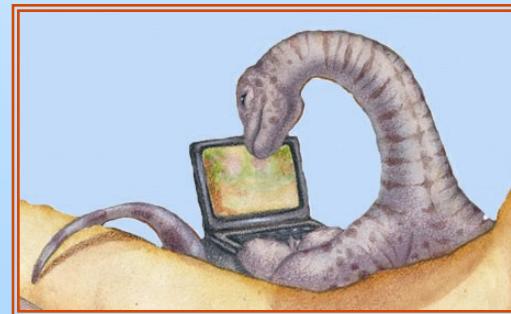


第11章 文件系统实现



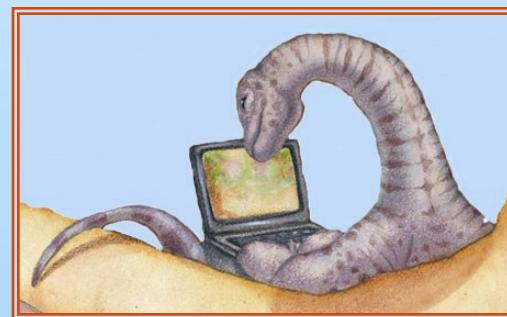


内容

1. 文件系统
2. 连续分配
3. 链接分配
4. 索引分配
5. 空闲空间管理



1、文件系统



文件系统结构
文件系统实现
虚拟文件系统
网络文件系统
常用文件系统



文件系统

- 在存储设备上组织文件的方法和数据结构
- 操作系统中负责管理和存储文件信息的模块
- 系统角度
 - ① 对存储设备的空间进行组织和分配
 - ② 负责文件检索、读写等操作
 - ③ 目标：存取速度和存储空间效率
- 用户角度
 - ① 提供按名存取的文件访问机制
 - ② 文件的组织管理，如目录等
 - ③ 目标：方便的文件存取机制





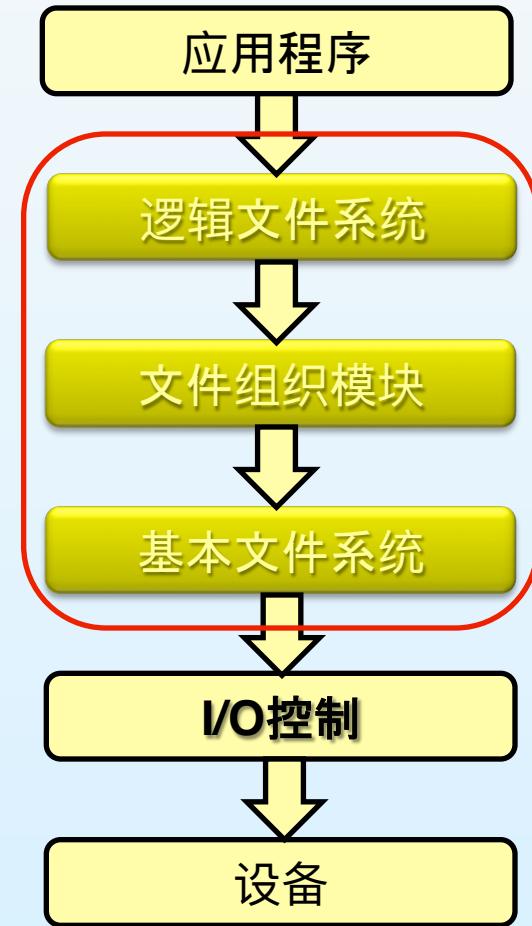
文件系统的层次架构

■ I/O控制

- 设备驱动程序（Device Drivers）
- 中断

■ 设备驱动程序

- 控制I/O设备运行
- 向硬件控制器发送专门控制命令
- 操作系统通过设备驱动程序控制设备进行文件的读写操作

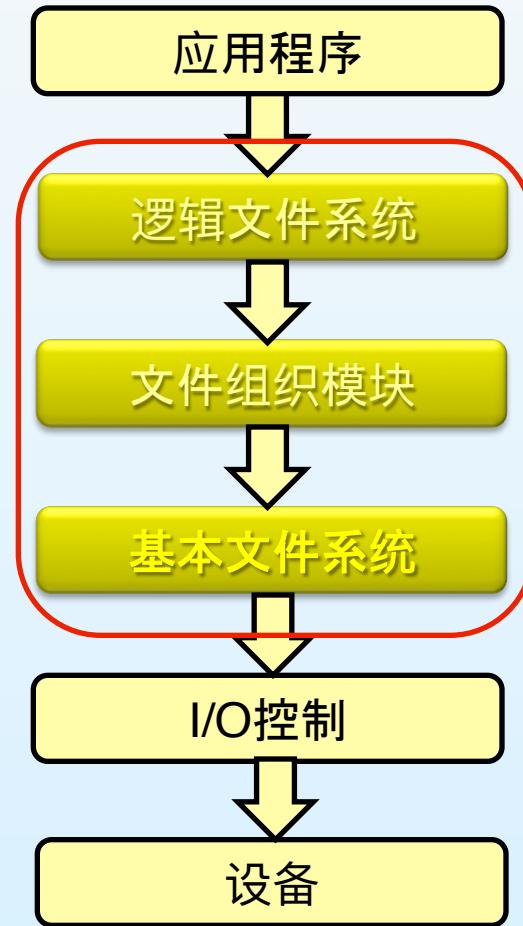




文件系统的层次架构

■ 基本文件系统

- 负责物理块读写
- 向设备驱动程序发送控制命令控制设备控制器对存储设备进行读写操作
 - ▶ 例如： read drive 1, cylinder 72, track 2, sector 10, into memory location 1060

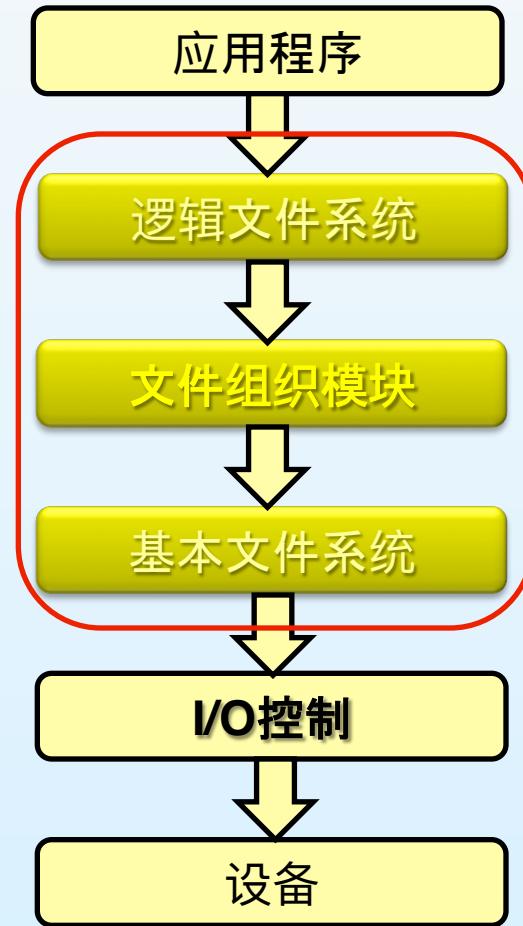




文件系统的层次架构

■ 文件组织模块

- 管理文件、逻辑块和物理块
- 把文件的逻辑地址转换为物理地址
- 管理空闲空间
- 为文件分配物理块

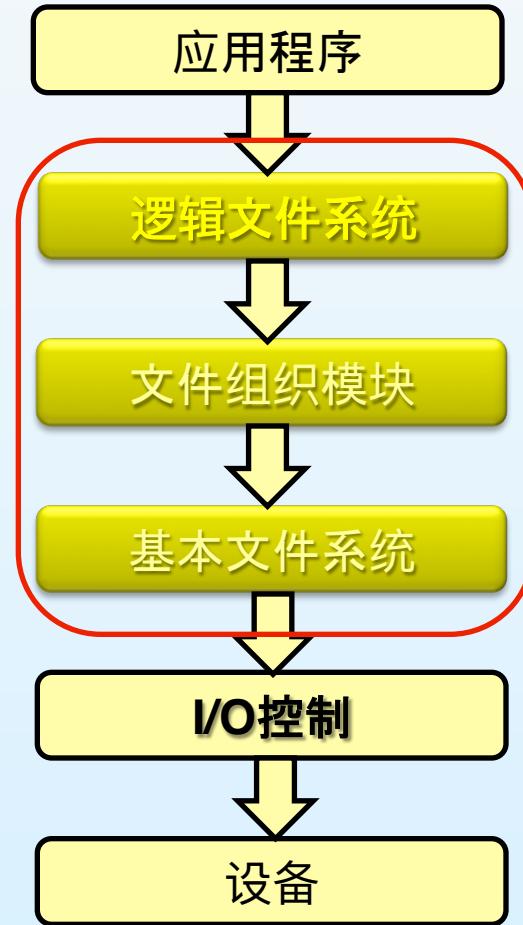




文件系统的层次架构

■ 逻辑文件系统

- 管理文件系统中的元数据
 - ▶ 除了文件数据外的所有结构数据
- 文件按名存取
- 文件目录组织管理
- 把文件名转换为文件ID，文件句柄
- 管理FCB
- 存储保护





■ 文件系统中不包括的模块是()。

- A.逻辑文件系统
- B.I/O控制模块
- C.文件组织模块
- D.基本文件系统





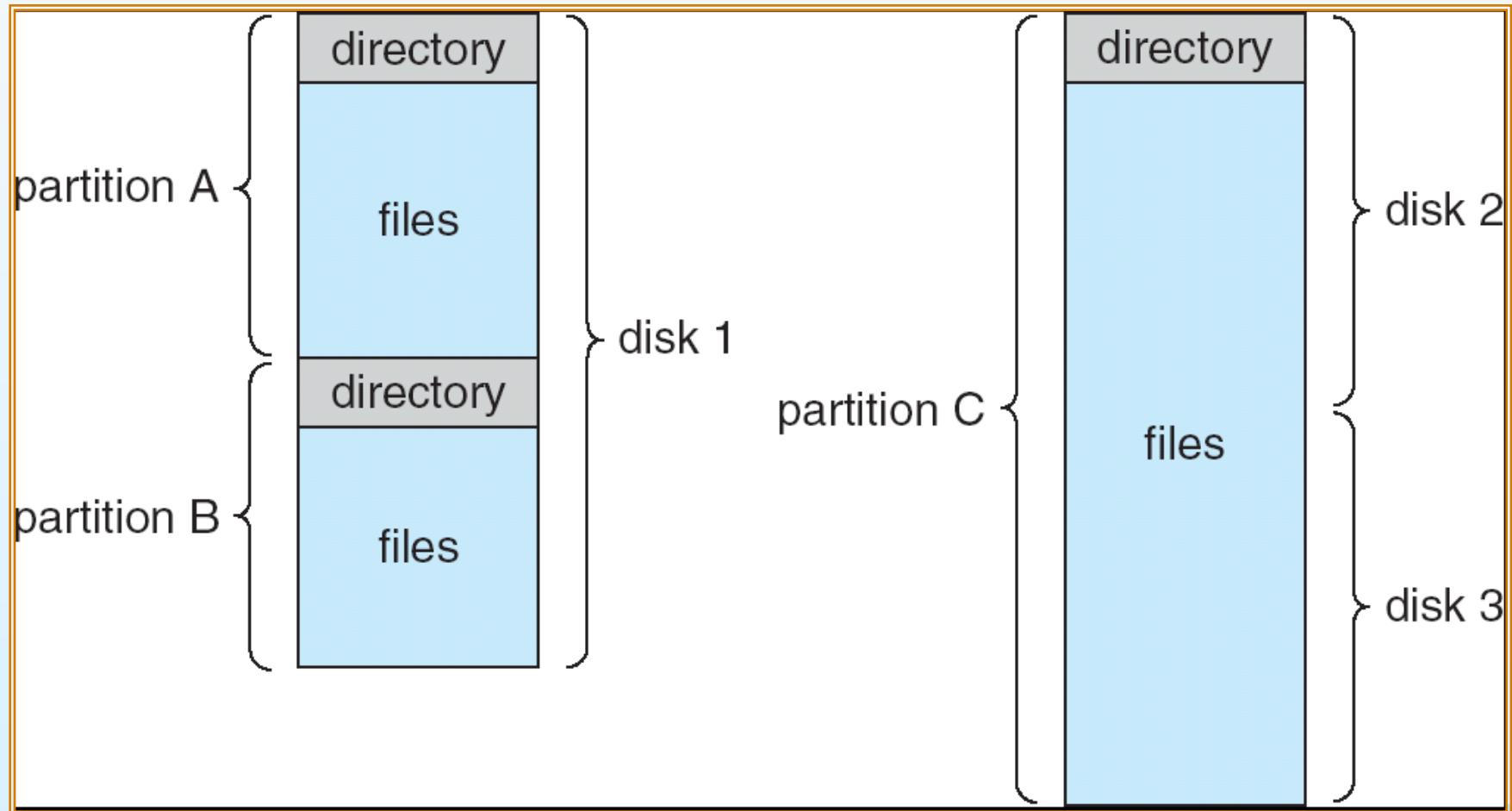
文件系统实现-基本概念

- 物理块（簇）：一个或多个（ 2^n ）扇区组成，基本文件读写单位
- （物理）分区（Partition）：磁盘分割成若干个独立的空间，每个空间称为分区
 - 两大类分区：主分区和扩展分区
 - 主分区：能够安装操作系统的启动分区
 - 扩展分区：不能直接使用，必须分成若干逻辑分区
- 卷（逻辑磁盘）（Volume）：磁盘上的逻辑分区，建立在物理分区上
 - 一般每个卷可以建立一个文件系统

磁盘 0 基本 1862.89 GB 联机	data (E:) 886.33 GB NTFS 状态良好 (主分区) soft (D:) 976.56 GB NTFS 状态良好 (页面文件, 主分区)
磁盘 1 基本 111.67 GB 联机	(C:) 111.57 GB NTFS 状态良好 (启动, 故障转储, 主分区)



典型文件系统





文件系统实现

■ 两种文件系统

- 磁盘文件系统
- 内存文件系统

■ 磁盘文件系统结构

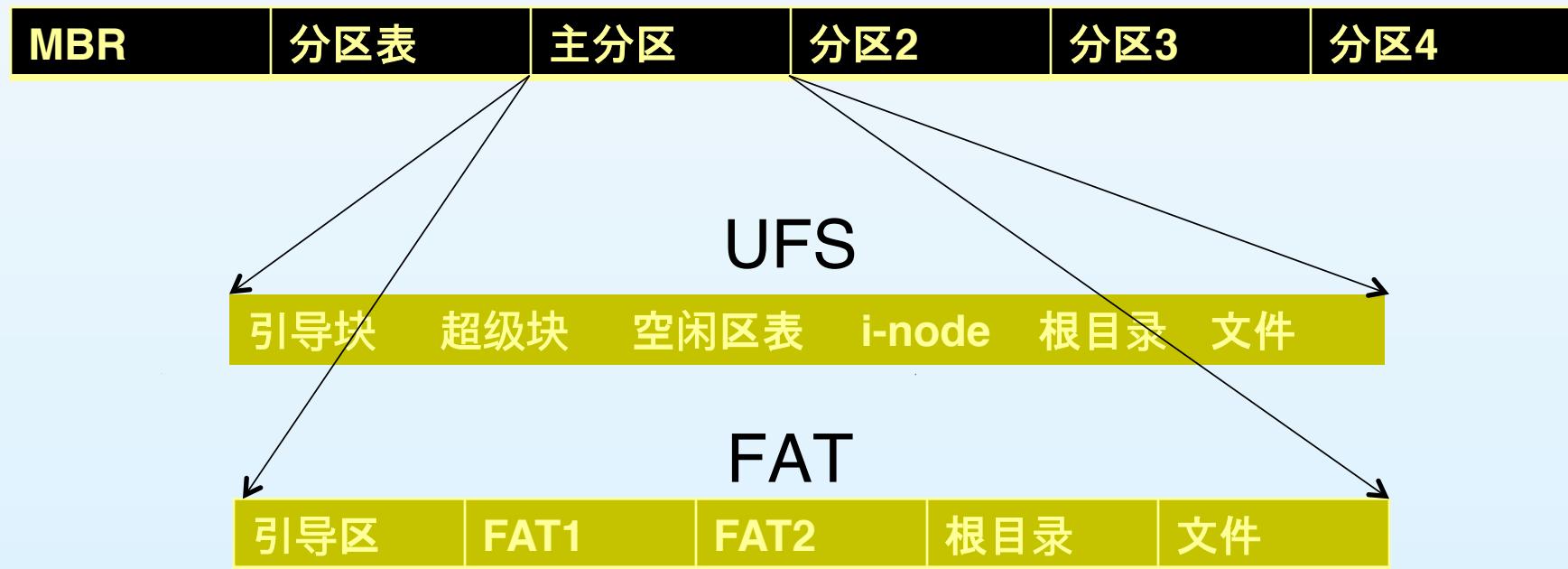
- 引导控制块（Boot control block）：包含了系统引导操作系统的各种信息，只有安装操作系统的分区才有引导控制块
 - UFS：引导块（Boot block）
 - NTFS：分区引导扇区（Partition boot sector）
- 分区控制块（Partition control block）：包含分区信息
 - 总的块数、空闲块数、块大小等信息
 - UFS：超级块（superblock）
 - NTFS：主控文件表（master file table）
- 目录和FCB
- 用户文件





磁盘文件系统

磁盘结构



MBR主引导记录：存储在磁盘的0柱面、0磁头、1扇区，存储主引导程序等信息，在计算机启动时运行

硬盘分区表DPT存储了硬盘的分区信息，这个例子中有4个分区





内存文件系统

■ 包括：

- 分区表：所有安装分区信息
- 目录缓冲结构：保存最近访问的目录信息
- 系统打开文件表
- 进程打开文件表

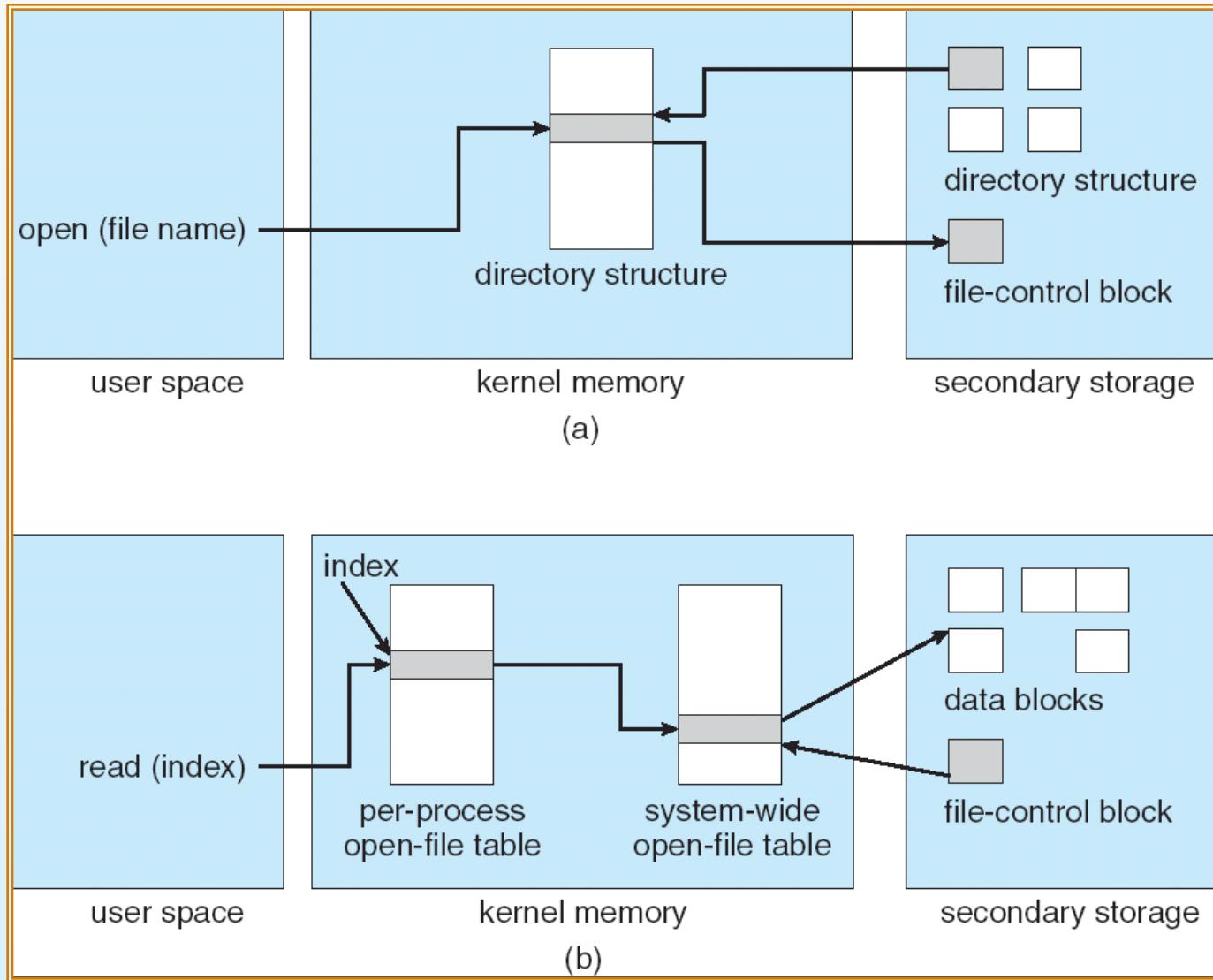
■ 文件操作需要用到内存文件系统

■ 目的：通过缓冲技术提高文件系统性能





内存中文件系统结构





虚拟文件系统(Virtual File System)

■ 目的:

- 为了支持多个文件系统，引入虚拟文件系统VFS
- 把多个文件系统整合成一个目录结构
- 为用户屏蔽各个文件系统的差异

■ 虚拟文件系统(VFS):

- 提供了一种面向对象的方法来实现文件系统
- 为不同类型的文件系统提供了接入VFS的接口
- 为用户提供了统一的系统调用接口(API)





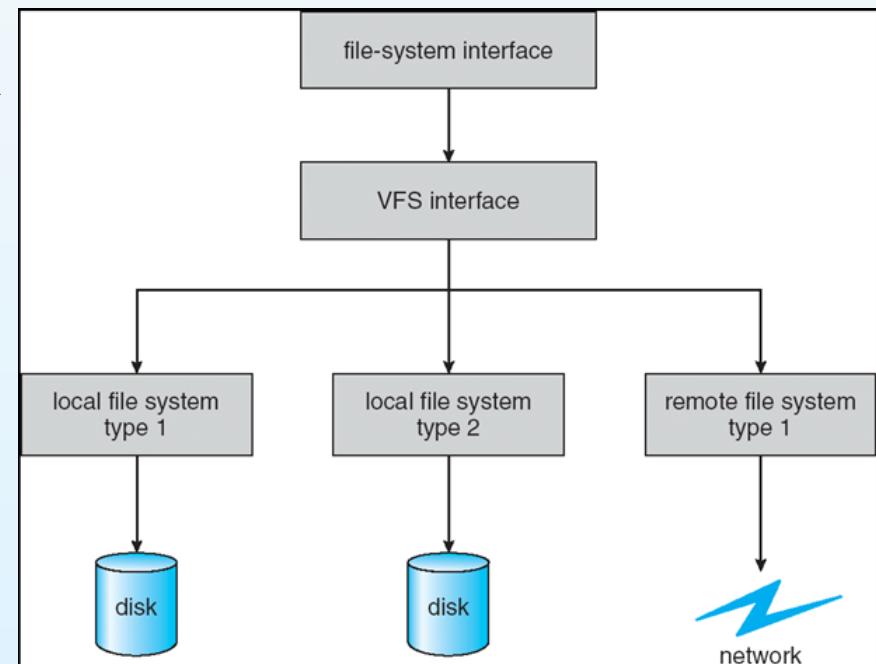
虚拟文件系统

■ 文件系统接口（File system interface）

- 统一的应用程序访问文件的接口
- 各个文件系统提供给应用程序的接口可能不同，如：open，openfile、openf、open_file等

■ VFS接口（VFS interface）

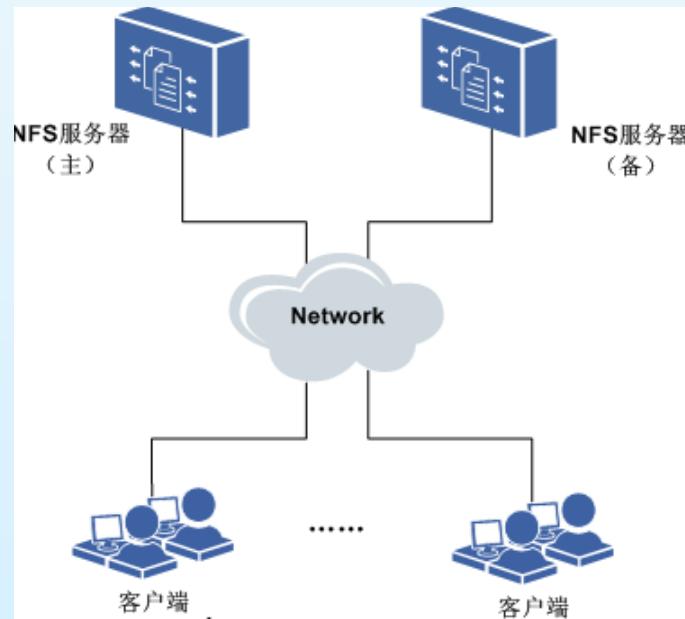
- 为各类不同的文件系统定义VFS接口
- 符合该接口的文件系统都可以接入VFS





网络文件系统:NFS

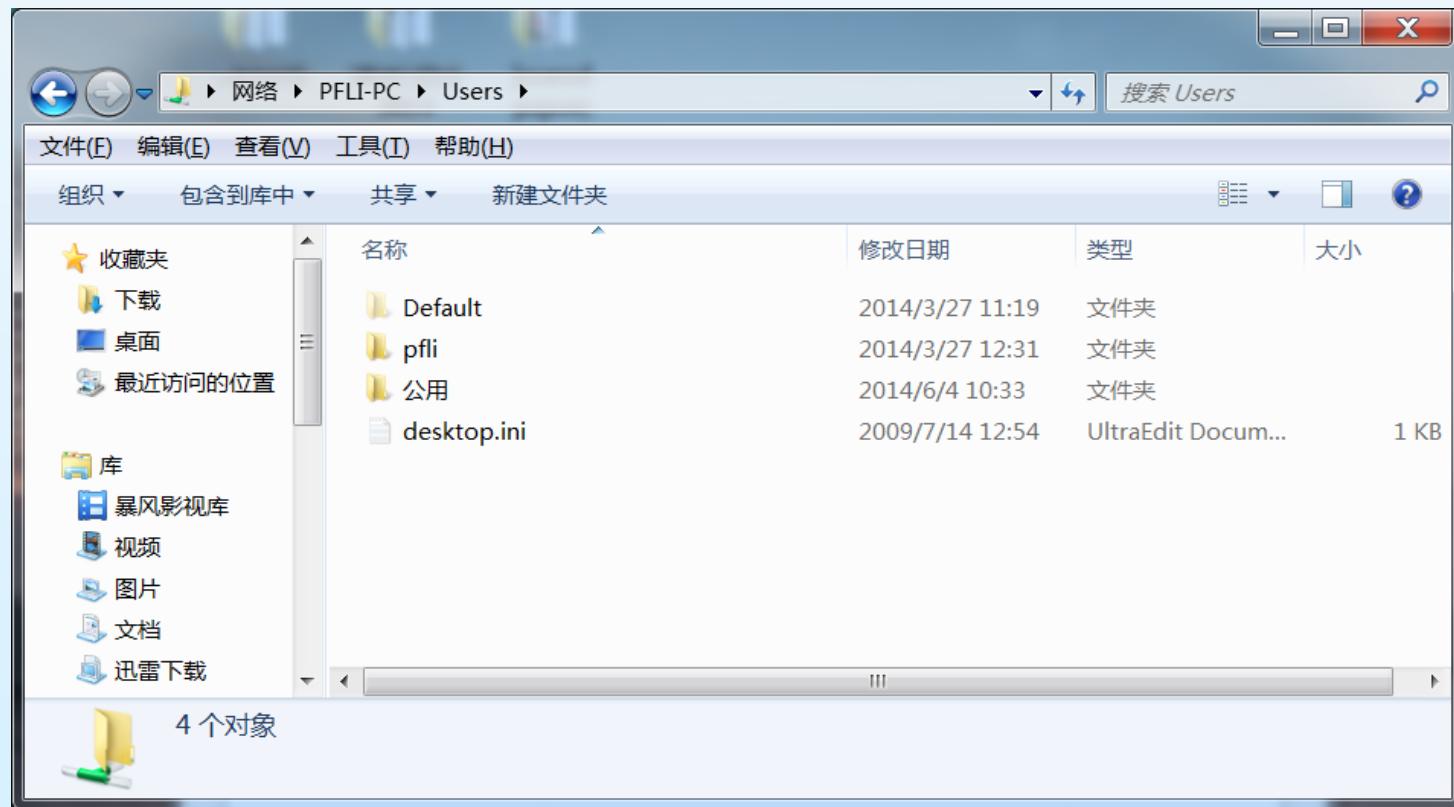
- 网络文件系统NFS(Network File System)
- 用于通过LAN(或WAN)访问远程文件系统的软件系统的实现或规范
- 好处:节省存储空间, 实现共享





CIFS

- 通用Internet文件系统(Common Internet File System)
- 在Windows主机之间进行网络文件共享
- CIFS 使用客户/服务器模式





常用文件系统

■ Windows

- FAT (File Allocation Table)
- NTFS (New Technology File System)
- ReFS (Resilient File System)

■ Linux

- Ext (Ext2、Ext3、Ext4)

■ Mac OS

- HFS(Hierarchical File System)

■ CD

- CDFS

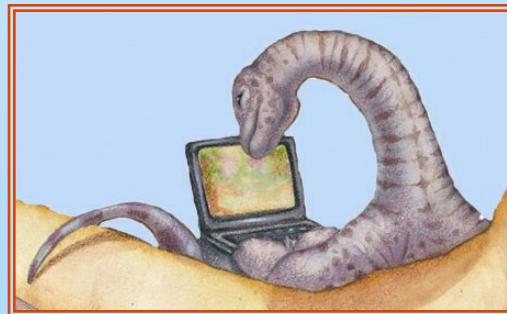




- 引入内存文件系统的目的是为了节省外存空间。
- 虚拟文件系统可以把多个文件系统整合成一个目录结构，为用户屏蔽各个文件系统的差异。



连续分配





内 容

- 物理块
- 存储空间分配方式
- 连续分配
- 性能分析
- 连续分配的改进



物理块

■ 读写存储设备的基本单位

- 文件读写操作时，以块为单位进行读写
- 如：程序需要读1个字节，则OS把包含该字节的一块读入
- 好处：减少读写次数，提高访问效率

■ 存储设备的基本分配单位

- 以物理块为单位为文件分配存储空间

■ 和内存的页面大小相对应

- 页面大小：4KB
- 物理块大小：4KB的倍数
 - ▶ 如，4K/8K/16K/32K/64K

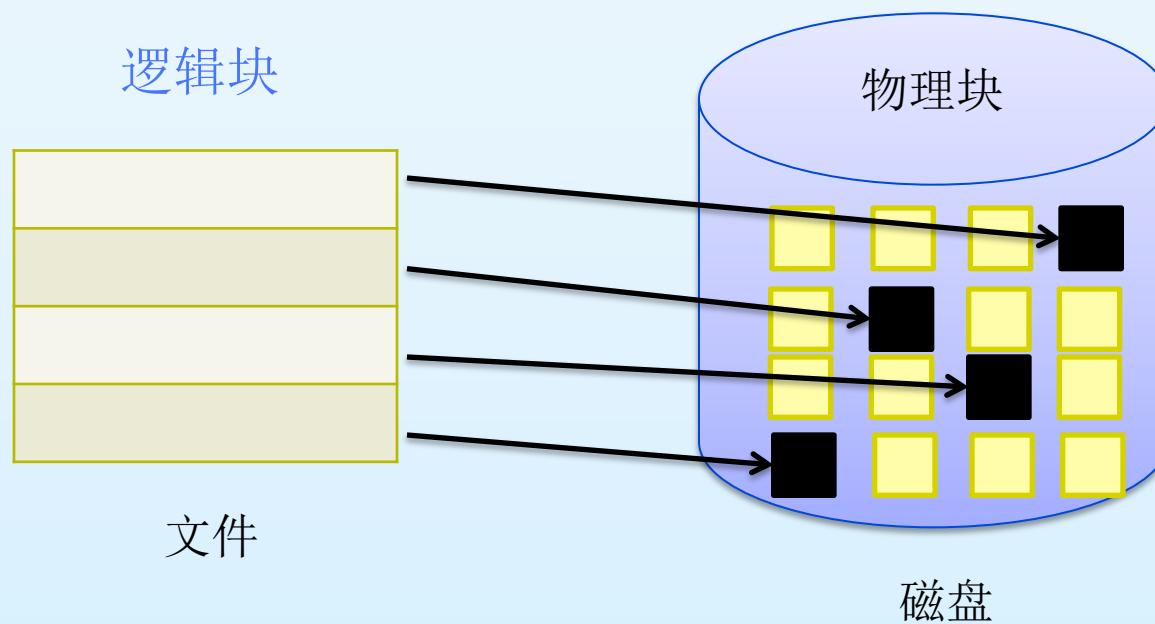




逻辑块

■ 逻辑块：在文件空间中的块

- 大小和物理块一致
- 一个逻辑块存储在一个物理块中





存储空间分配方式

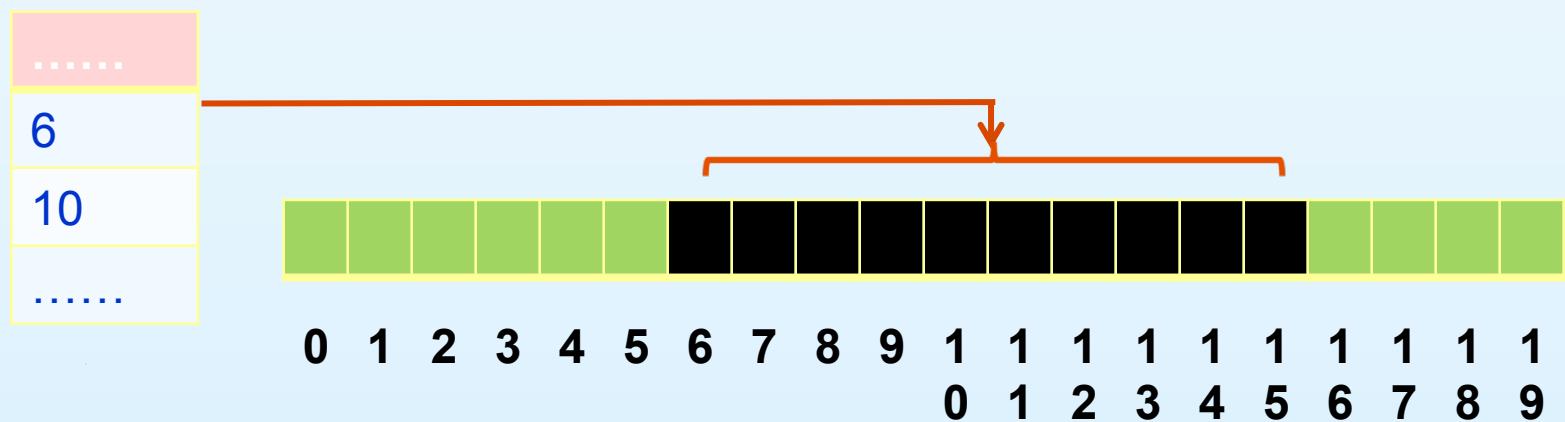
- 连续分配 → 连续存储空间
- 链接分配]
- 索引分配]
- 连续存储空间分配，和内存的连续分配相似，是指一个文件在磁盘上存储在连续的物理块中；
- 离散存储空间分配，是指一个文件的物理块可以分布在磁盘的各处，类似于内存分配中的分页和分段
- 物理块块号
 - 一维空间
 - 从0开始编号
 - 可以根据物理设备的特性进行转换



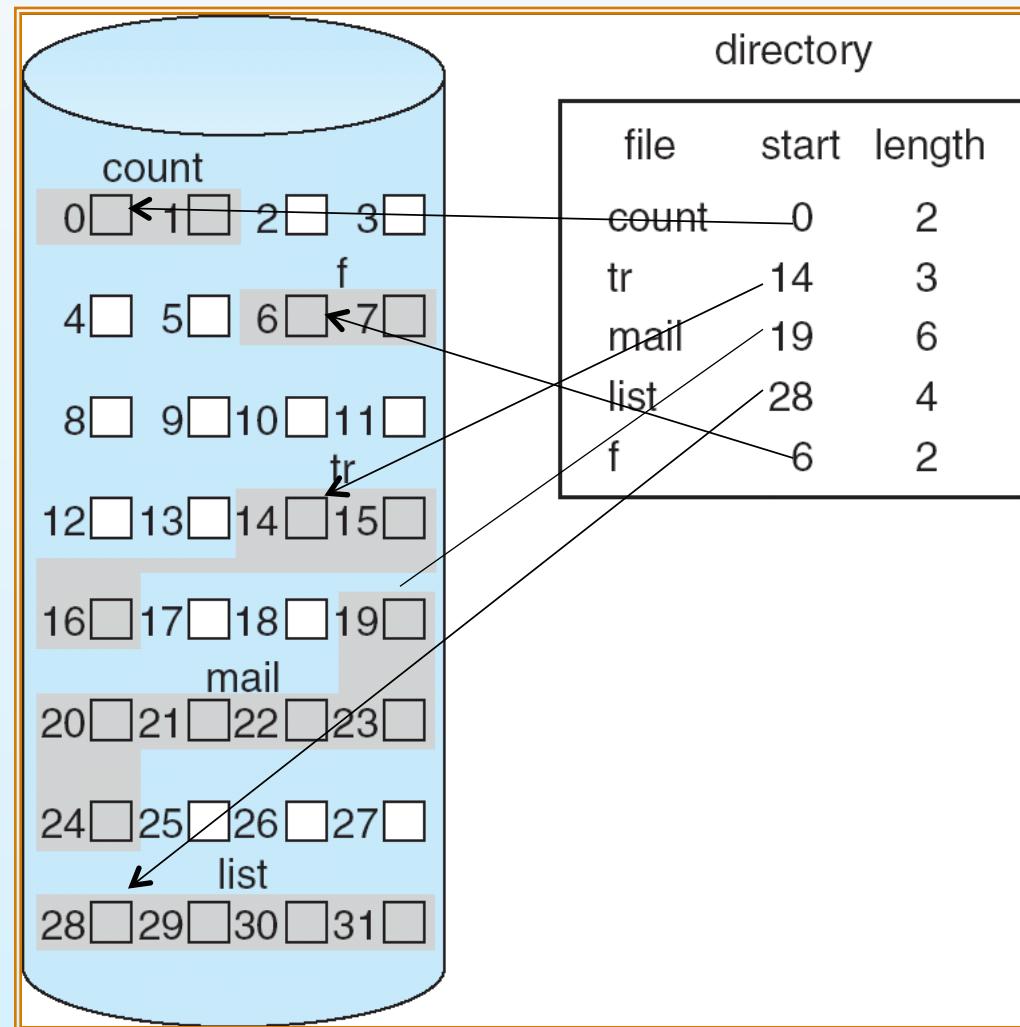


连续分配

- 每个文件在磁盘上占用一组连续的物理块
- FCB仅需给出：
 - 起始块号
 - 长度



连续分配例子





地址映射

- 逻辑地址LA: 文件内相对地址（一维）
- 物理地址(B,D): 存在在物理块中的地址（二维）

块号B 块内偏移D

- 物理块大小: S



◆ 物理地址:

▶ 访问块号 $B = Q + \text{起始块号}$

● 块内偏移D

商 \swarrow Q 逻辑块号
 LA/S
余数 \searrow D 块内偏移





性能分析

■ 优点

- 支持随机访问（可直接访问指定块号的物理块）
- 存取速度快（上一个块到下一个块移动距离短）
- 适用一次性写入操作

■ 例子：文件中偏移为12321位置的数据（块大小4KB）

- $Q=[12321/4KB]=3 \quad D=33$
- 读入块号 $B=Q+起始块=3+6=9$



0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1

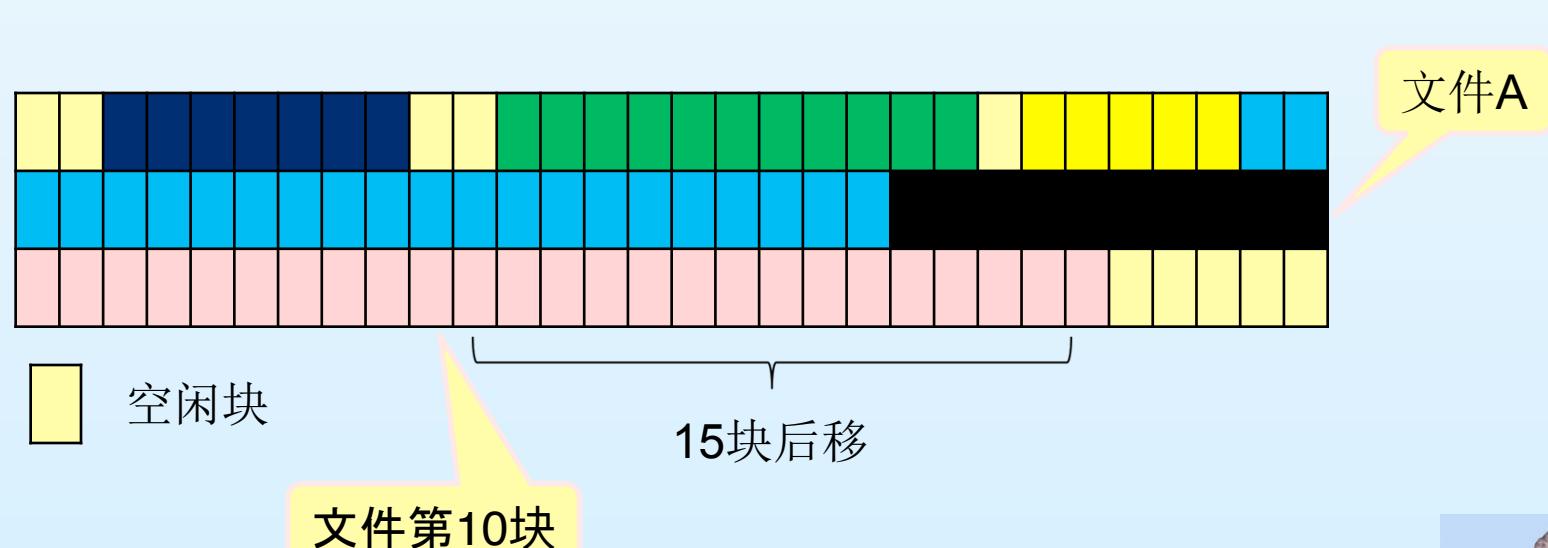




性能分析

■ 缺点

- 浪费空间（小空间无法分配）
- 文件不能动态增长（文件A）
- 不利于文件的插入和删除（需要移动数据）





连续分配的改进

- 改进的连续分配方案(Veritas File System)
- 基于扩展的文件系统（局部连续）
 - 扩展是一组连续的磁盘块集合
 - 扩展在文件分配时被分配
 - 一个文件可能包含一个或多个扩展
 - 需要一个指向下一个扩展的指针



链接分配





内 容

- 链接分配
- 隐式链接
- 显示链接
- 例子：FAT





链接分配

■ 离散物理块分配方式

- 链接分配
- 索引分配

■ 链接分配

- 文件信息存放在若干个不连续物理块中
- 文件的所有物理块通过指针链接成链表结构

■ 分类

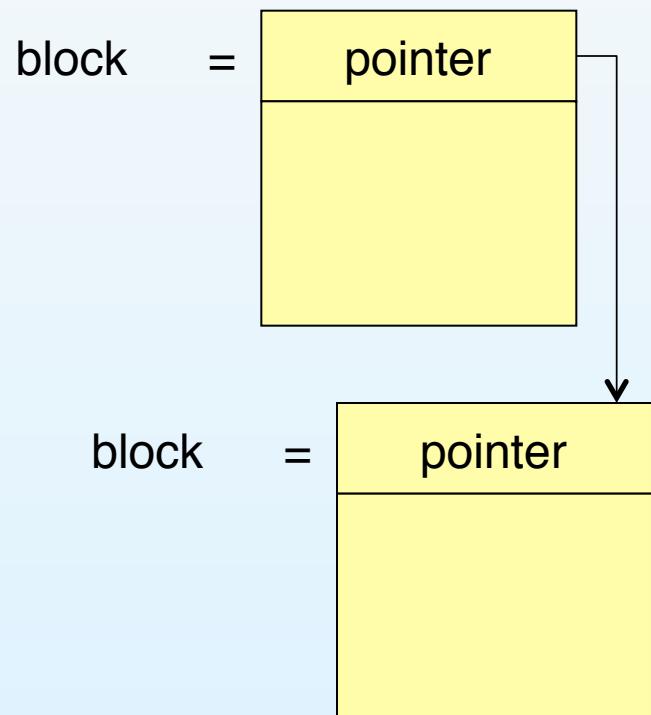
- 显示链接
- 隐式链接





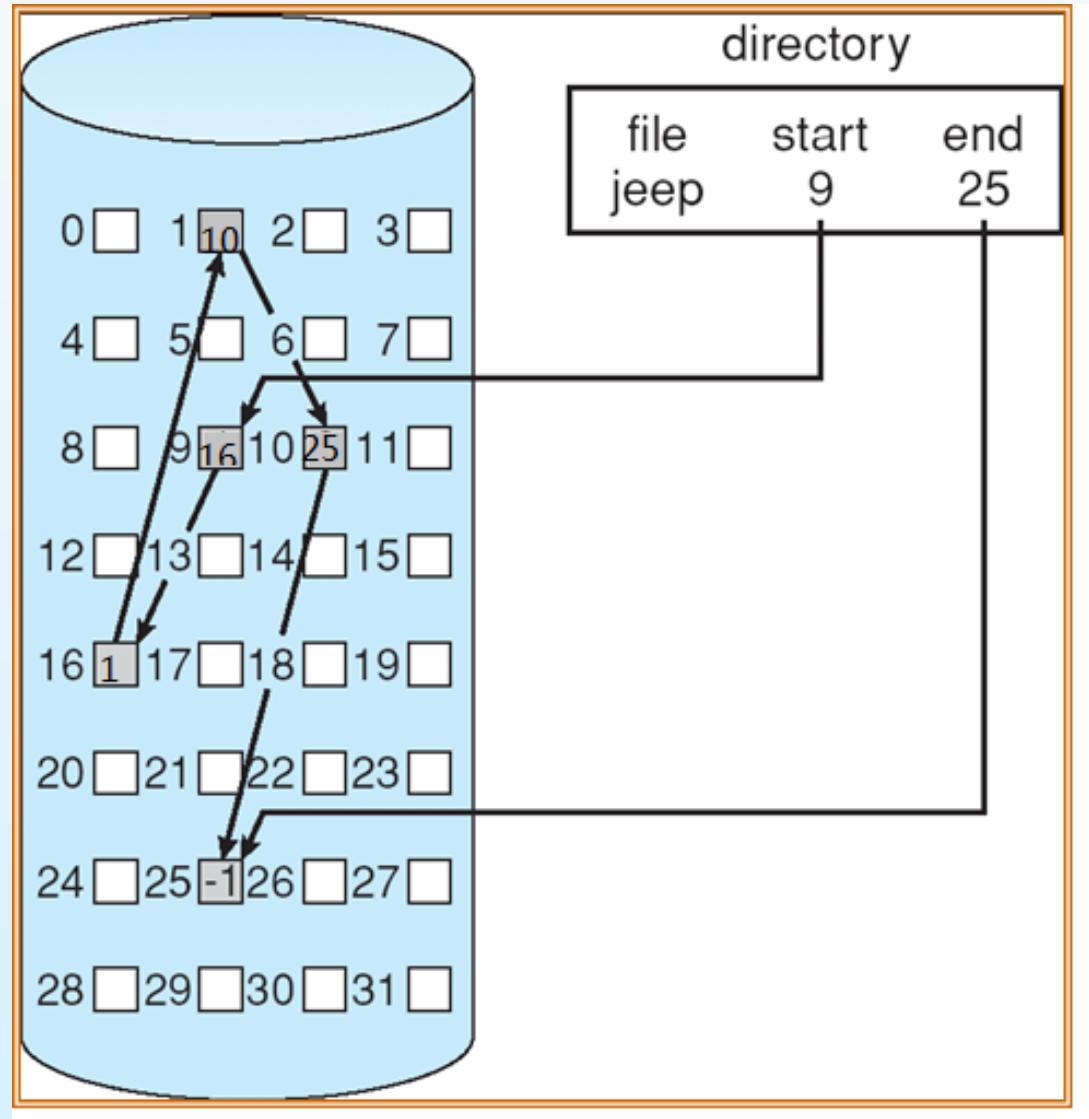
隐式链接

- 链表的指针隐藏在物理块中
- 每个物理块中的指针指向下一个物理块
- FCB给出文件首块地址
- 文件结束于空指针
- 每个物理块用于存放文件信息的空间变小
 - 减去指针占据的空间
 - 4KB物理块，指针4Bytes : 4092Bytes





隐式链接例子



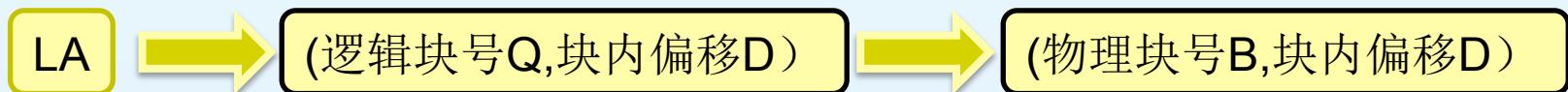


地址映射

- 逻辑地址LA: 文件内相对地址 (一维)
- 物理地址(B,D): 存在在物理块中的地址 (二维)

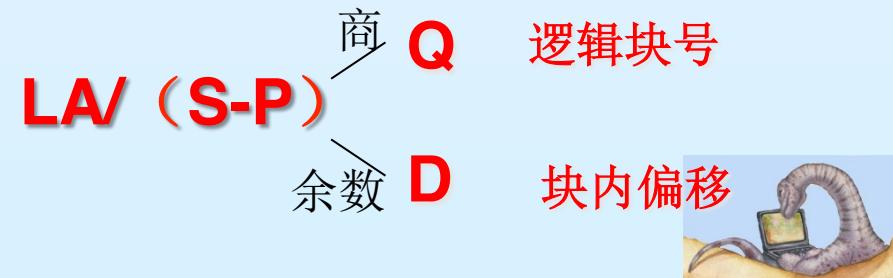


- 物理块大小: S 指针大小: P



- 物理地址

- 访问块号B= 链表中第Q项对应的物理块块号
- 块内偏移D= D





性能分析

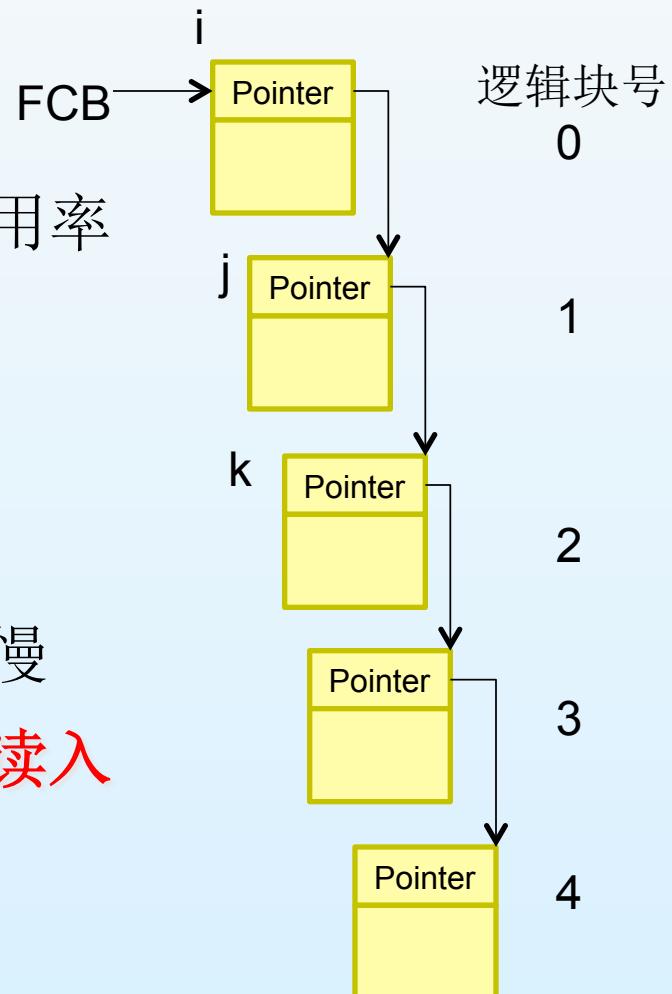
■ 优点

- 可以离散存放，提高磁盘的利用率
- 可以动态扩充文件大小
- 便于文件的插入和删除操作

■ 缺点

- 无法实现随机访问，访问文件慢
访问第*i*块，需要把0-(*i*-1)块都读入
- 可靠性差

■ 优化方法：多块集合成组





- 一个文件大小为5MB, 采用隐式链接分配。其中, 每个物理块大小为1KB, 块中指针大小为4字节。那么该文件需要占用的物理块数量是()。
 - A. 5120块
 - B. 5140块
 - C. 5141块
 - D. 以上都不是





显示链接

■ 隐式链接的问题

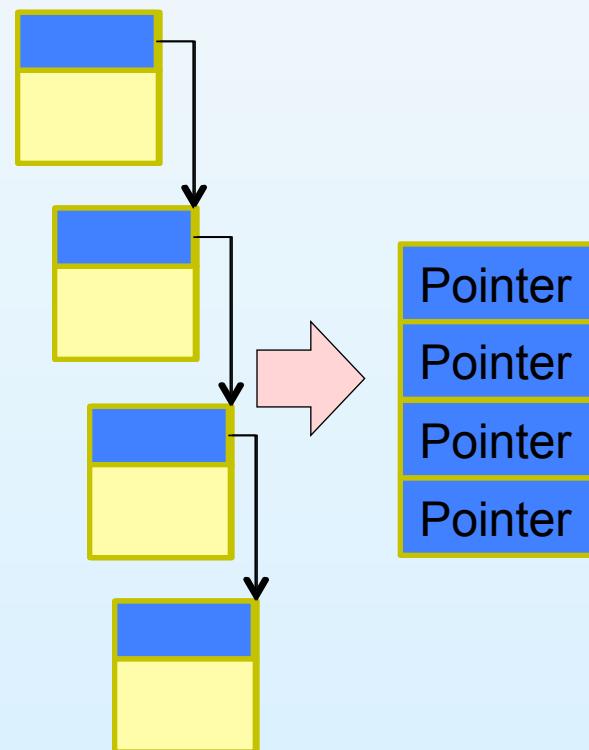
- 指针分散存放
- 为了读到一个指针而读入整个物理块

■ 显式链接

- 指针集中存放
- 把所有指针存放在一张链接表中

■ 大大提高了检索速度

- 先访问链接表，再访问物理块





显示链接

- ❖ 链接表一般在文件系统装载时装入内存
- ❖ 链接表大小
 - 表项16位：最大 $2^{16} \times 2\text{Bytes} = 128\text{KB}$
 - 表项32位：最大 $2^{32} \times 4\text{Bytes} = 16\text{GB}$
- 不适合大容量磁盘
 - 如4TB磁盘，物理块4KB
 - 链接表大小= $(4\text{TB}/4\text{KB}) \times 4\text{Bytes} = 4\text{GB}$





显示链接例子：FAT

- FAT文件系统是微软最早在MS-DOS开始使用的文件系统
- FAT (File Allocation Table)
 - FAT12
 - FAT16
 - FAT32
 - FAT64 (exFAT)

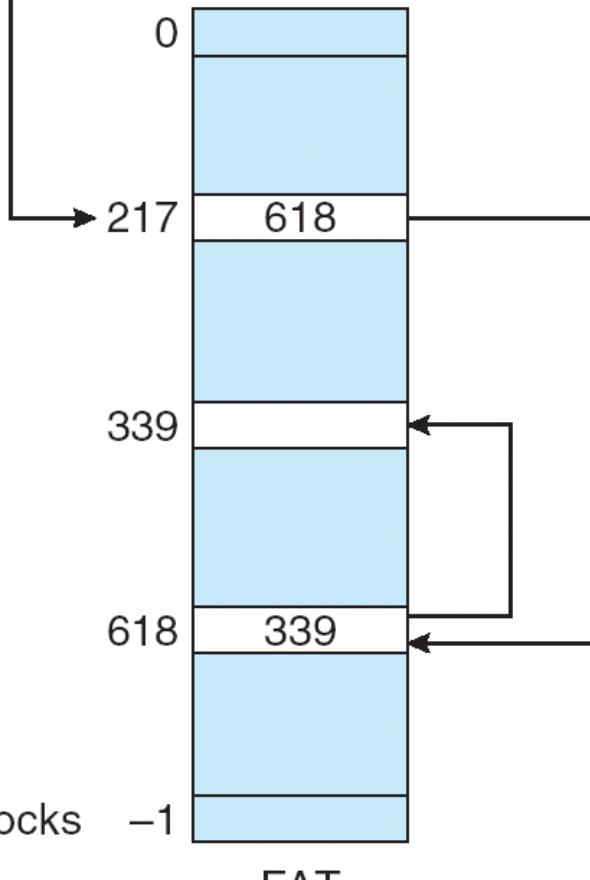
卷	布局	类型	文件系统	状态	容量
	简单	基本		状态良好 (EFI 系统分区)	100 MB
(C:)	简单	基本	NTFS	状态良好 (启动, 故障转储, 主分区)	111.57 GB
(G:)	简单	基本	exFAT	状态良好 (主分区)	119.04 GB
(I:)	简单	基本	FAT32	状态良好 (主分区)	14.31 GB

文件分配表 (FAT)

directory entry

test	...	217
------	-----	-----

name start block



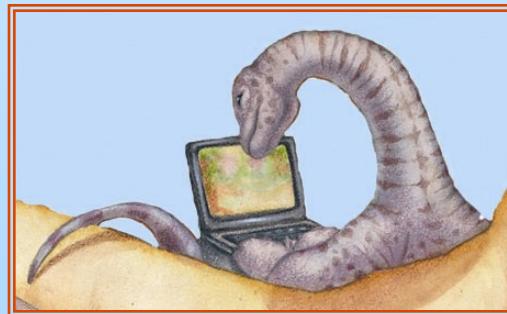


FAT 32

- 两份FAT表
- 每个簇(物理块)固定为4KB~32KB
- FAT表的表项占据32位
- FAT表最大表项数 2^{32} 项
- 单个文件不能大于4G
- FAT32管理的单个最大磁盘空间： $4KB * 2^{32} = 2TB$



索引分配





内 容

- 索引分配
- 多级索引分配
- 混合索引策略



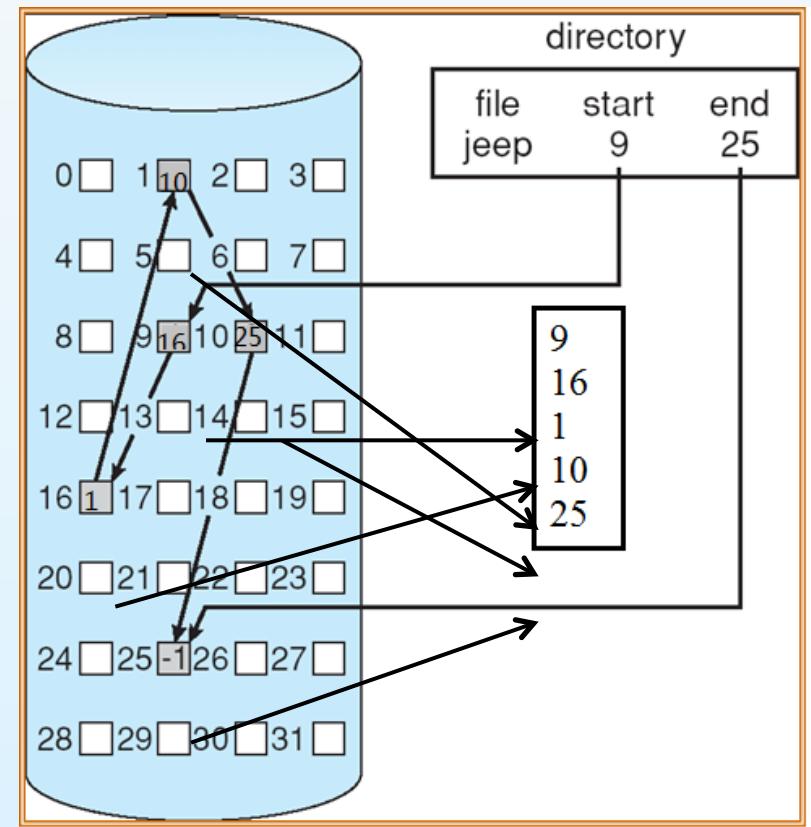
索引分配

■ 隐式链接分配问题

- 指向下一个块的指针分散在各个块中
 - ❖ 改进1: **文件分配表 (FAT)**
- 系统: 存放指针的文件分配表
- 大文件系统会导致文件分配表过大

■ 解决方法-**分散的FAT**

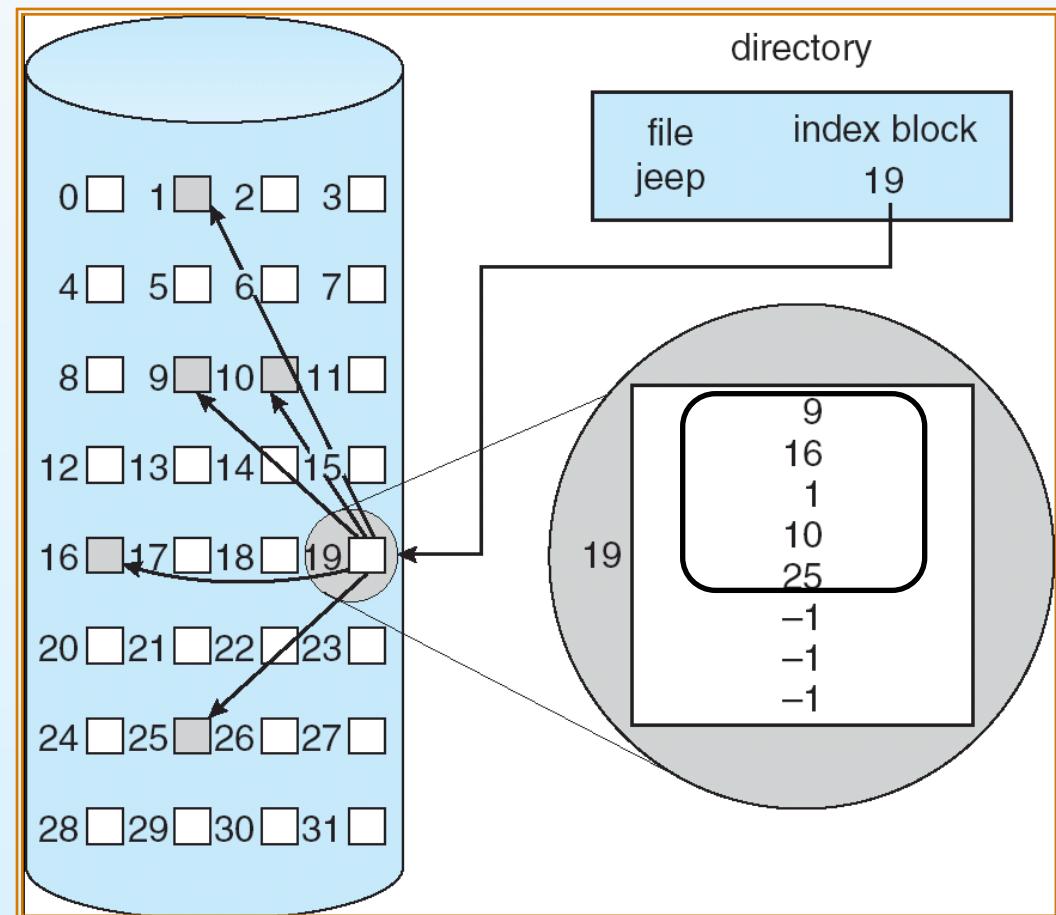
- 每个文件一张文件分配表, 即索引表





索引分配

- 索引块：存放指向文件每个物理块块号的物理块
- 索引块中的第 i 个项：存放第 i 个逻辑块对应的物理块块号
- FCB指向索引块





索引分配性能

■ 优点：

- 支持随机访问（先访问索引块，然后访问具体的物理块）
- 离散存放，没有碎片

■ 缺点

- 需要额外空间存放索引表
- 磁盘访问时间增加（物理块分布在磁盘各地）





地址映射

- 逻辑地址LA: 文件内相对地址（一维）
- 物理地址(B,D): 存在在物理块中的地址（二维）

块号B 块内偏移D

- 物理块大小: S



- 物理地址
 - 访问块号B= 索引表中第Q项存放的块号
 - 块内偏移D

LA/S Q 商 逻辑块号
 D 余数 块内偏移





多级索引

■ 大文件无法用单级索引实现

- 物理块大小 $S=4KB$
- 表项大小： 4个字节 （最多 2^{32} 块）
- 每个物理块存放的块号数目： $4K / 4 = 1K$ 个
- 单级目录最大文： $1K * 4KB = 4MB$
- 大于4MB的文件如何存放？



索引块:4KB

■ 解决方法：多级索引

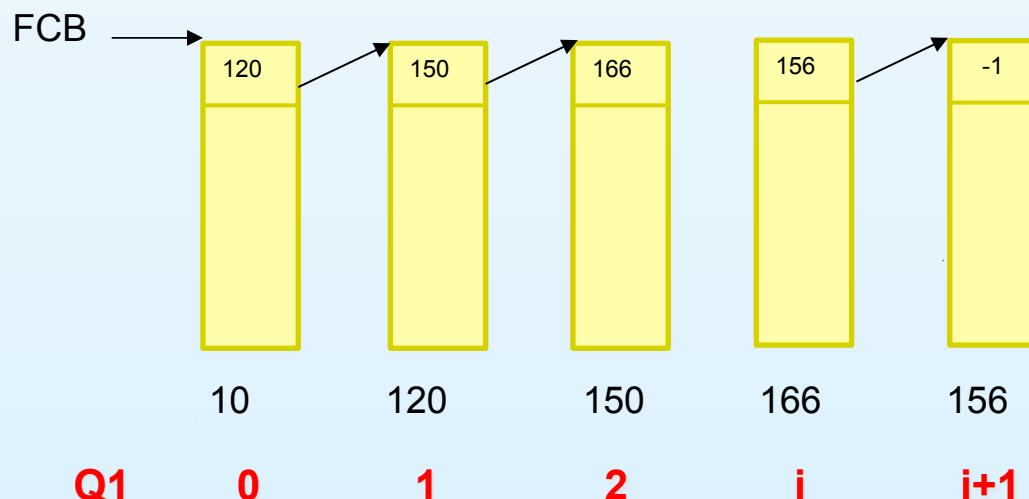




链接策略

■ 把索引块通过链表组织（没有长度限制）

- 访问块号 $B =$ 索引表中第 Q_1 块索引块中第 Q_2 项存放的块号
- 块内偏移 $D = R_2$



$Q_1 =$ 索引块块号

$$LA / (S \times (S-P)) \quad \begin{cases} Q_1 \\ R_1 \end{cases}$$

R_1 再次计算：

$$R_1 / S \quad \begin{cases} Q_2 \\ R_2 \end{cases}$$

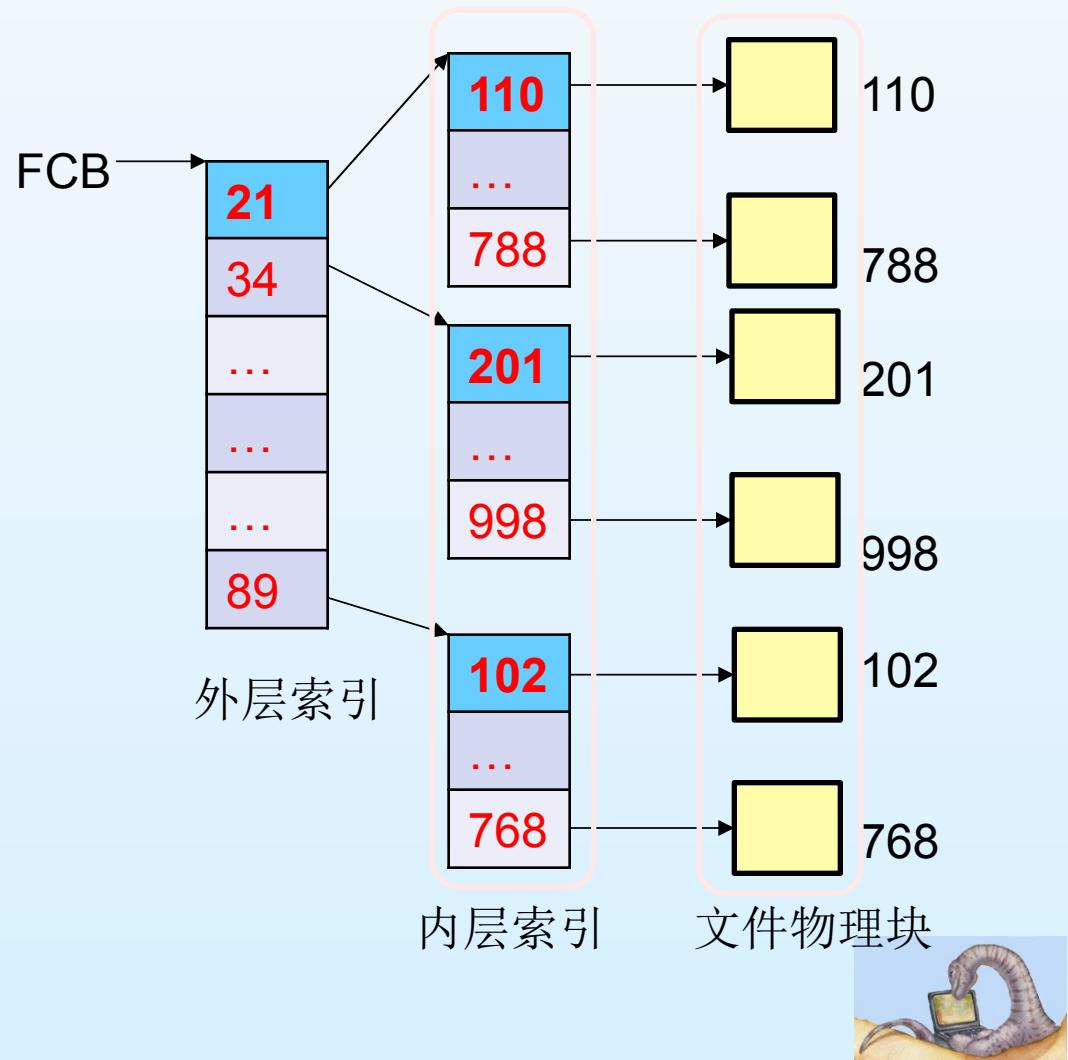
问题：可能需要读入多个索引块





多级索引-二级索引

- ❖ 把索引块通过再次索引的模式组织（有长度限制）
- ❖ 二级索引
 - ❖ 外层索引表（一个物理块）
 - ❖ 内层索引表（物理块数目=外层索引表的项数）

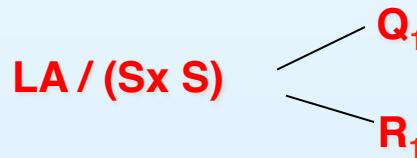




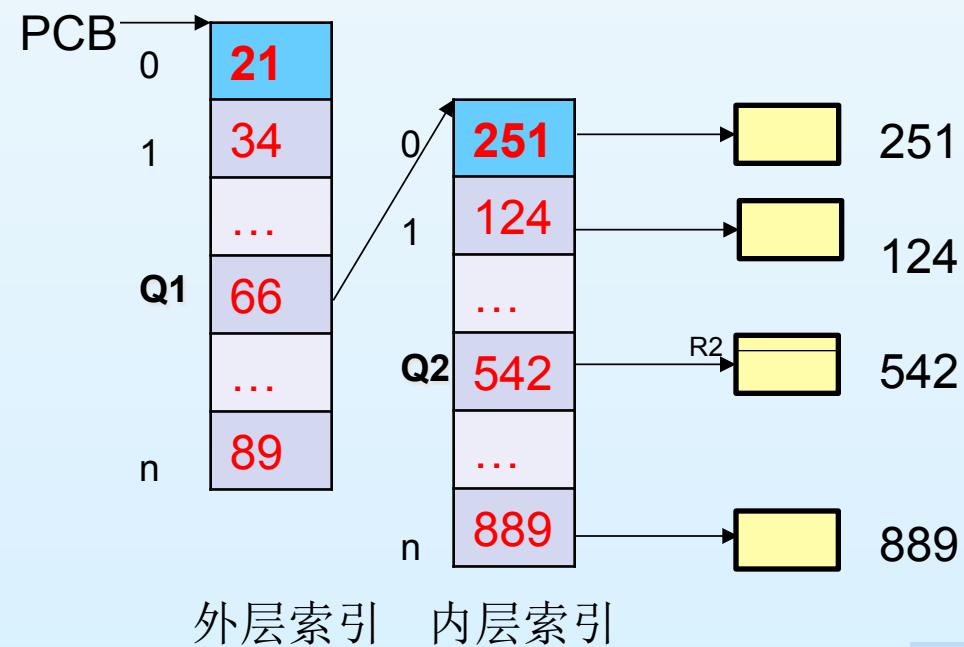
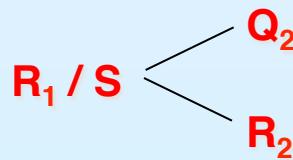
二级索引地址映射

- 访问块号 $B =$ 外层索引表中第 Q_1 项中存放的块号对应的内层索引块中第 Q_2 项中存放的块号
- 块内偏移 $D = R_2$

$Q_1 =$ 外层索引块中表项



R_1 再次计算:





索引和文件大小

■ 假如

- 物理块大小 $S=4KB$
- 索引项（块号）大小： 4Bytes

■ n级索引文件大小= $(1K)^n * 4KB$

- 单级索引： $=(1K)^1 * 4KB = 4MB$
- 二级索引： $=(1K)^2 * 4KB = 4GB$
- 三级索引： $=(1K)^3 * 4KB = 4TB$
- 四级索引： $=(1K)^4 * 4KB = 4PB$





联合策略：UNIX（每块4KB）

■ 多种索引混合

■ 0级索引

- FCB中12个指针指向文件开头的12个逻辑块

■ 一级索引

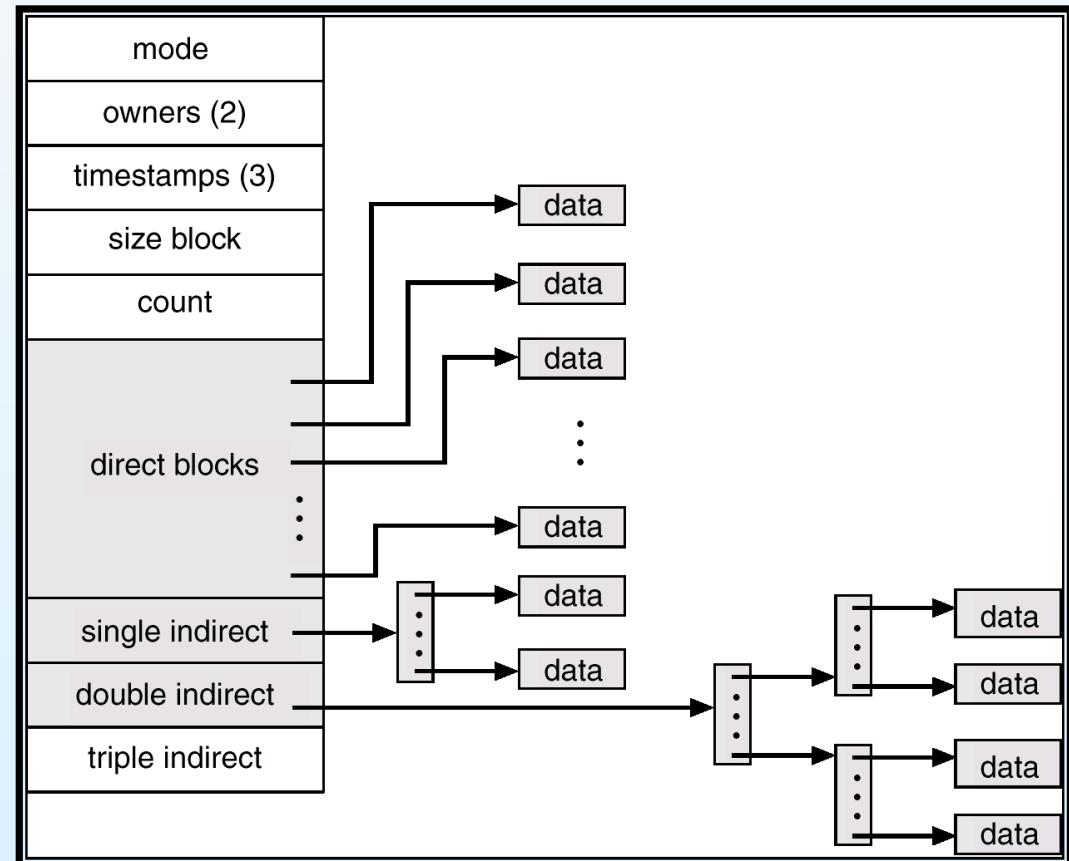
- 1K项，对应1K个物理块，总大小为4MB

■ 二级索引

- 1M块，容量为4GB

■ 三级索引

- 1G块，容量为4TB

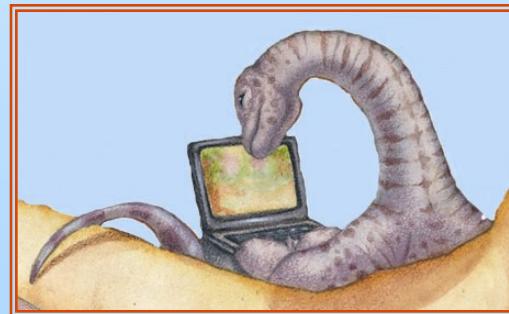




性能

- 好的分配方法依赖于访问类型
 - 连续分配可用于随机或顺序访问，效率高
 - 链接分配适合顺序访问，不适合随机访问
 - 在文件创建时根据访问类型选择链接还是连续
 - 索引分配更加复杂
 - 依赖于索引结构、文件大小、块大小
 - 增加磁盘I/O速度是提高性能的一个因素
 - Intel Core i7 990x 3.46Ghz = 159,000 MIPS(Million Instructions Per Second)
 - 典型的磁盘 250 I/Os/秒
 - ▶ 每次I/O操作时间可以执行 $159,000 \text{ MIPS} / 250 = 630 \text{ MIPS}$
 - 快速SSD 60,000 Ios/秒
 - ▶ 每次I/O操作时间可以执行 $159,000 \text{ MIPS} / 60,000 = 2.65 \text{ MIPS}$
- 

5、空闲空间管理





内容

■ 文件系统不仅需要记录下磁盘上还没有分配出去的空闲物理块，还需要为新文件分配物理块和回收被删除文件的物理块，所以文件系统需要包含空闲空间管理模块

- 空闲表
 - 空闲链表
 - 位示图
 - 成组链接
 - 一致性检查
 - 空闲空间整理
- } 空闲空间管理技术





空闲空间管理方法

- 空闲表
- 空闲链表
- 位示图
- 成组链接



空闲表

- 空闲区：连续的未分配物理块集合
- 空闲表
 - 每个表项对应一个空闲区
 - 内容：空闲区的第一块号、空闲块的块数
- 空闲表适用连续分配
- 分配和回收方式：和内存的连续分配类似
- 缺点：需要额外空间来存放空闲表

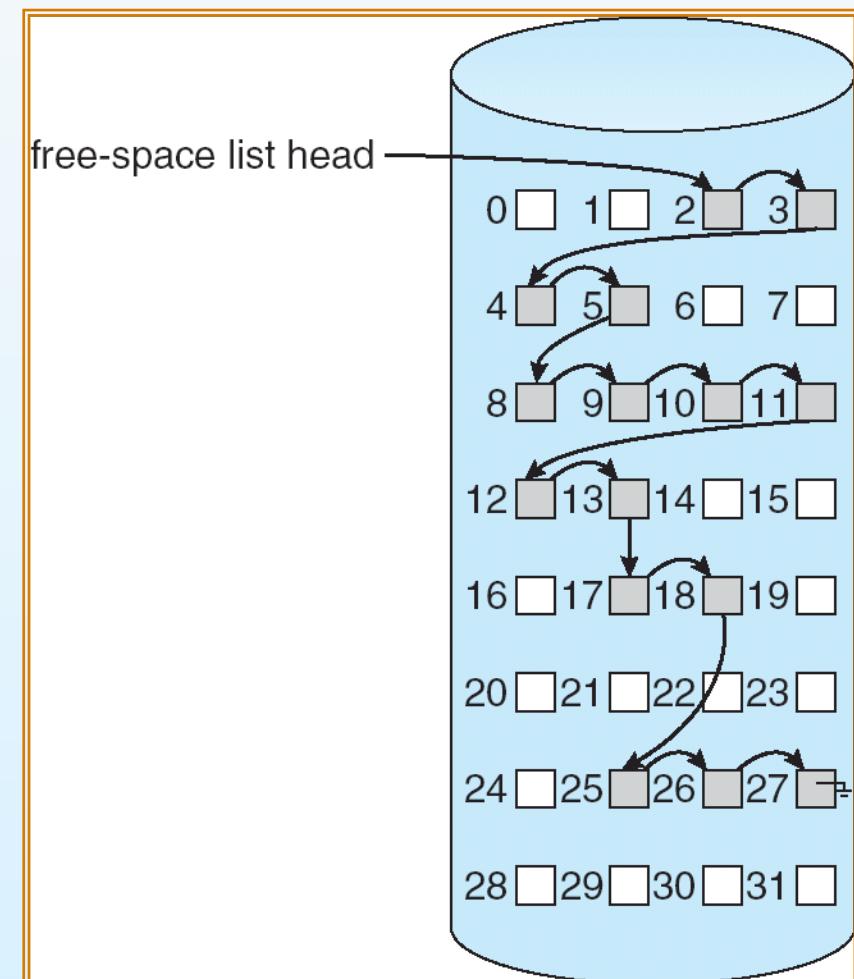
首块块号	长度
13	21
55	67
150	122
...	...
...	...
1990	12





空闲链表

- 将磁盘上所有空闲块链接在一个链表中
- 分配：从链表头依次摘下适当数目的空闲块
- 回收：空闲块加入链表尾部
- 优点：不需专用块存放管理信息，不浪费空间
- 缺点：增加I/O操作，得到连续空间难





位示图

- 利用二进制一位 (bit) 来表示一个块的使用情况
 - 1: 盘块空闲
 - 0: 盘块已分配
- 所有块都有一个二进制位与之对应
- 所有块对应的位形成位示图

第0字节
第1字节
第2字节
第3字节

7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1





位示图

- 位示图需要额外的空间。例如

block size = 2^{12} bytes

disk size = 2^{30} bytes (1 gigabytes)

$n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)

- 位示图存放在物理块中
- 分配: **1->0** 回收: **0->1**
- 比较容易得到连续物理块
- Linux和Windows常用的方式





位示图:块号计算

■ 已知位示图中位置

- 第*i*个字节
- 第*j*位

块号 $k = i * 8 + (7 - j)$

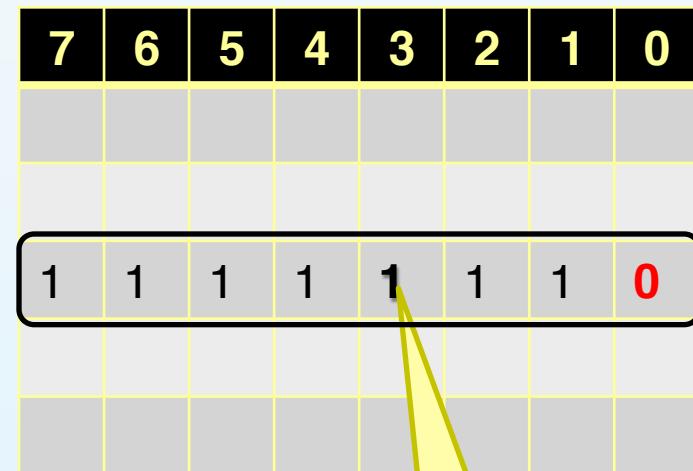
■ 已知块号*k*

- 位示图中字节*i*

$$i = [k/8]$$

- 该字节中的位*j*

$$j = 7 - k \% 8$$



第2个字节
的第3位

块号是 $2 * 8 + (7 - 3) = 20$
对应第20个物理块





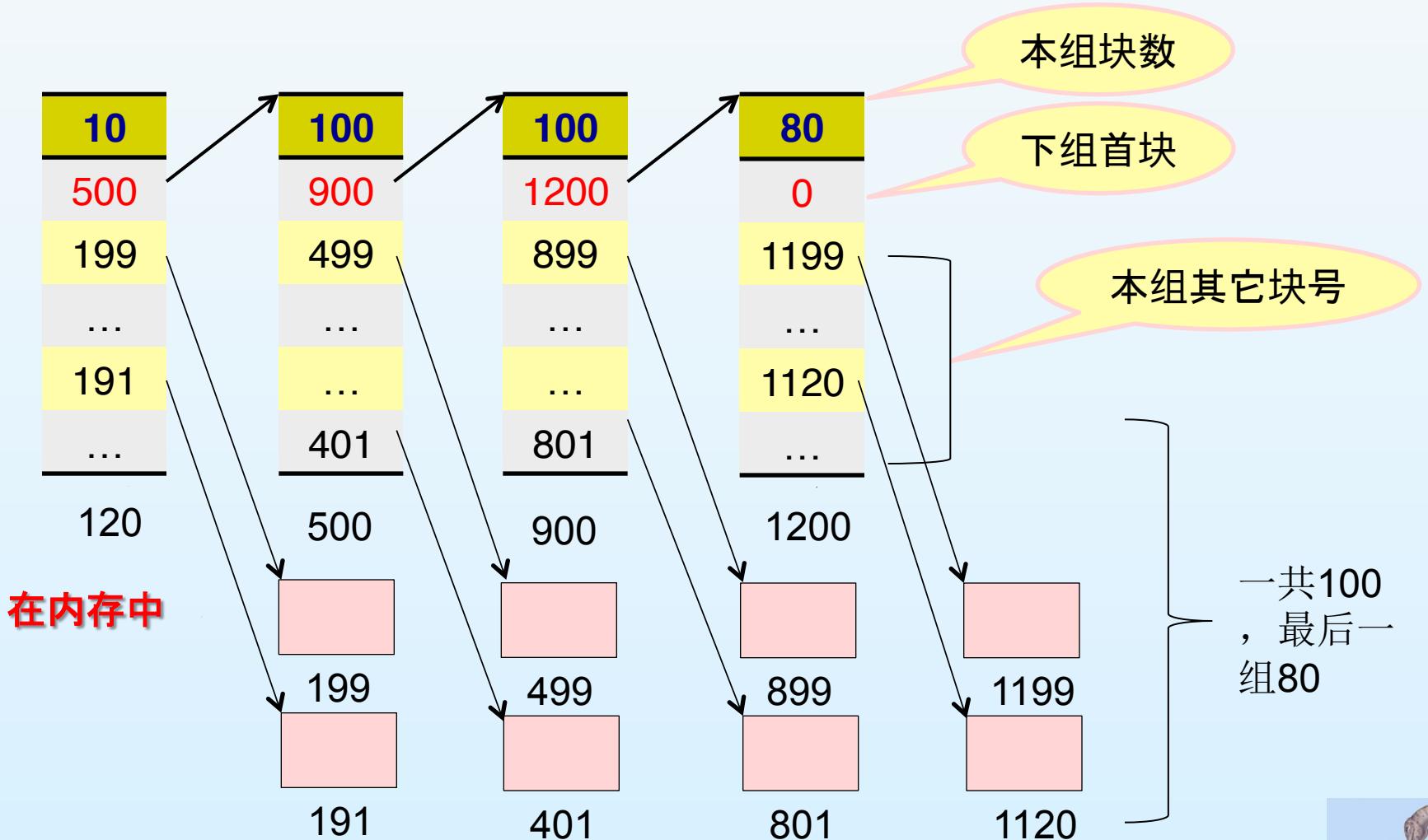
成组链接

- 结合了空闲表和空闲链表
- 例子:UNIX系统
 - 将空闲块分成若干组，每100个块为一组
 - 每组第一空闲块：
 - ▶ 空闲块总数
 - ▶ 下一组空闲块首块的块号
 - ▶ 本组其它空闲块的块号列表
 - 如果下一组空闲块首块的块号等于0，则该组是最后一组





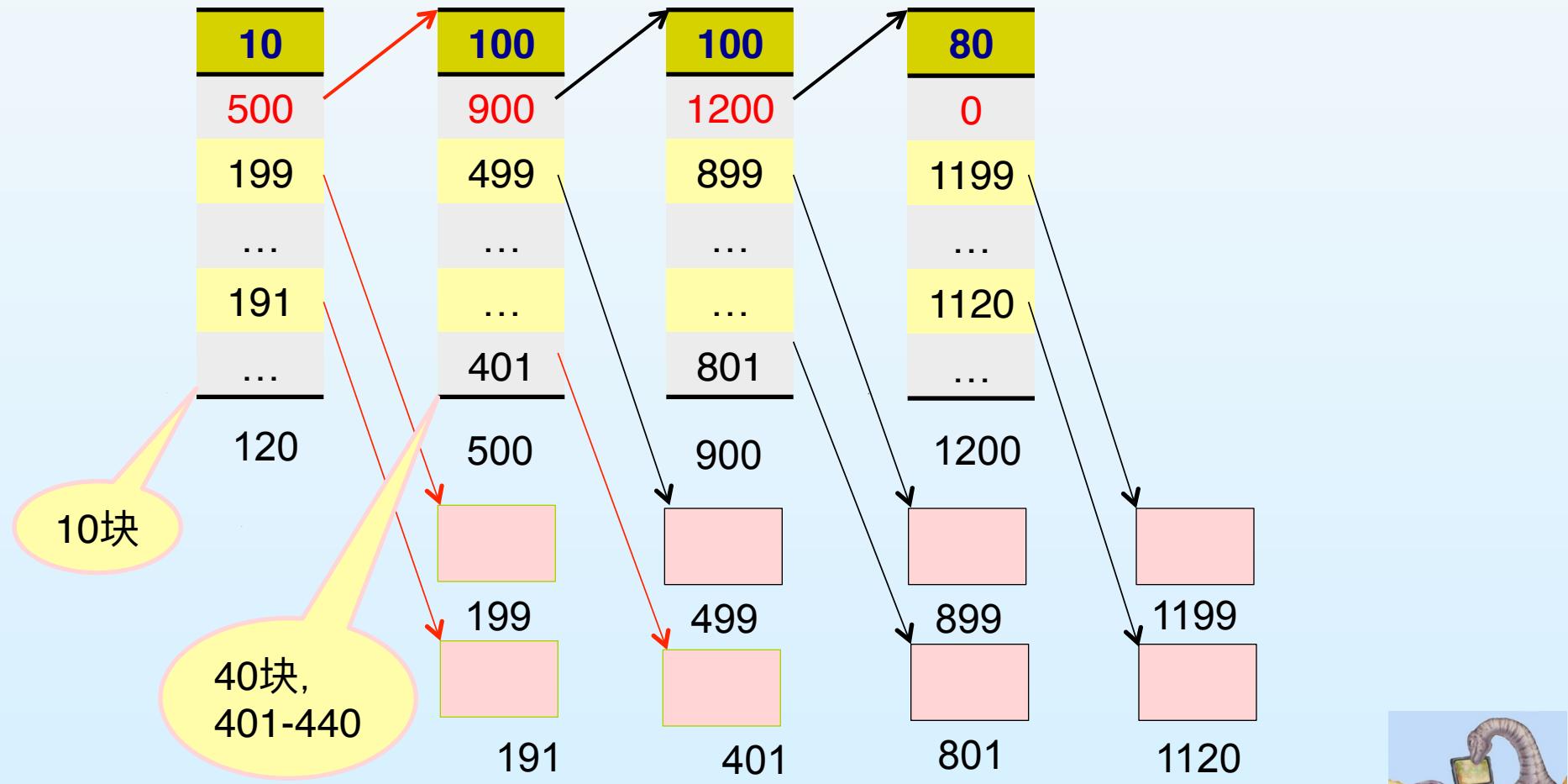
成组链接例子



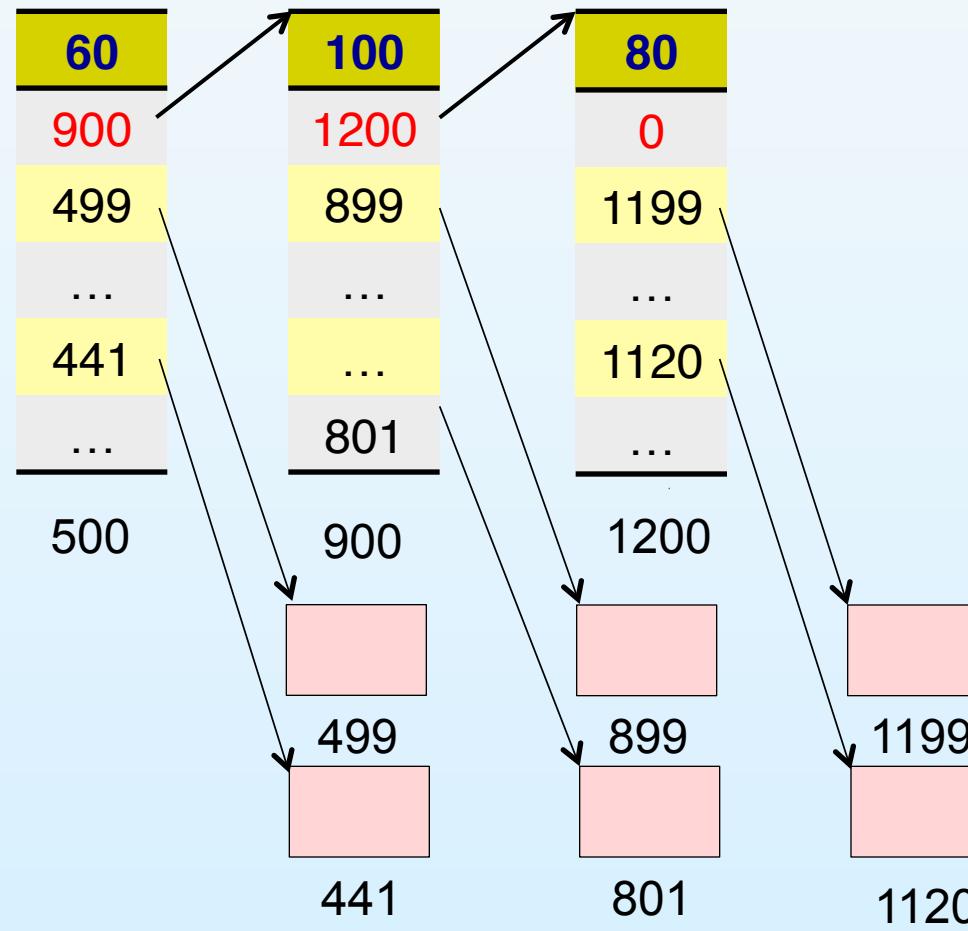


分配50块空闲块

■ 从前往后分配，从第一组开始分配，分完再分下组



分配50块空闲块





空闲块回收

- 先将释放的空闲块放到第一组
- 满100块后，在第一组前再开辟一组
- 原来的第一组变成第二组，新组为第一组





一致性检查

- 每个文件的目录或FCB中会记录这个文件分配到的物理块信息，位示图同样会记录每个物理块是否被使用，这两者应该是一致的
- 即文件系统中所有的物理块减去文件用掉的，剩余的都是空闲块。但是有时不是这样，存不一致性
- 所以，文件系统提供了一致性检查，将目录结构数据与磁盘空闲块结构相比较，纠正发现的不一致
 - 空闲块在某个文件的物理块中（A）
 - 非空闲块不属于任意一个文件（B）
 - 一个物理块属于多个文件（C）

文件使用块

7	6	5	4	3	2	1	0
1	1	0	0	1	1	1	0

7	6	5	4	3	2	1	0
1	1	0	0	1	1	1	0

7	6	5	4	3	2	1	0
1	1	0	0	2	1	1	0

空闲块

7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	1

7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	1

7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1

A

B

C

