

操作系统原理

PRINCIPLES OF OPERATING SYSTEM

北京大学计算机科学技术系 陈向群

Department of computer science and Technology

Peking University

2015 春季

第10讲

文件系统2

文件管理

- ◎ 文件系统实例—FAT
- ◎ 文件操作的实现
- ◎ 文件系统的管理
- ◎ 文件系统的安全性
- ◎ 文件系统的性能问题
- ◎ 文件系统的两个练习

文件系统实例——FAT



WINDOWS — FAT16文件系统

- 簇大小：1、2、4、8、16、32或64扇区
- 文件系统的数据记录在“引导扇区”中
- 文件分配表FAT的作用
描述簇的分配状态、标注下一簇的簇号等
- **FAT表项：2字节**
- 目录项：**32字节**
- 根目录大小固定



FAT12、FAT16、FAT32

引导区	文件分配表1	文件分配表2	根目录	其他目录和文件
-----	--------	--------	-----	---------

FAT文件系统——MBR

◎ 主引导记录 — 0号扇区

字节偏移量 (16进制)	域长	含义
00 – 1BD	446字节	引导代码
1BE – 1CD	16字节	分区表项1
1CE – 1DD	16字节	分区表项2
1DE – 1ED	16字节	分区表项3
1EE – 1FD	16字节	分区表项4
1FE – 1FF	2字节	扇区结束标记 (55AA)



FAT文件系统——DBR

FAT32

引导区	文件分配表1	文件分配表2	根目录	其他目录和文件
-----	--------	--------	-----	---------



字节偏移量 (16进制)	域长	样值 (16进制)	含义
00	3字节	EB 3C 90	转移指令
03	8字节	MSDOS5.0	文件系统标志(ASCII码)
0B	25字节		BIOS参数块 (BIOS Parameter Block, BPB)
24	54字节		扩展BIOS参数块(Extended BIOS Parameter Block, EBPB)
5A	410字节		引导代码
1FE	2字节	55 AA	扇区结束标记

引导扇区—BIOS参数块

FAT32

字节偏移量 (16进制)	域长	样值(16进制)	含义
0B	2字节	00 02	每扇区字节数
0D	1字节	08	每簇扇区数
0E	2字节	01 00	保留扇区数: 从分区引导扇区到第一个文件分配表开始的扇区数
10	1字节	02	文件分配表个数
11	2字节	00 02	根目录项数
13	2字节	00 00	扇区数(小): 卷上的扇区数, 如果该数适合于16位(65535)的话
15	1字节	F8	介质类型: F8表明为硬盘, F0表明为软盘
16	2字节	C9 00	每个文件分配表的扇区数(FAT32不用)
18	2字节	3F 00	每磁道扇区数
1A	2字节	10 00	磁头数
1C	4字节	3F 00 00 00	隐藏扇区数
20	4字节	51 42 06 00	扇区数(大): 如果小扇区数域的取值为0, 该域包含的是卷中的扇区总数

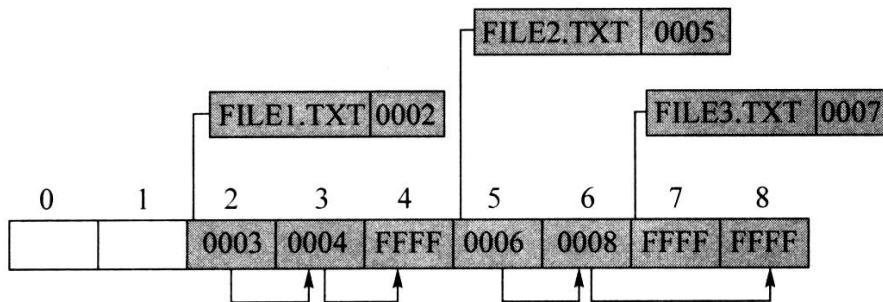
引导扇区—扩展BIOS参数块(EBPB)

FAT32

字节偏移量(16进制)	域长	含义
→ 24	4字节	每个文件分配表的扇区数(FAT32用)
28	2字节	标记: bit7为1, 表示只有一个FAT; 否则, 两个FAT互为镜像
2A	2字节	版本号
→ 2C	2字节	根目录起始簇号, 通常为2
30	2字节	FSINFO所在扇区, 通常1扇区
32	2字节	引导扇区备份, 通常是6号扇区
34	12字节	未用
40	1字节	BIOS Int 13H设备号
41	1字节	未用
42	1字节	扩展引导标识, 如果后面三个值有效, 设为0x29
43	4字节	卷序列号: 当格式化卷时创建的一个唯一的数字
47	11字节	卷标(ASCII码), 建立文件系统时由用户指定
52	8字节	系统ID: 根据磁盘的格式, 该域的取值为FAT12或FAT16

文件分配表FAT

- 可以把文件分配表看成是一个整数数组，每个整数代表磁盘分区的一个簇号
- 状态
 - 未使用、坏簇、系统保留、被文件占用（下一簇簇号）、最后一簇（0xFFFF）
- 簇号从0开始编号，簇0和簇1是保留的



FAT16目录项

偏移	域长	含义
00h	8	文件名
08h	3	文件扩展名
0Bh	1	文件属性字节
0Ch	10	保留
16h	2	最后一次修改的时间
18h	2	最后一次修改的日期
1Ah	2	起始簇号
1Ch	4	文件大小



位	7-6	5	4	3	2	1	0
	保留	归档	目录	卷标	系统	隐藏	只读

FAT32文件系统

- ◎ **FAT32**的根目录区（**ROOT区**）不是固定区域、固定大小，而是数据区的一部分，采用与子目录文件相同的管理方式
- ◎ 目录项仍占**32**字节，但分为各种类型（包括：“.”目录项、“..”目录项、短文件名目录项、长文件名目录项、卷标项（根目录）、已删除目录项（第一字节为**0xE5**）等）
- ◎ 支持长文件名格式
- ◎ 支持Unicode
- ◎ 不支持高级容错特性，不具有内部安全特性

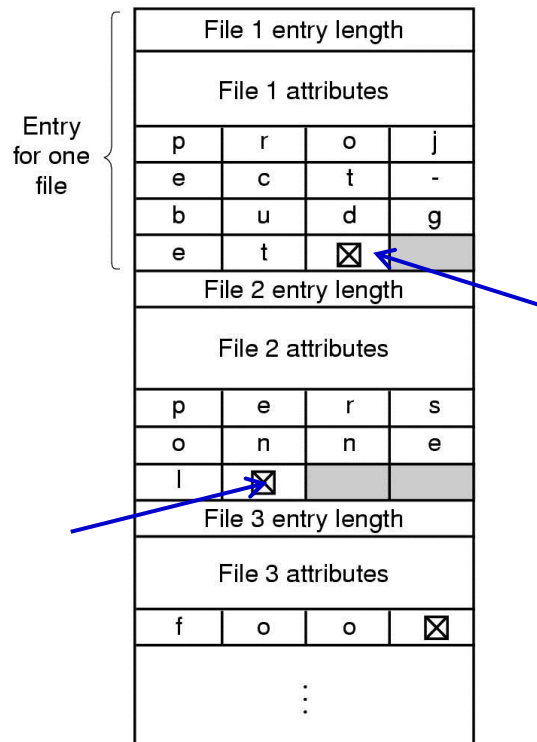


FAT32目录项

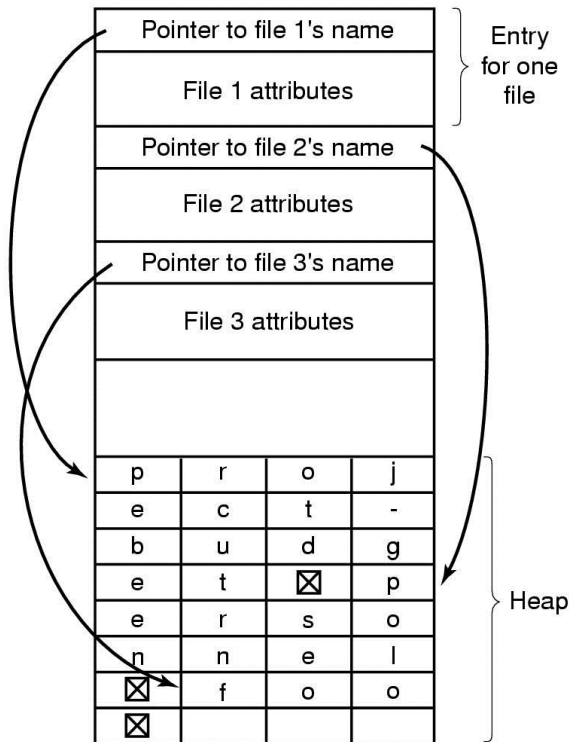
7	6	5	4	3	2	1	0
		归档	目录	卷标	系统	隐藏	只读

偏移	域长(字节)	含义
00h	8	文件名
08h	3	文件扩展名
0Bh	1	文件属性字节
0Ch	1	保留
0Dh	1 →	创建时间，精确到1/10秒
0Eh	2 →	文件创建时间
10h	2 →	文件创建日期
12h	2 →	文件最后访问日期
14h	2 →	起始簇号的高16位
16h	2	最后一次修改的时间
18h	2	最后一次修改的日期
1Ah	2	起始簇号的低16位
1Ch	4	文件长度（子目录为0）

一般长文件名的实现方式



(a)



(b)

FAT32—长文件名目录项格式

偏移	长度	含义
00h	1	位0-5给出序号, 位6 表示长文件最后一个目录项
01h	10	长文件名(unicode码) ① 前5个字符
0Bh	1	0x0F(长文件名目录项标志)
0Ch	1	保留
0Dh	1	校验值(由短文件名计算得出)
0Eh	12	长文件名(unicode码) ② 6个字符
1Ah	2	文件起始簇号, 常置0
1Ch	4	长文件名(unicode码) ③ 2个字符

例子：文件名为The quick brown.fox，采用Unicode编码

第2个长文件名目录项(最后一个)

长文件名目录项属性标识

0x42	w	n	.	f	o	0x0F	0x00	check sum	x
0x0000	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0x0000	0xFFFF	0xFFFF		
0x01	T	h	e	q		0x0F	0x00	check sum	u
i	c	k	b		0x0000	r	o		
T	H	E	Q	U	I	~	1	F	O
X						0x20	NT	Create Time	
Create Date	Last Access Date	0x0000	Last Modified Time	Last Modified Date	First Cluster			File Size	

短目录项

第1个长文件名目录项

5个目录项



The quick brown fox jumps over the lazy dog

Bytes	68	d	o	g	A	0	C	K							0				
	3	o	v	e	A	0	C	K	t	h	e		l	a	0	z	y		
	2	w	n		f	o	A	0	C	K	x		j	u	m	p	0	s	
	1	T	h	e		q	A	0	C	K	u	i	c	k		b	0	r	o
	T	H	E	Q	U	I	~	1	A	N	S	Creation	Last	Upp	Last		Low		Size

Windows 为其建立了五个目录项、四个保存长文件名、一个保存压缩文件名 THEQUI~1

文件操作的实现

文件操作的实现

创建文件:

建立系统与文件的联系，实质是建立文件的FCB

- 在目录中为新文件建立一个目录项，根据提供的参数及需要填写相关内容
- 分配必要的存储空间

打开文件:

根据文件名在文件目录中检索，并将该文件的目录项读入内存，建立相应的数据结构，为后续的文件操作做好准备

文件描述符/文件句柄

文件操作—建立文件

create（文件名，访问权限）

① 检查参数的合法性

例如：文件名是否符合命名规则；

有无重名文件；

合法→②，否则→报错、返回

② 申请空闲目录项，并填写相关内容；

③ 为文件申请磁盘块；（?）

④ 返回

引导记录

超级数据块

空闲区管理

I-节点区

根目录区

文件和目录区

文件操作—打开文件

为文件读写做准备

给出文件路径名，获得文件句柄(file handle)或文件描述符(file descriptor)，需将该文件的目录项读到内存

fd=open (文件路径名, 打开方式)

- ① 根据文件路径名查目录，找到目录项 (或I节点号)；
- ② 根据文件号查系统打开文件表，看文件是否已被打开；
是 → 共享计数加1
否则 → 将目录项 (或I节点)等信息填入系统打开文件表空表项，共享计数置为1；
- ③ 根据打开方式、共享说明和用户身份检查访问合法性；
- ④ 在用户打开文件表中获取一空表项，填写打开方式等，并指向系统打开文件表对应表项

返回信息：**fd**：文件描述符，是一个非负整数，用于以后读写文件

文件操作—指针定位

seek (fd, 新指针的位置)

系统为每个进程打开的每个文件维护一个读写指针，即相对于文件开头的偏移地址（读写指针指向每次文件读写的开始位置，在每次读写完成后，读写指针按照读写的数据量自动后移相应数值）

- ① 由fd查用户打开文件表，找到对应的表项；
- ② 将用户打开文件表中文件读写指针位置设为新指针的位置，供后继续读写命令存取该指针处文件内容

文件操作—读文件

read (文件描述符, 读指针, 要读的长度, 内存目的地址)

① 根据打开文件时得到的文件描述符, 找到相应的文件控制块 (目录项)

确定读操作的合法性

读操作合法→②, 否则→出错处理

问题: 文件尚未打开?

② 将文件的逻辑块号转换为物理块号

根据参数中的读指针、长度与文件控制块中的信息, 确定块号、块数、块内位移

③ 申请缓冲区

④ 启动磁盘I/O操作, 把磁盘块中的信息读入缓冲区, 再传送到指定的内存区 (多次读盘)

⑤ 反复执行③、④直至读出所需数量的数据或读至文件尾

讨论

怎样实现系统调用
rename (给文件重命名) ?



文件系统的管理

文件系统的可靠性

可靠性:

抵御和预防各种物理性破坏和人为性破坏的能力

- ◎ 坏块问题
- ◎ 备份

通过转储操作，形成文件或文件系统的多个副本

文件系统备份

全量转储：

定期将所有文件拷贝到后援存储器

增量转储：

只转储修改过的文件，即两次备份之间的修改，减少系统开销

物理转储：

从磁盘第0块开始，将所有磁盘块按序输出到磁带

逻辑转储：

从一个或几个指定目录开始，递归地转储自给定日期后所有更改的文件和目录

文件系统一致性

问题的产生：

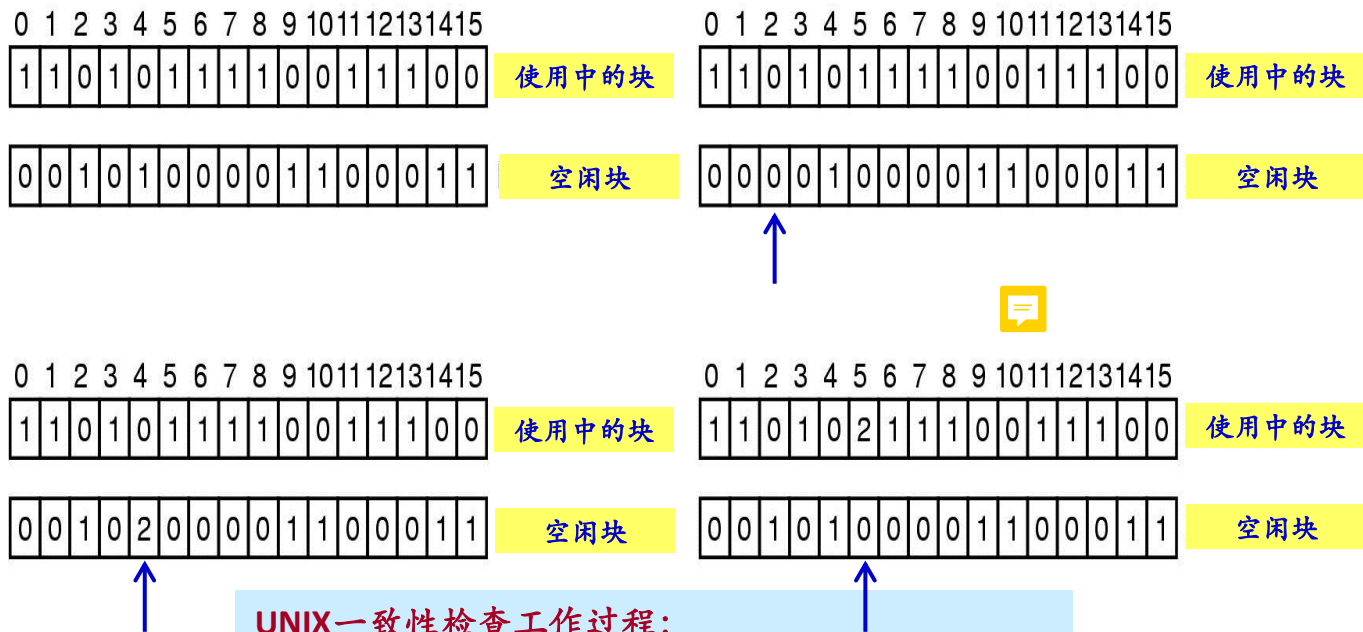
磁盘块 → 内存 → 写回磁盘块

若在写回之前，系统崩溃，则文件系统出现不一致

解决方案：

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

磁盘块的一致性检查



UNIX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


文件系统的写入策略

- 通写（**write-through**）

内存中的修改立即写到磁盘

缺点：速度性能差

例：FAT文件系统



考虑文件系统
一致性和
速度

- 延迟写（**lazy-write**）

利用回写（**write back**）缓存的方法得到高速

可恢复性差

- 可恢复写（**transaction log**）

采用事务日志来实现文件系统的写入

既考虑安全性，又考虑速度性能

例：NTFS

确保未经授权的用户不能存取某些文件

文件系统的安全性

文件保护机制

- ⊙ 用于提供安全性、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

实现：

- * 用户身份验证
- * 访问控制

用户是谁？
用户拥有什么？
用户知道什么？

文件的访问控制

访问控制

主动控制：访问控制表

- ✓ 每个文件一个
- ✓ 记录用户ID和访问权限
- ✓ 用户可以是一组用户
- ✓ 文件可以是一组文件

能力表(权限表)

- ✓ 每个用户一个
- ✓ 记录文件名及访问权限
- ✓ 用户可以是一组用户
- ✓ 文件可以是一组文件

UNIX的文件访问控制

采用文件的二级存取控制

审查用户的身份、审查操作的合法性

第一级：对访问者的识别
对用户分类：

- ✓ 文件主 (owner)
- ✓ 文件主的同组用户 (group)
- ✓ 其他用户 (other)

第二级：对操作权限的识别
对操作分类：

- ✓ 读操作 (r)
- ✓ 写操作 (w)
- ✓ 执行操作 (x)
- ✓ 不能执行任何操作 (-)

例子： `rwX rwX rwX`

`chmod 711 file1` 或 `chmod 755 file2`

各种提高文件系统性能的方法

文件系统的性能1

文件系统的性能问题

磁盘服务

→ 速度成为系统性能的主要瓶颈之一

设计文件系统应尽可能减少磁盘访问次数

提高文件系统性能的方法：

目录项(FCB)分解、当前目录、磁盘碎片整理

块高速缓存、磁盘调度、提前读取、合理分配磁盘空间、信息的优化分布、RAID技术... ..

块高速缓存(BLOCK CACHE)

又称为文件缓存、磁盘高速缓存、缓冲区高速缓存
是指：在内存中为磁盘块设置的一个缓冲区，保存了
磁盘中某些块的副本

- 检查所有的读请求，看所需块是否在块高速缓存中
- 如果在，则可直接进行读操作；否则，先将数据块读入块高速缓存，再拷贝到所需的地方
- 由于访问的局部性原理，当一数据块被读入块高速缓存以满足一个I/O请求时，很可能将来还会再次访问到这一数据块

关于实现

块高速缓存满时需要进行置换

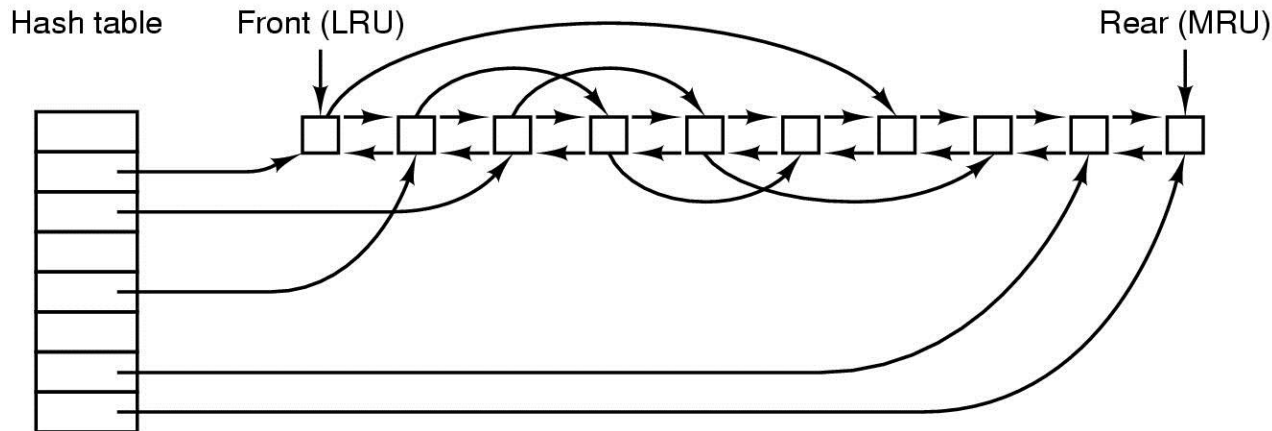
- 块高速缓存的组织

- 块高速缓存的置换（修改LRU）

该块是否不久后会再次使用

- 块高速缓存写入策略

该块是否会影响文件系统的一致性



提前读取

- 思路：每次访问磁盘，多读入一些磁盘块
- 依据：程序执行的空间局部性原理
- 开销：较小(只有数据传输时间)
- 具有针对性

WINDOWS 的文件访问方式(1/3)

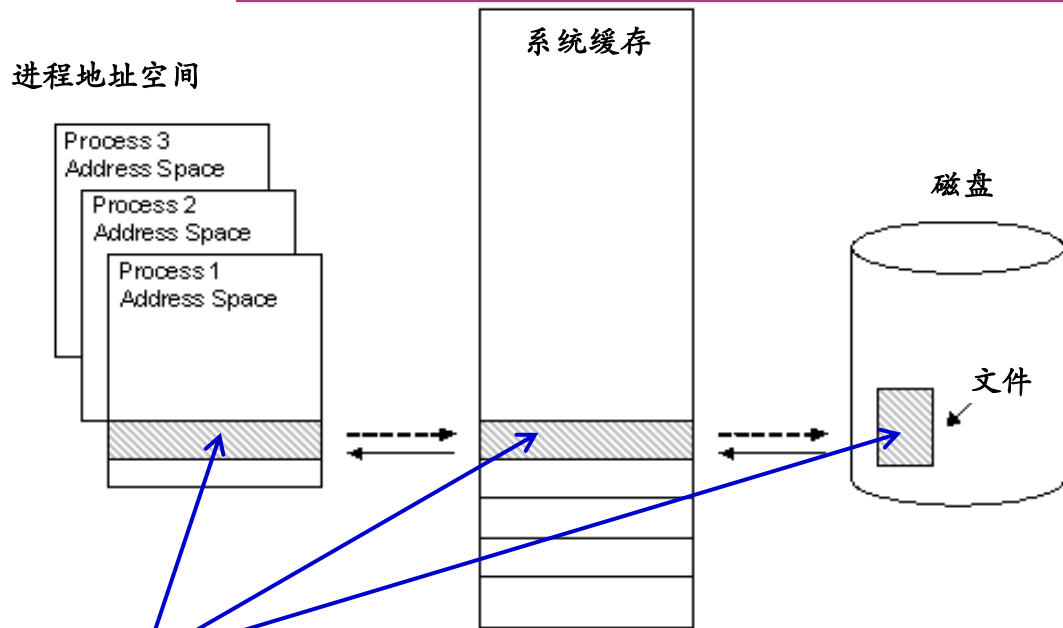
- ◉ 不使用文件缓存
 - ◉ 普通方式
 - ◉ 通过Windows提供的FlushFileBuffer函数实现
- ◉ 使用文件缓存
 - ◉ 预读取。每次读取的块大小、缓冲区大小、置换方式
 - ◉ 写回。写回时机选择、一致性问题
- ◉ 异步模式
 - ◉ 不再等待磁盘操作的完成
 - ◉ 使处理器和I/O并发工作

WINDOWS 的文件访问方式(2/3)

用户对磁盘的访问通过访问文件缓存来实现

- 由Windows的Cache Manager实现对缓存的控制
 - 读取数据的时候预取
 - 在Cache满时，根据LRU原则清除缓存的内容
 - 定期更新磁盘内容使其与Cache一致（1秒）
- Write-back机制
 - 在用户要对磁盘写数据时，只更改Cache中的内容，由Cache Manager决定何时将更新反映到磁盘

WINDOWS 的文件访问方式(3/3)



阴影部分为需要访问的数据，数据在磁盘、系统缓存和进程地址空间有3份拷贝，通常下用户对数据的修改并不直接反映到磁盘上，而是通过write-back机制由lazy writer定期地更新到磁盘

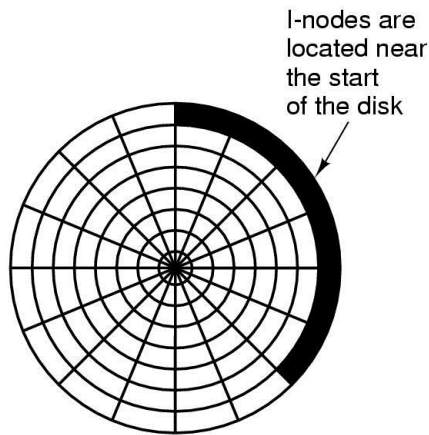
合理分配磁盘空间

分配磁盘块时，把有可能顺序存取的块放在一起

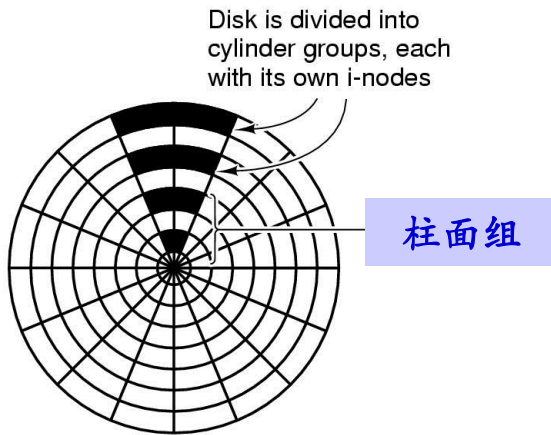
→ 尽量分配在同一柱面上，从而减少磁盘臂的移动次数和距离



例子：



(a)



(b)

各种提高文件系统性能的方法

文件系统的性能2

磁盘调度

当有多个访盘请求等待时，采用一定的策略，对这些请求的服务顺序调整安排

→ 降低平均磁盘服务时间，达到公平、高效

公平：一个I/O请求在有限时间内满足

高效：减少设备机械运动带来的时间开销

一次访盘时间 = 寻道时间 + 旋转延迟时间 + 传输时间

- 减少寻道时间
- 减少延迟时间

磁盘调度算法(1/9)

例子：假设磁盘访问序列：

98, 183, 37, 122, 14, 124, 65, 67

读写头起始位置：**53**

要求计算：

- 磁头服务序列
- 磁头移动总距离（道数）

磁盘调度算法(2/9)

- 先来先服务(FCFS)

按访问请求到达的先后次序服务

- 优点：简单，公平

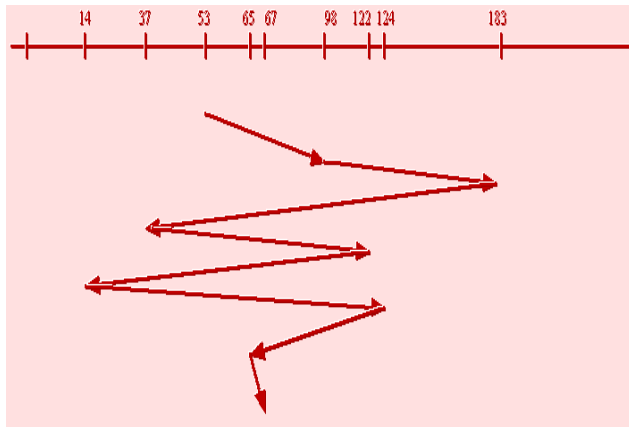
- 缺点：效率不高，相临两次请求可能会造成最内到最外的柱面寻道，使磁头反复移动，增加了服务时间，对机械也不利

磁盘访问序列：

98, 183, 37, 122, 14, 124,
65, 67

读写头起始位置：53

640 磁道 (平均 80)



磁盘调度算法(3/9)

最短寻道时间优先(Shortest Seek Time First)

优先选择距当前磁头最近的访问请求进行服务

主要考虑寻道优先

优点：改善了磁盘平均服务时间

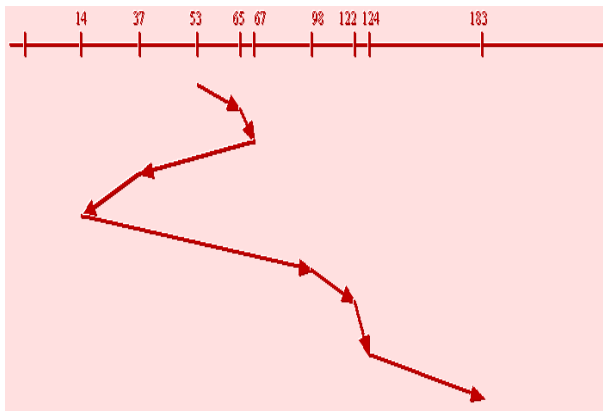
缺点：造成某些访问请求长期等待得不到服务

假设磁盘访问序列：

98, 183, 37, 122, 14, 124,
65, 67

读写头起始位置：53

236磁道 (平均 29.5)



磁盘调度算法(4/9)

折中权衡
距离、方向

◎ 扫描算法SCAN (电梯算法)

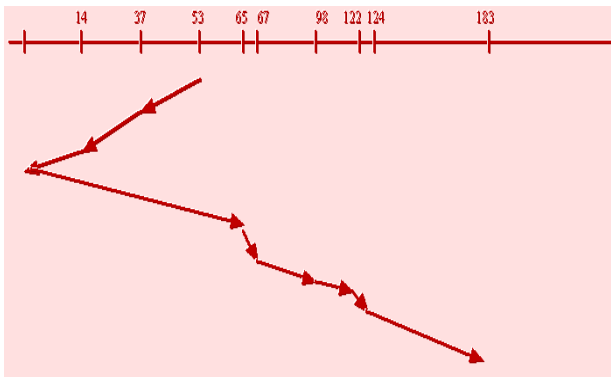
当设备无访问请求时，磁头不动；当有访问请求时，磁头按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务，如此反复

假设磁盘访问序列：

98, 183, 37, 122, 14, 124,
65, 67

读写头起始位置：53

218 磁道 (平均 27.25)



磁盘调度算法(5/9)

减少了新请求
的最大延迟

◎ 单向扫描调度算法C-SCAN

- 总是从0号柱面开始向里扫描
- 按柱面(磁道)位置选择访问者
- 移动臂到达最后一个柱面后，立即带动读写磁头快速返回到0号柱面
- 返回时不为任何的等待访问者服务
- 返回后可再次进行扫描

磁盘调度策略(6/9)

克服“磁头臂
的粘性”

◎ N-step-SCAN策略

- 把磁盘请求队列分成长度为N的子队列，每一次用SCAN处理一个子队列
- 在处理某一个队列时，新请求添加到其他子队列中
- 如果最后剩下的请求数小于N，则它们全都将在下一次扫描时处理
- N值比较大时，其性能接近SCAN；当 $N=1$ 时，即FIFO

磁盘调度策略(7/9)

克服“磁头臂
的粘性”

◎ FSCAN策略

- 使用两个子队列
- 扫描开始时，所有请求都在一个队列中，而另一个队列为空
- 扫描过程中，所有新到的请求都放入另一个队列中
- 对新请求的服务延迟到处理完所有老请求之后

磁盘调度算法(8/9)

◎ 旋转调度算法

旋转调度：根据延迟时间来决定执行次序的调度

三种情况：

- 若干等待访问者请求访问同一磁头上的不同扇区
- 若干等待访问者请求访问不同磁头上的不同编号的扇区
- 若干等待访问者请求访问不同磁头上具有相同的扇区

磁盘调度算法(9/9)

◎ 解决方案:

- 对于前两种情况: 总是让首先到达读写磁头位置下的扇区先进行传送操作
- 对于第三种情况: 这些扇区同时到达读写磁头位置下, 可任意选择一个读写磁头进行传送操作

例子:



请求顺序	柱面号	磁头号	扇区号
①	5	4	1
②	5	1	5
③	5	4	5
④	5	2	8

各种提高文件系统性能的方法

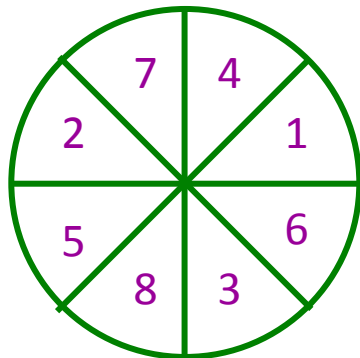
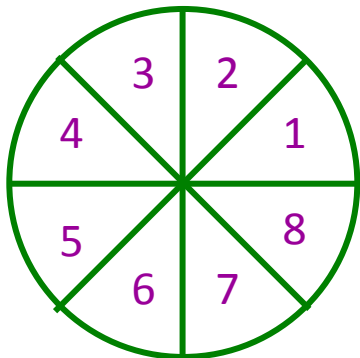
文件系统的性能3

信息的优化分布

记录在磁道上的排列方式也会影响输入输出操作的时间

例子：

处理程序要求顺序处理8个记录；磁盘旋转一周为20毫秒/周；花5毫秒对记录进行处理



记录的成组与分解

➤ 记录的成组

把若干个逻辑记录合成一组存放一块的工作

➤ 进行成组操作时必须使用内存缓冲区，缓冲区的长度等于逻辑记录长度乘以成组的块因子

➤ **成组目的：**提高了存储空间的利用率；减少了启动外设的次数，提高系统的工作效率

➤ 记录的分解

从一组逻辑记录中把一个逻辑记录分离出来



典型例子——
目录文件

RAID技术

美国加州伯克利分校
D.A.Patterson教授
1988年提出

RAID（独立磁盘冗余阵列）

(Redundant Arrays of Independent Disks)

多块磁盘按照一定要求构成一个独立的存储设备

目标：提高可靠性和性能

考虑：磁盘存储系统的 速度、容量、容错、数据
灾难发生后的数据恢复

RAID技术的思想

基本思路

数据是如何组织的？

- 通过把多个磁盘组织在一起，作为一个逻辑卷提供磁盘跨越功能
- 通过把数据分成多个数据块，**并行**写入/读出多个磁盘，以提高数据传输率（**数据分条stripe**）
- 通过镜像或校验操作，提供容错能力（**冗余**）

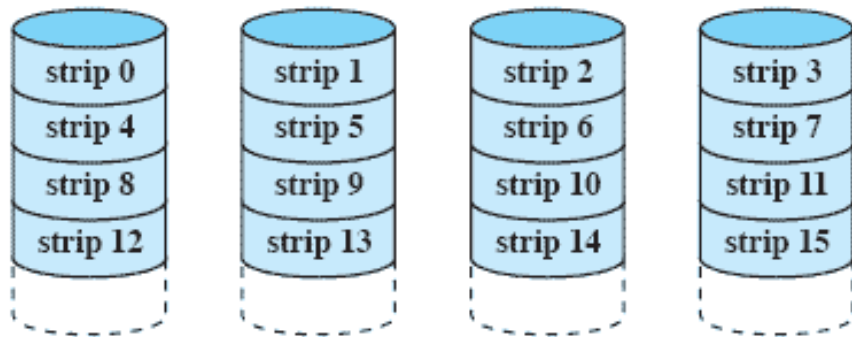
最简单的**RAID**组织方式：**镜像**

最复杂的**RAID**组织方式：**块交错校验**

RAID 0 - 条带化

无冗余(即
无差错控制)
性能最佳

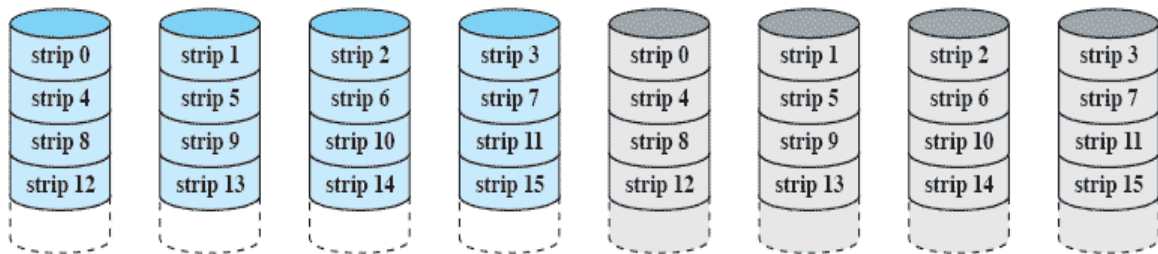
- 数据分布在阵列的所有磁盘上
- 有数据请求时，同时多个磁盘并行操作
- 充分利用总线带宽，数据吞吐率提高，驱动器负载均衡



RAID 1 - 镜像

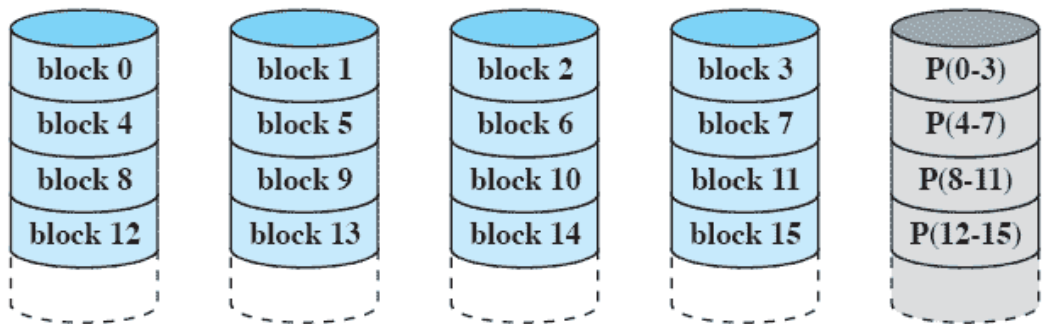
数据安全性
最好

- 最大限度保证数据安全及可恢复性
- 所有数据同时存在于两块磁盘的相同位置
- 磁盘利用率**50%**



RAID 4 交错块奇偶校验

- 带奇偶校验
- 以数据块为单位



文件系统的两个练习

练习1

有一个文件系统，根目录常驻内存，如图所示。

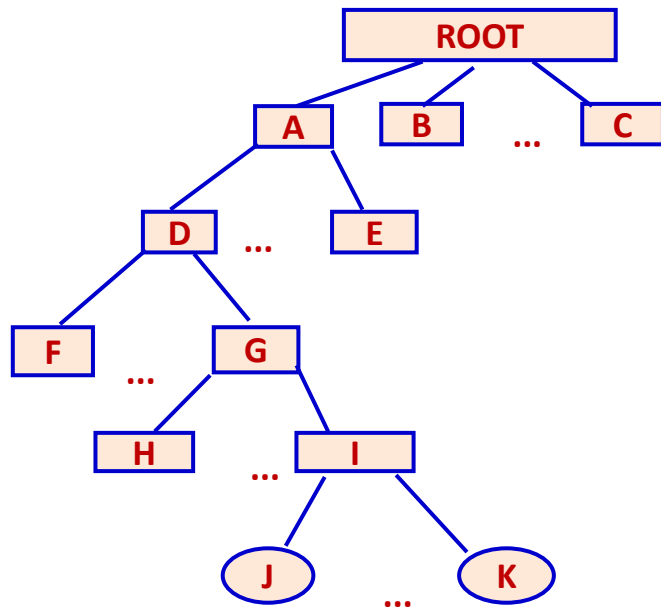
目录文件采用链接结构，规定一个目录下最多存放50个下级文件。下级文件可以是目录文件，也可以是普通文件。每个磁盘块可存放10个下级文件的目录项，若下级文件为目录文件，则目录项给出该目录文件的第一块地址，否则给出普通文件的FCB的地址。

假设文件按自左向右的顺序建立，...表示由若干内容未显示。

(1) 假设普通文件采用UNIX的三级索引结构，即FCB中给出13个磁盘地址，前10个磁盘地址指出文件前10块的物理地址，第11个磁盘地址指向一级索引表（给出256个磁盘地址）；第12个磁盘地址指向二级索引表，二级索引表中指出256个一级索引表的地址；第13个磁盘地址指向三级索引表，三级索引表中指出256个二级索引表的地址。若要读文件\A\D\G\I\K中的某一块，最少要启动磁盘几次？最多要启动磁盘几次？

(2) 若普通文件采用链接结构，要读\A\D\G\I\K的第55块，最少启动硬盘几次？最多几次？

(3) 若普通文件采用顺序结构，要读\A\D\G\I\K的第5555块，最少启动硬盘几次？最多几次？



练习2

- 假设磁盘分区大小为 **2M**；每块/簇 为**512**字节；要求画出 **UNIX** 和 **FAT16**文件系统布局
- \ mkdir A
- A mkdir B
- B create File1(4块/簇)
- \ mkdir C
- \ mkdir D
- C mkdir E
- E create File2 (16块/簇)
- E mkdir F
- F create File3 (8块/簇)
- F create File4 (2块/簇)

本讲重点

- ◎ 掌握文件操作的实现流程
- ◎ 掌握文件系统可靠性、一致性、写入策略、安全性的基本概念
- ◎ 掌握提高文件系统性能的各种方法
- ◎ 理解FAT文件系统的实现

本周要求

- 重点阅读教材

第4章相关内容：4.4、4.5

- 重点概念

文件操作及实现 文件系统的可靠性 文件系统的一致性
文件写入策略 文件的保护机制
文件系统的性能：目录项分解法、当前目录、磁盘高速缓存、提前预取、磁盘调度、RAID技术、合理分配磁盘空间、记录的成组与分解、信息的优化分布

THANKS

The End