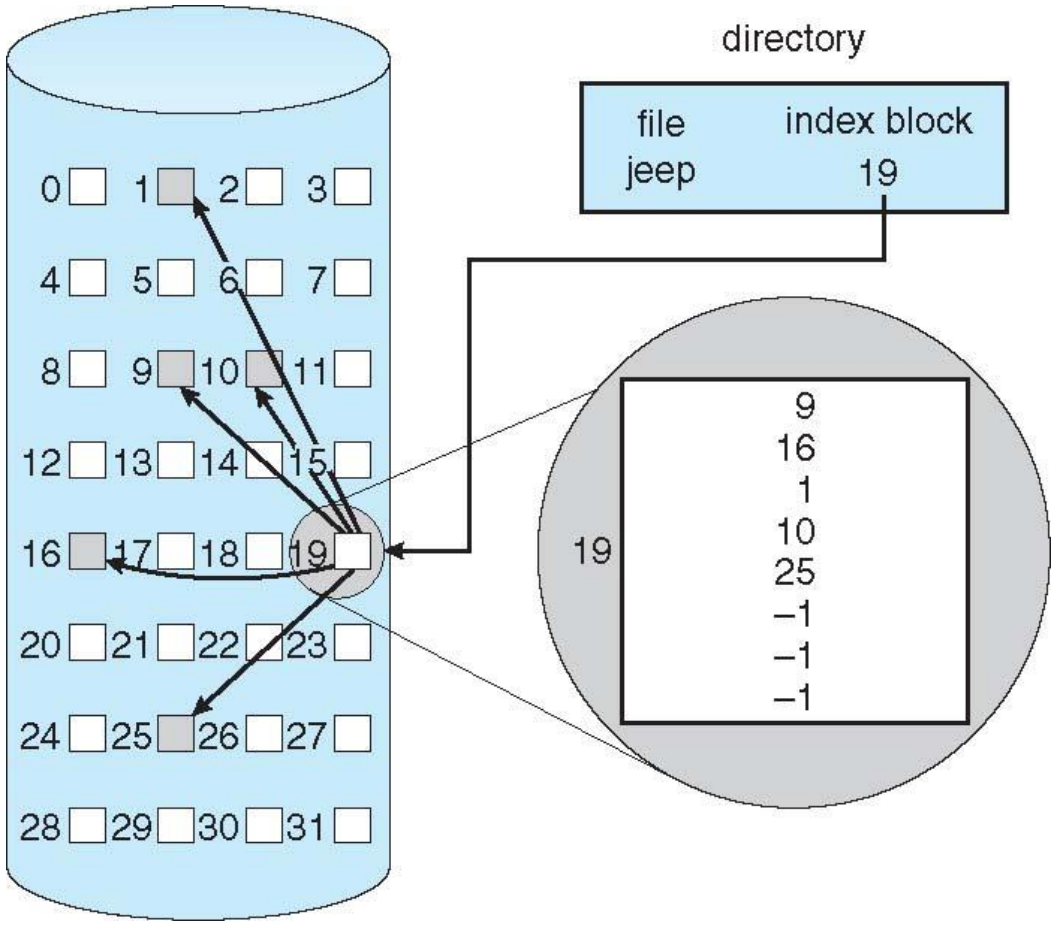
文件分配表例

- 假定磁盘块的大小为1KB，对于1.2MB的软盘，其文件分配表FAT需要占用多少存储空间？若硬盘容量为200MB时，FAT需要占用多少空间?
  - 若硬盘大小为200MB，硬盘共有盘块：
    200M/1K=200K
  - 又 128K＜200K＜256K，
  - 故200K个盘块号要用18位二进制表示。为方便文件分配表的存取，每个表目用20位二进制表示，即文件分配表的每个表目大小为2.5个字节。
  - FAT要占用的存储空间总数为：
    2.5×200K=500KB

3. 索引分配

- 链接分配方式虽解决了连续分配方式中存在的问题，但又出现了新的问题：
  - 不支持随机存取
  - 链接指针要占用一定数量的磁盘空间
- 在索引分配方法中，系统为每个文件分配一个索引块，索引块中存放索引表，索引表中的每个表项对应分配给文件的一个物理块。

索引分配示意图


directory
file
jeep
index block
19
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
19
9
16
1
10
25
−1
−1
−1

索引分配的特点

优点：

- 索引分配方法支持直接访问，不会产生外部碎片；

问题：

- 索引块要占用一定的存储空间
- 存取文件需要两次访问外存。

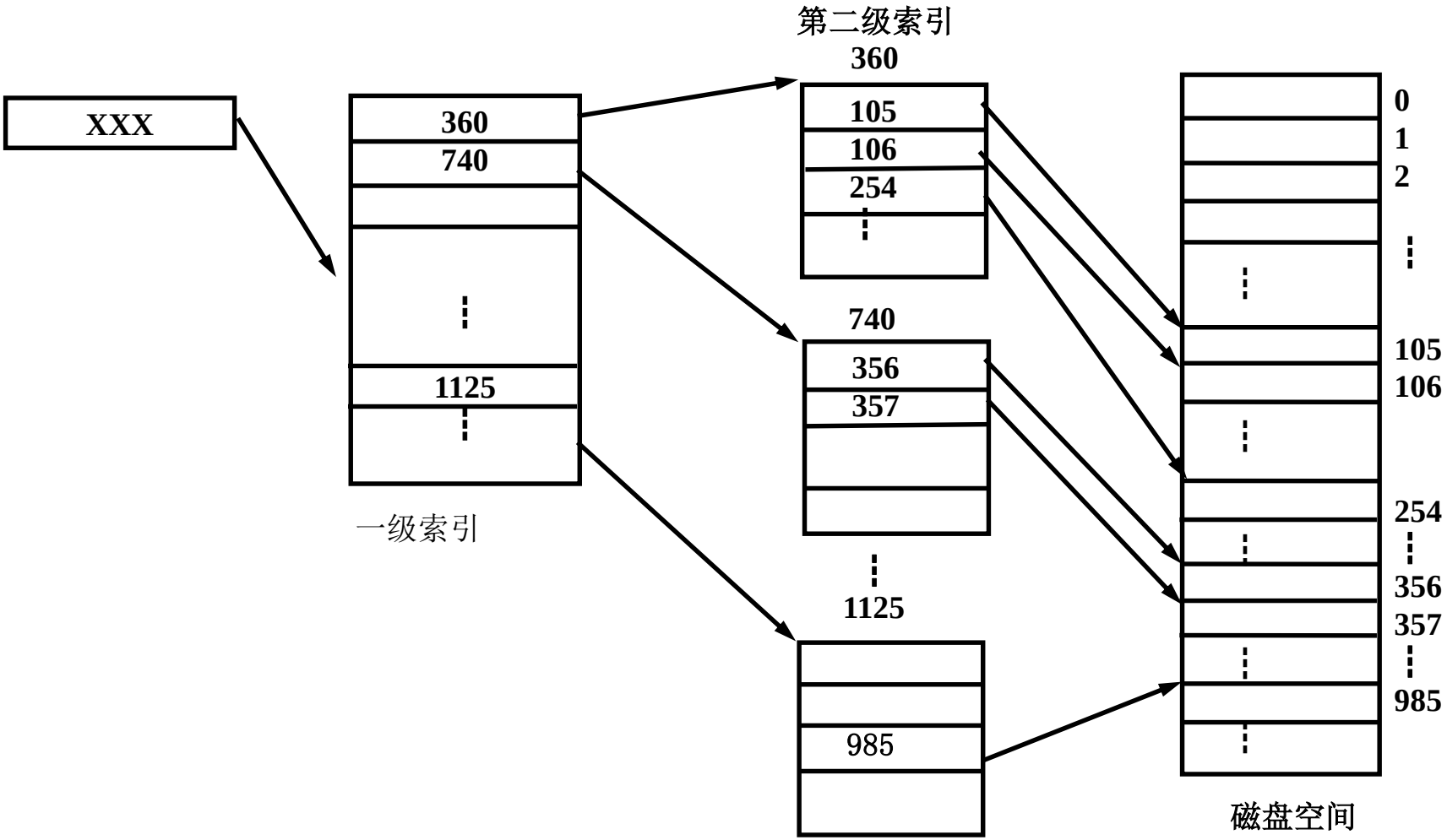
- 采用索引分配，即使只有一个或两个指针非空，也需要分配完整的索引块，浪费空间！
  - 索引块应为多大？越小越好吗？
  - 对于大文件，应该采用什么样的机制来设计索引？

二级索引和多级索引

- 当文件很大，其索引表的大小超过了一个物理块时，可以将索引表本身作为一个文件，再为其建立一个“索引表”，该“索引表”是文件索引的索引，从而构成了二级索引。

- 第一级索引表的表目指向第二级索引，第二级索引表的表目指向文件信息所在的物理块号。以此类推可再逐级建立索引，进而构成多级索引。

两级索引分配示意图


第二级索引
360
XXX
360
740
1125
一级索引
105
106
254
740
356
357
1125
985
0
1
2
105
106
254
356
357
985
磁盘空间

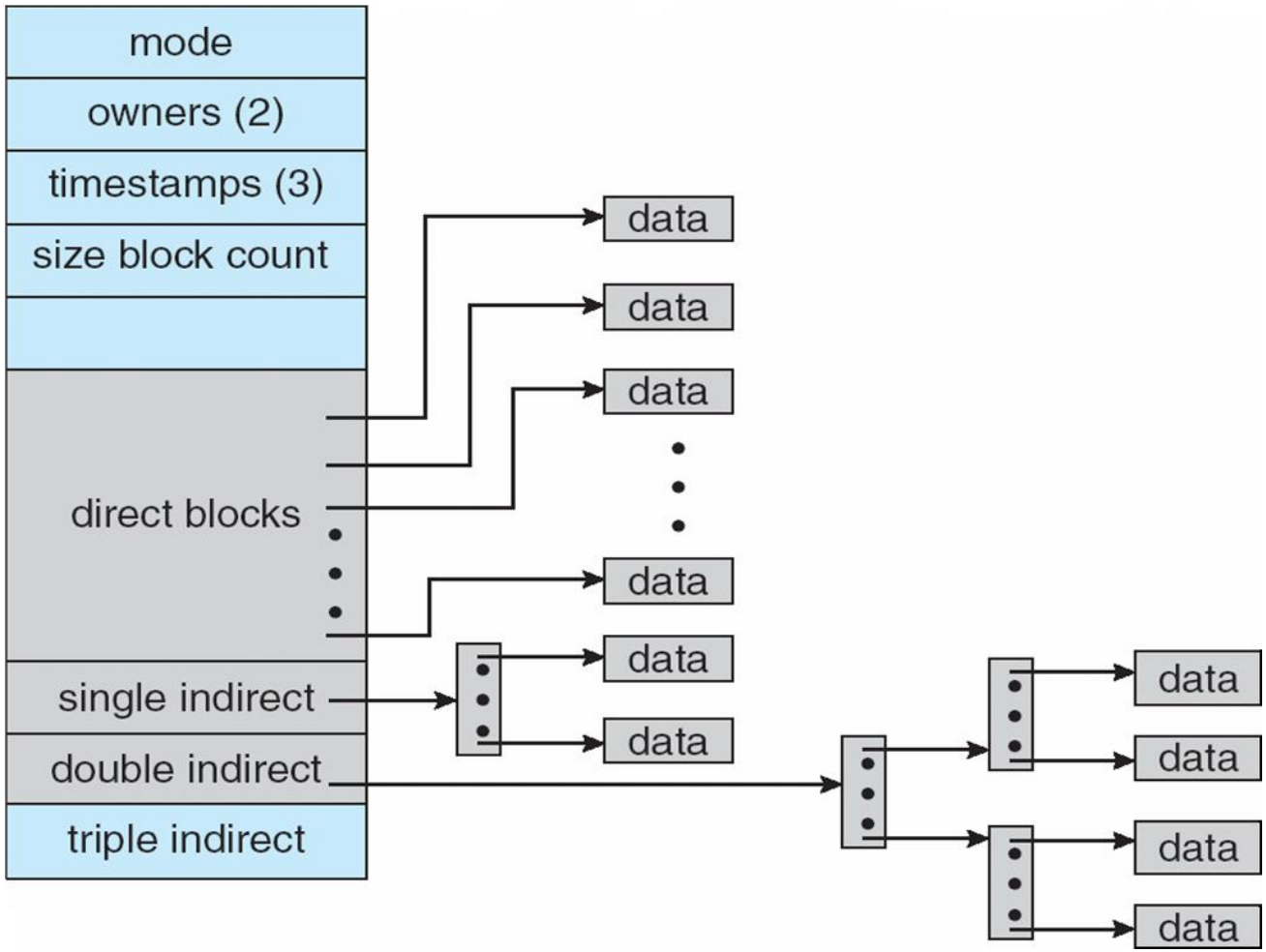
两级索引分配允许的文件最大长度

- 在两级索引分配方式下，如果每个盘块的大小为1KB，每个盘块号占4字节，则：
- 一个索引块中可以存放：
  1KB/4B=256个盘块号
- 两级索引最多可以存放的盘块数为：
  256×256=64K个盘块号
- 因此可以允许的最大文件长度为：
  64K×1KB=64MB

混合索引分配实例

- 混合索引分配方式是将多种索引分配方式相结合而形成的一种分配方式。这种方式已用于UNIX、Linux等系统中。
- 在UNIX中，共设有15个地址项，包括12个直接地址项、1个一次间接地址项、1个二次间接地址项和1个三次间接地址项。
- 假定一个盘块的大小为4KB，32位地址空间，一个盘块号占4字节。

混合索引方式示意图


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
data

混合索引方式寻址范围

- 直接地址：
  - 为了提高对文件的检索速度，在索引节点中建立了12个直接地址项，每个地址项中存放相应文件所在的盘块号。
  - 寻址范围：12×4KB=48KB
- 一次间接寻址：
  - 这里第一级存放的不是存储文件数据的盘块号，而是先将多个盘块号存放在一个磁盘块中，再将该磁盘块的块号存放在一次间接地址项中。
  - 盘块大小为4KB，若一个盘块号占4字节，则一个盘块中可以存放下：4KB/4B=1K个磁盘块号。
  - 寻址范围为1K×4KB=4MB
- 二次间接寻址
  - 寻址范围：1K×1K×4KB=4GB
- 三次间接寻址
  - 寻址范围：1K×1K×1K×4KB=4TB
- 总支持文件的大小空间4TB^{+}

课堂练习（考研真题）

- 设文件索引节点中有7个地址项，其中4个地址项是直接地址索引，2个地址项是一级间接地址索引，1个地址项是二级间接地址索引，每个地址项大小为4字节。若磁盘索引块和磁盘数据块大小均为256字节，则可表示的单个文件最大长度是多少？

目录

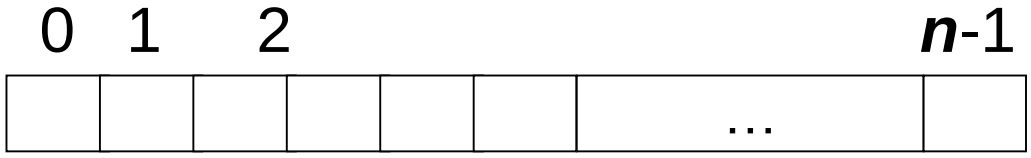
空闲存储空间管理

- 为了实现文件存储空间的分配，首先应记住空闲存储空间的情况。

- 常用的空闲存储空间管理方法有：
  - 位图法(Bit map)
  - 链接法(Linked List)
  - 分组法(Grouping)
  - 计数法(Counting)

(1)位图法：Bit vector or bit map

- 文件系统维护一个空闲空间列表去跟踪可用的区块/簇



$$bit[i] = \begin{cases} 1，表示block[i]\ 空闲 \\ 0，表示block[i]\ 被占用 \end{cases}$$

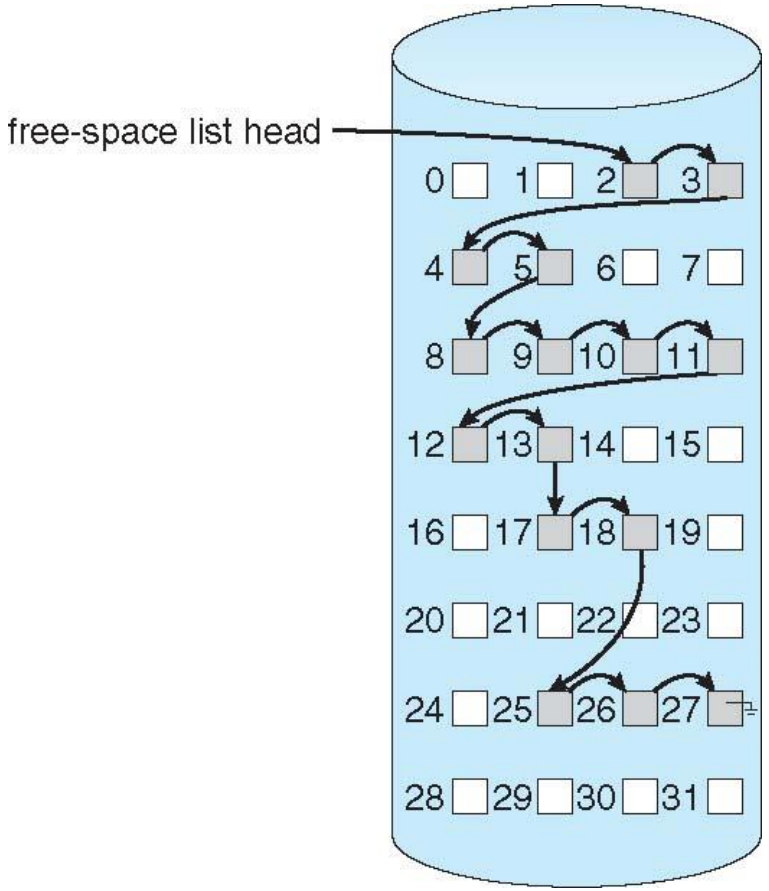
- 寻找依据：找到第一个为1的位，其指向的块即空闲块
- 空闲块号的计算方法
  - 字长 ×(值为0的字的个数) + 第一个为1的位的偏移
- 许多CPU都具有指令：能返回在一个word中，第一个等于1的位的偏移量（如x86的BSF/BSR）、共有多少个为1的位（如popcnt）等

(1)位图法（Cont.）

- 位图法需要额外存储
  - Example:
    - 块大小 4KB，磁盘1TB
    - 则共有268,435,456块（1T/4K=256M）
    - 需要占用32MB空间（256M/8=32MB），即8k块（32M/4K=8K）
  - 如果以4块为1簇，则共有67,108,864簇（1T/16K=64M）
    - 需要占用8MB空间（64M/8=8MB），即512簇（8M/16K=512）
  - 当磁盘有1PB时候，占用空间为32G
- 位图法适合连续存储法

(2)链接法

- 无法很容易地分配连续空间
- 但不浪费空间

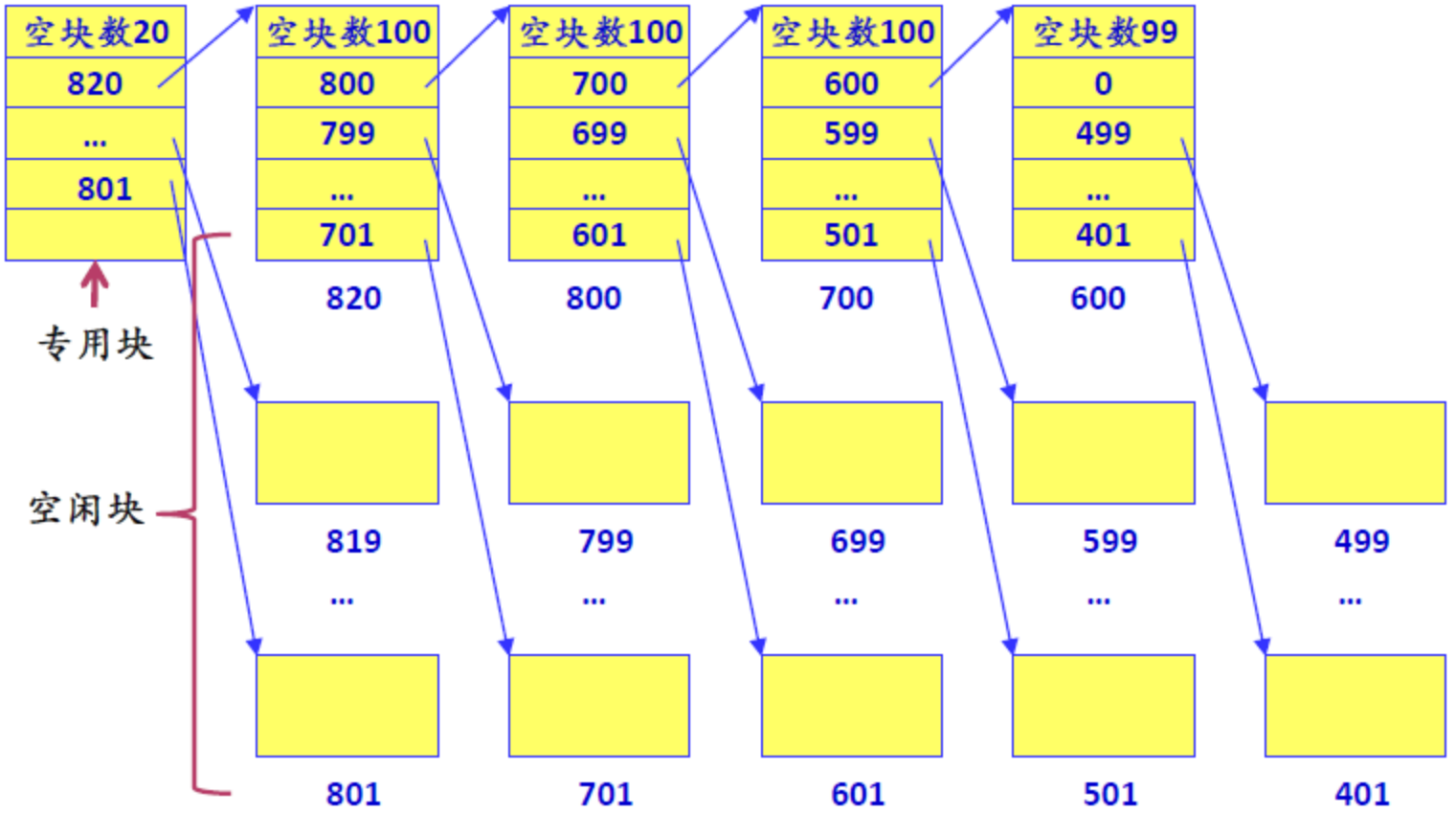


(3)分组法

- 分组法
  - 将n个空闲块的地址存储到第一个空闲块中，在最后一块，再添加一个指针指向另外n个空闲块的地址，如此类推
  - 有利于快速找到大量空闲块

- 分组法的一种变形：空闲块成组链接法
  - UNIX系统采用成组链接法对空闲盘块加以组织。
  - 空闲盘块的组织：将若干个空闲盘块划归一组，将每组中的所有盘块号存放在其前一组的第一个空闲盘块号指示的盘块中，而将第一组中的所有空闲盘块号放入专用块的空闲盘块号表中。
  - 在工作时，会把当前磁盘专用块复制到主存系统工作区

UNIX/Linux空闲块成组链接法

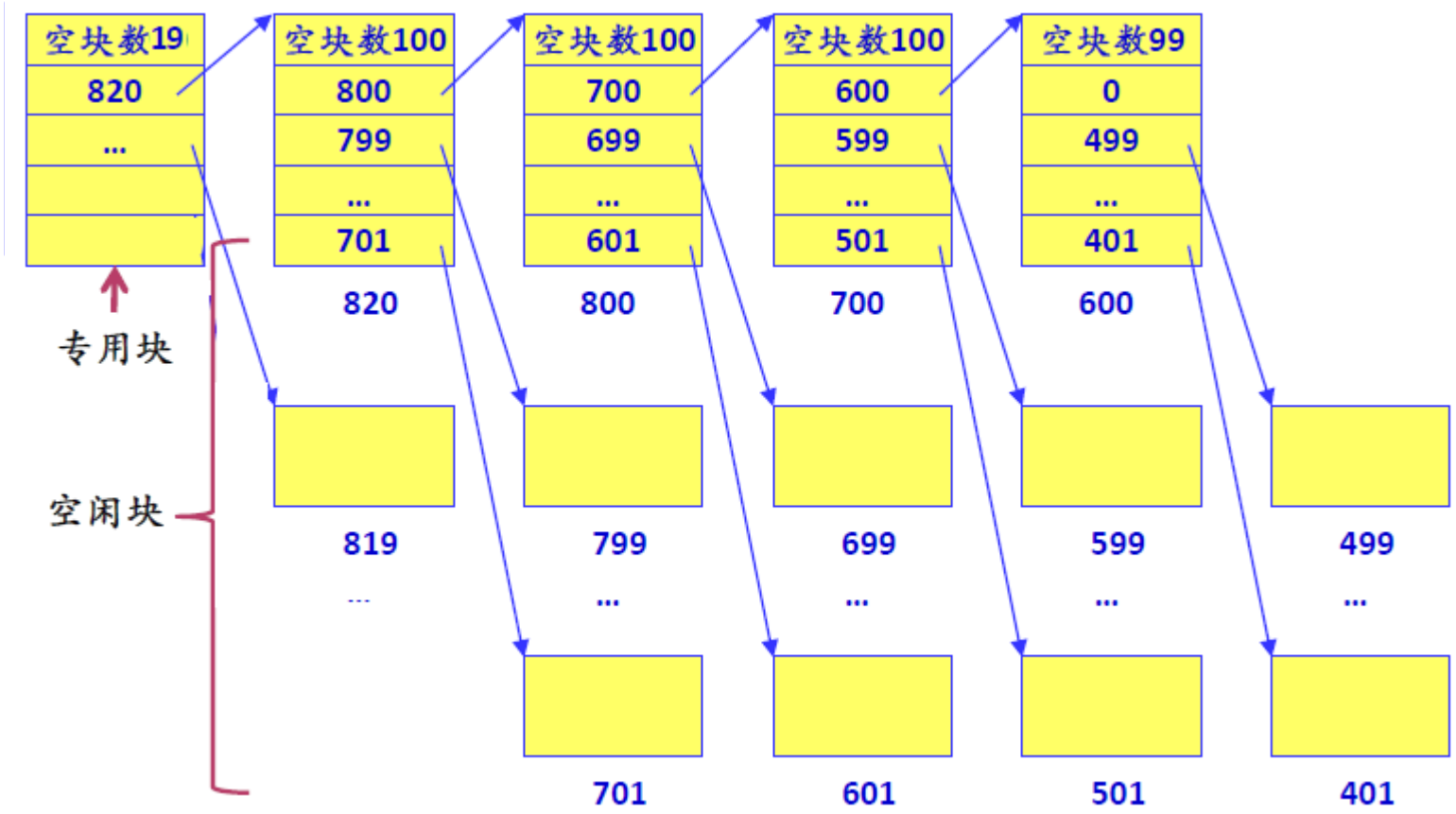


- 每100块划分一组，每组第一块登记下一组空闲块的盘物理块号和空闲总数。

UNIX/Linux空闲块成组链接法

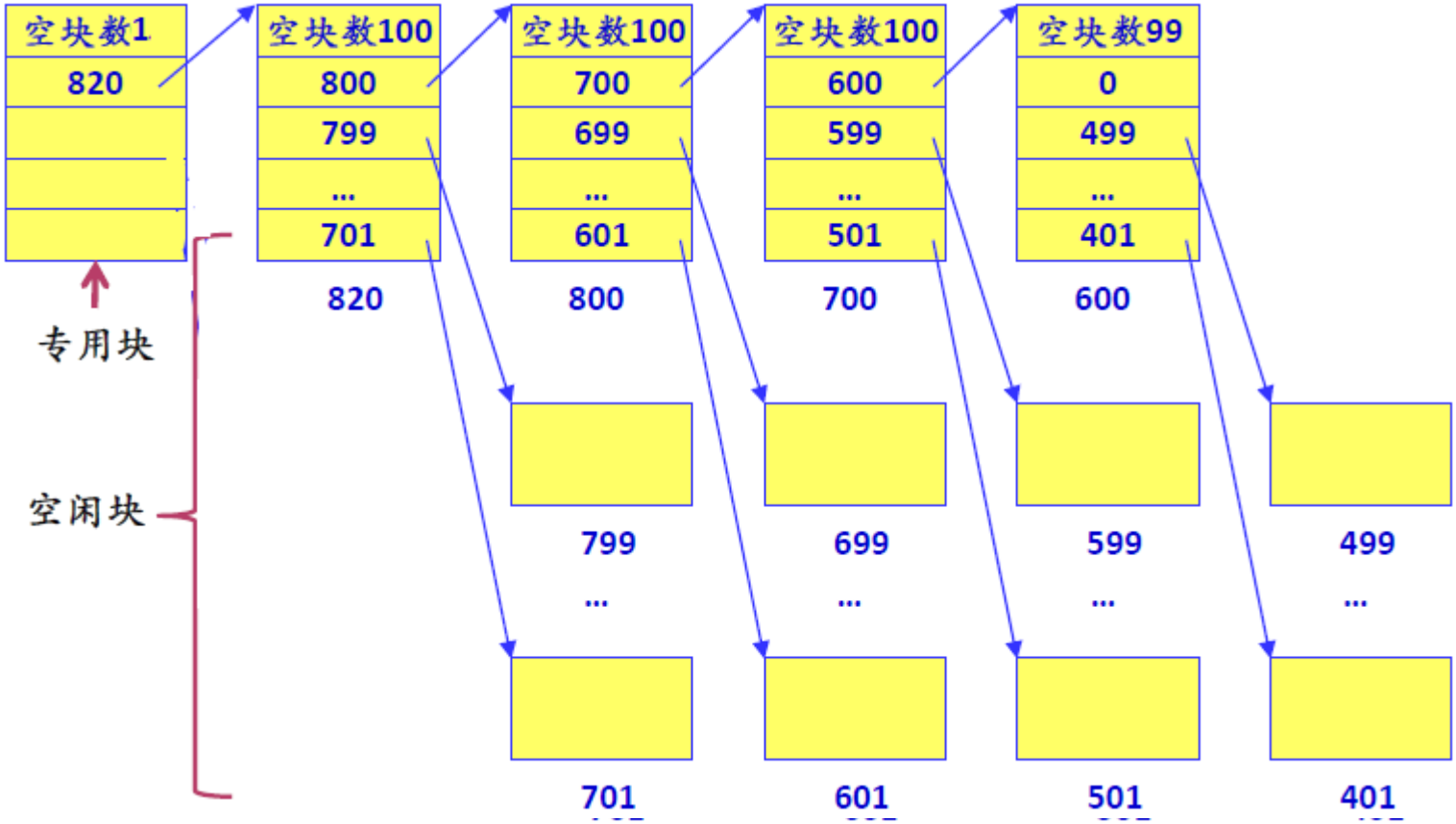
- 空闲盘块的分配
  - 当要分配一个盘块时，首先将专用块空闲盘块号表中下一个可用盘块分配出去；
  - 如果所分配盘块号是专用块中最后一个可用盘块号（记录了下一个空闲组的信息），则先将该盘块中的内容读入内存中专用块空闲盘块号表中，然后才将该盘块分配出去。

UNIX/Linux空闲块成组链接法



- 分配801块后

UNIX/Linux空闲块成组链接法



- 继续分配，...，直到分配819后

UNIX/Linux空闲块成组链接法

- 分配820后（820块是专用块中登记的最后一块）

UNIX/Linux空闲块成组链接法

- 空闲盘块的回收
  - 在回收空闲盘块时，如果专用块中的空闲盘块号表未满，可直接将回收盘块的编号放入空闲盘块号表中；
  - 若空闲盘块号表已满，需先将空闲盘块号表中的所有盘块号复制到新回收的盘块中，再将新回收盘块的编号放到专用块空闲盘块号表第一个位置，此块号就成了表中惟一的盘块号，空闲块计数重置为1，然后继续。

UNIX/Linux空闲块成组链接法

- 回收新的一块，假设块号为899。
- 如果当专用块中已经满了，再回收新的一块，怎么样？

(4)计数法

- 计数法
  - 针对频繁、连续的分配与释放
    - 不记录n个空闲块的地址，而是只记录第一个空闲块的地址，以及连续的空闲块个数
    - 有利于减少表的总长度

#	First free block	Count	Block Number
1	5	3	（5，6，7）
2	13	5	（13，14，15，16，17）
3	20	6	（20，21，22，23，24，25）
4	--	--	--

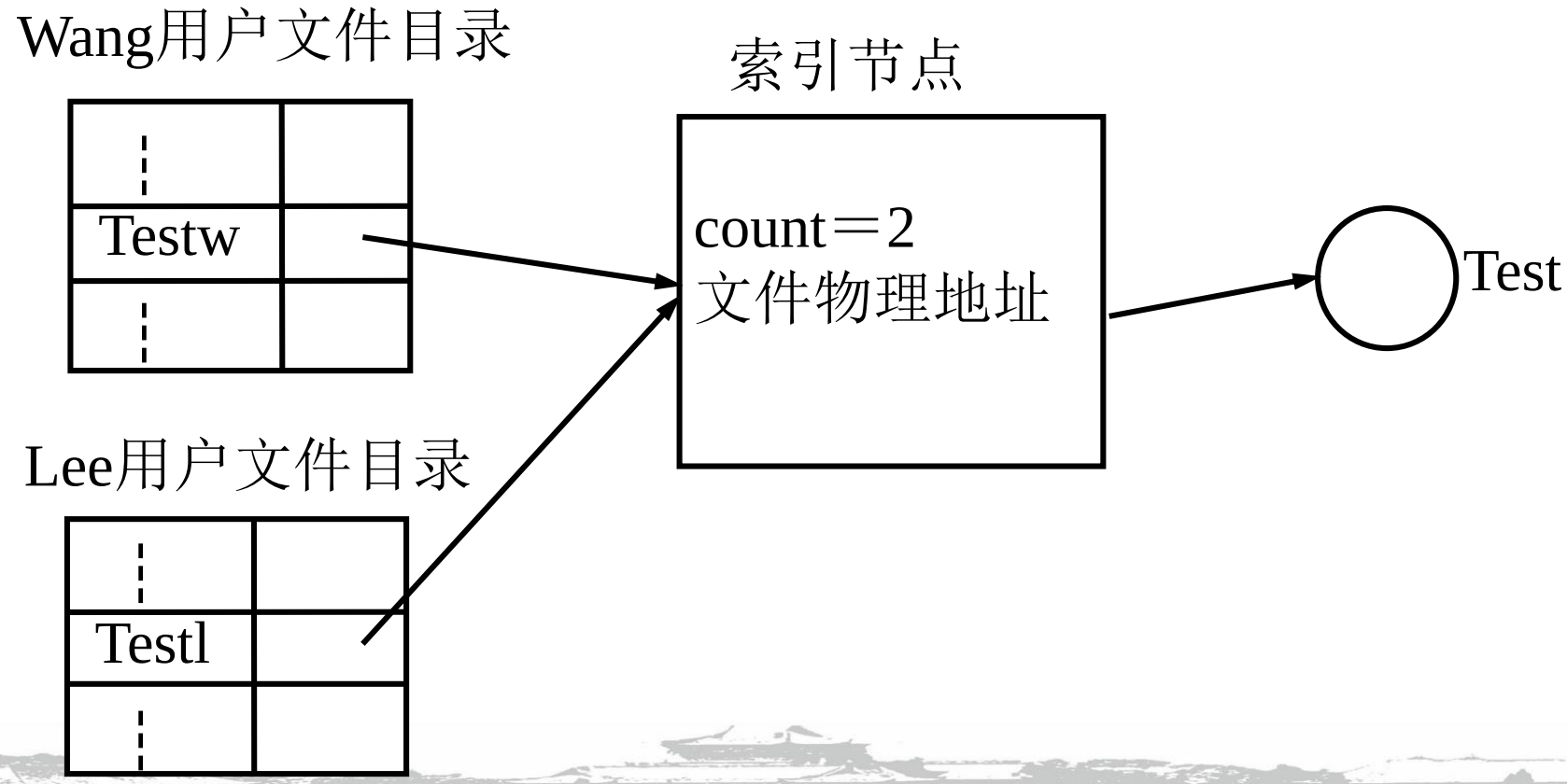
目录

文件共享

- 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。

(1)基于索引节点的共享

- 索引节点：文件属性信息构成的数据结构，又称inode
- 文件名和文件属性分离
- 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。

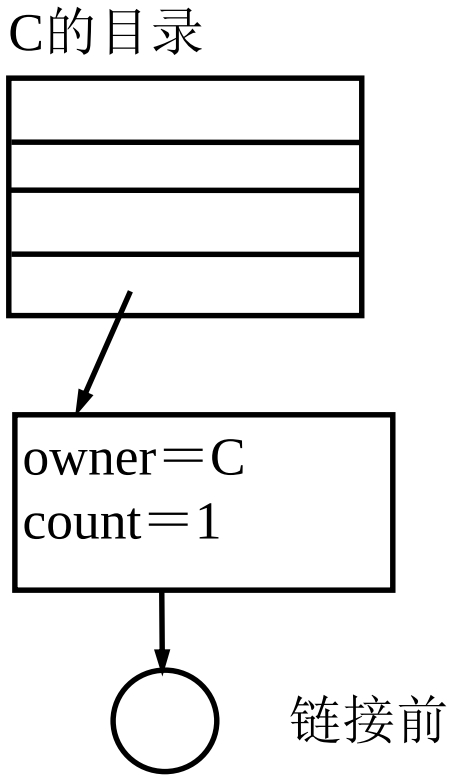


索引节点中的链接计数

- 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。

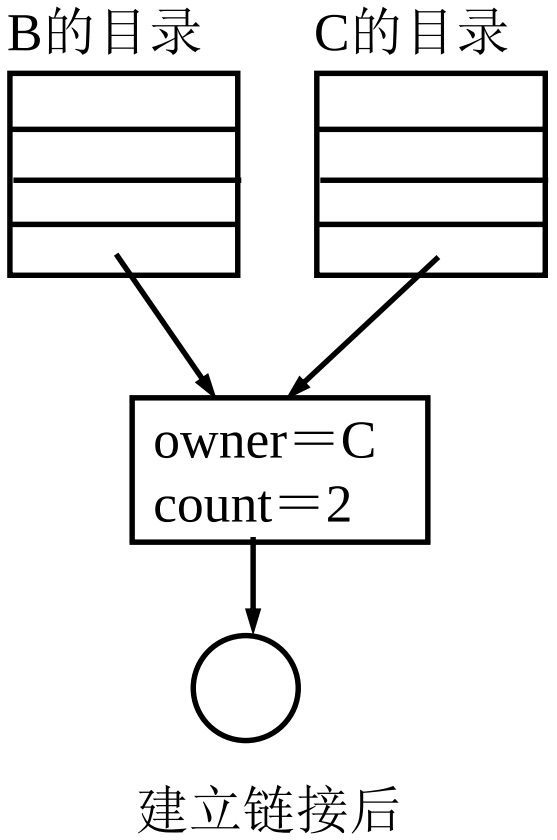
链接例--C创建一个新文件

- 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

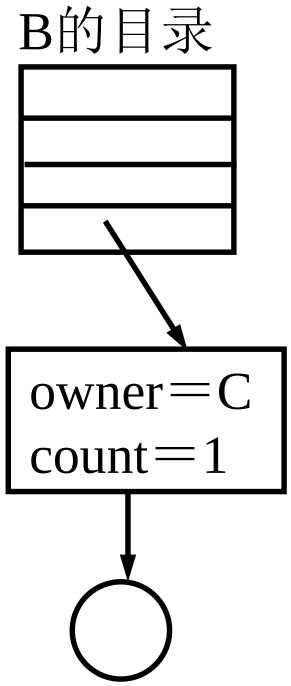
- 当用户B希望共享此文件时，应在用户B的目录中增加一个目录项，并设置指针指向该文件的索引节点，此时文件的所有者仍然是C，但索引节点的链接计数应加1（count＝2）。



建立链接后

链接例-- C删除文件

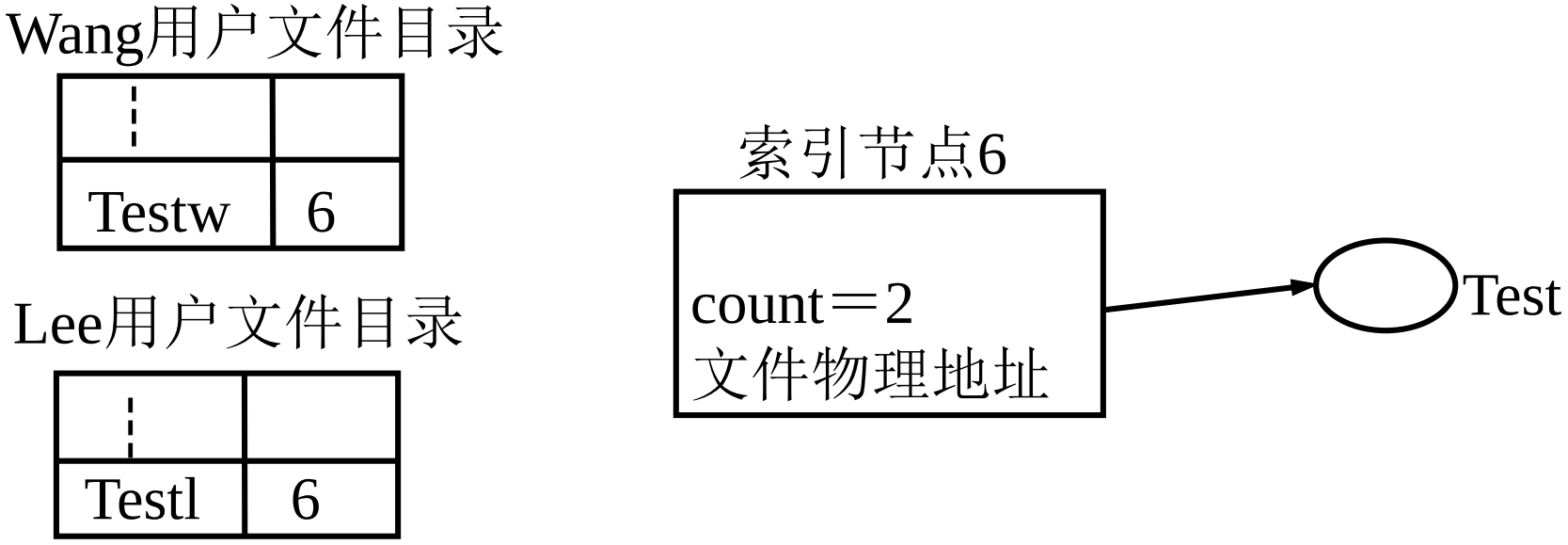
- 如果以后用户C不再需要该文件，则系统只删除C的目录项，并将count减1。
- 此时只有B拥有指向该文件的目录项，而该文件的所有者仍然是C。如果系统进行记账，C将继续为该文件付账。
- 当B不再需要它，count为0，该文作被删除。



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。

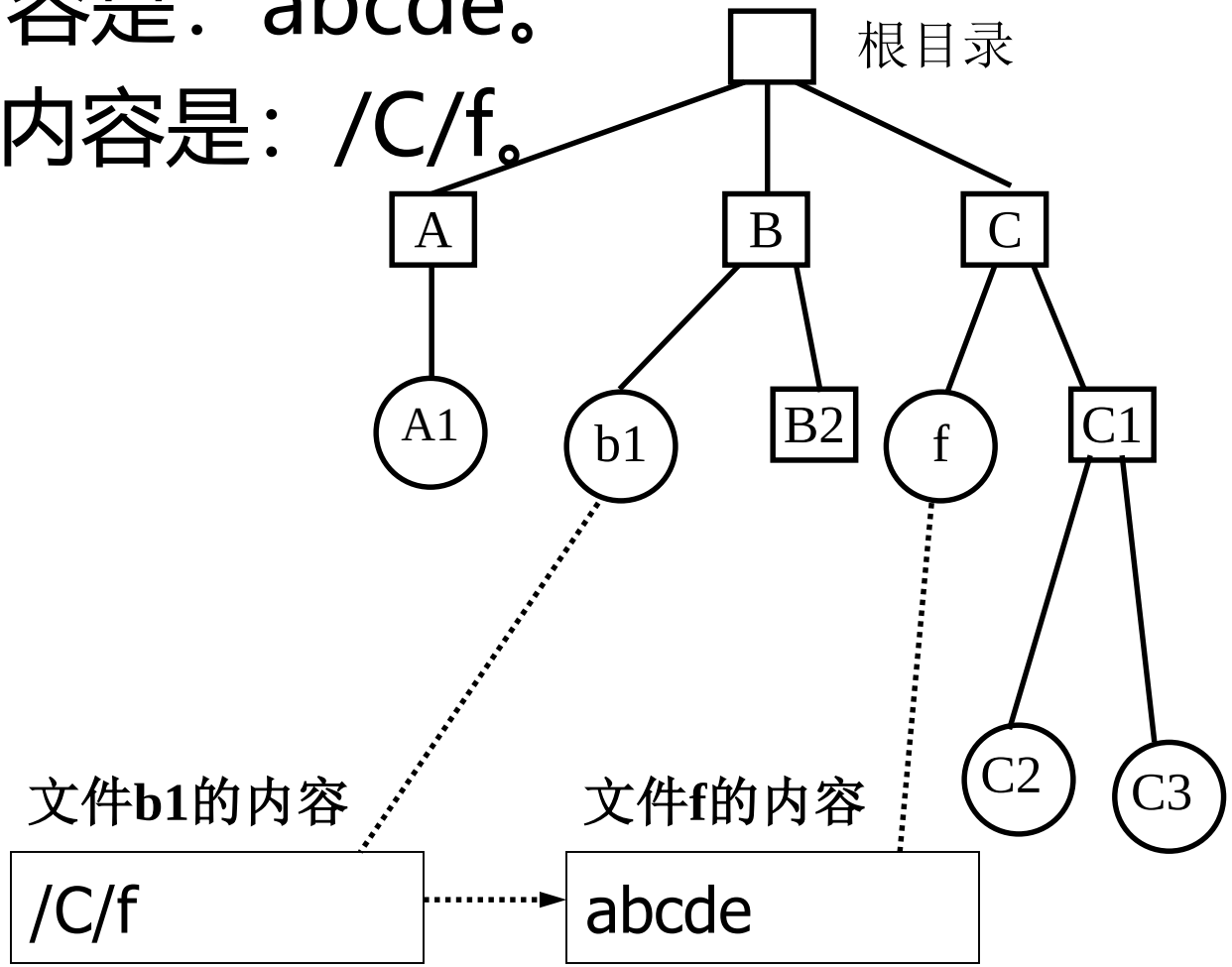


(2)利用符号链接实现文件共享

- 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。

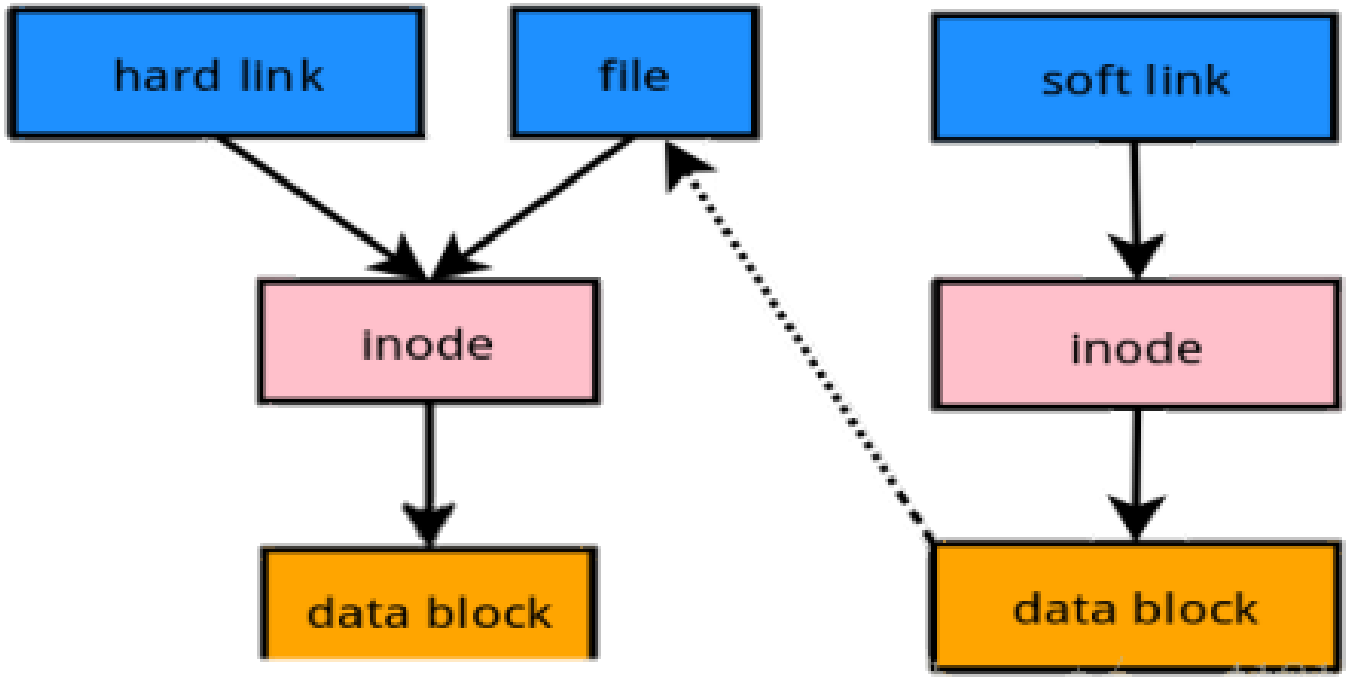
符号链接的特点

- 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。

总结：硬链接与软链接

- 硬链接是通过索引节点来链接，其结果是多个文件名指向同一索引节点，即一个文件拥有多个有效路径名
- 软链接类似于快捷方式，实际上只包含要链接的文件路径和文件名

- 软硬链接对引用数的影响
  - 创建硬链接时，引用数+1
  - 创建软连接时，由于只拷贝文件路径和文件名，引用数不变


hard link
file
soft link
inode
inode
data block
data block

总结：硬链接与软链接（cont.）

- 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
  - 删除原文件，对硬链接无影响，符号链接失效
- 文件真正删除的条件是与之相关的硬链接文件均删除

课堂练习（考研真题）

- 设文件F1的当前引用计数值为1，先建立F1的符号链接（软链接）文件F2，再建立F1的硬链接文件F3，然后删除F1。此时，F2和F3的引用计数值分别是
- A．0、1　B.1、1　C.1、2　D.2、1

作业

- 1、在实现文件系统时，为加快文件目录的检测速度，可利用“文件控制块分解法”。假设目录文件存放在磁盘上，每个盘块1024字节。文件控制块占64字节，其中文件名占8字节。采用分解法后，通常将文件控制块分解成两部分分别存储，第一部分占16字节（包括文件名和文件内部号），第二部分占64字节（包括文件内部号和文件其他描述信息）。
  - （1）假定某一目录文件共有254个文件控制块，试分别给出采用分解法前和分解法后，查找该目录的某一个文件控制块的平均访问磁盘次数。
  - （2）一般地，若目录文件分解前占用n个盘块，分解后改用m个盘块存放文件名和文件内部块号部分，请给出访问磁盘次数减少的条件。

作业

- 2、设某文件系统采用索引文件结构
  - 假定文件目录项中有10个表目用于描述文件的物理结构（每个表目占2B）
  - 磁盘块的大小和文件逻辑块大小相等，都是512B。
  - 经统计发现，此系统处理的文件具有如下特点，60%的文件其大小 ≤ 8个逻辑块，30%的文件其大小≤ 260个逻辑块，10%的文件其大小≤ 65000个逻辑块。
  - 设计此文件系统的索引结构，使得系统能够处理各类文件，并使读磁盘的次数理论上减少。