

操作系统A

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

2020 Autumn

本章要求掌握的概念

文件系统

文件

文件分类

文件控制块FCB

文件目录

目录文件

文件系统布局

文件逻辑结构

文件物理结构

文件描述符/文件
句柄

FAT/UNIX

文件基本操作

内存结构

文件共享

磁盘空间管理

文件管理

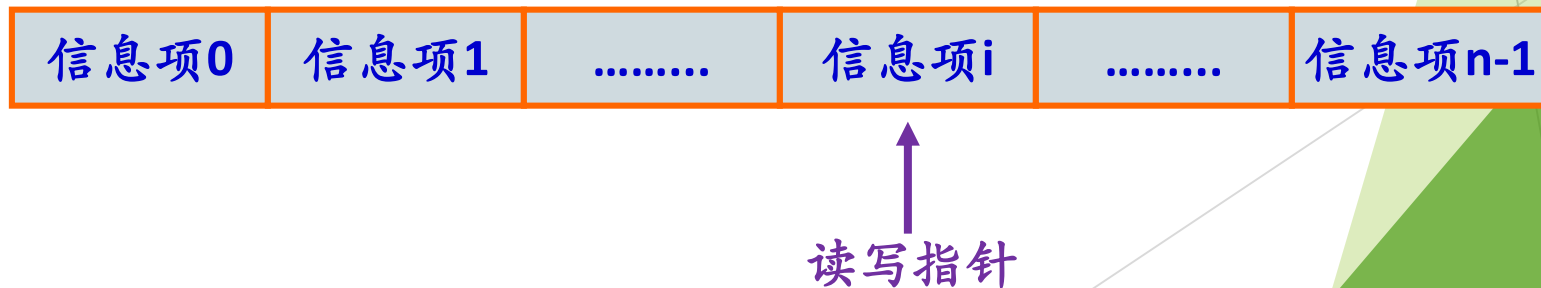
- 文件和文件目录
- 文件系统的实现
- 文件系统实例 (FAT、UNIX)
- 文件系统的管理
- 文件系统的性能
- 文件系统结构
- Windows文件系统NTFS

文件、文件系统、文件目录、目录文件.....

文件系统的基本概念

文件是什么？

- ▶ 文件 是 对磁盘的 抽象
- ▶ 所谓文件 是指 一组带标识（标识即为文件名）的、在逻辑上有完整意义的信息项的序列
- ▶ 信息项：构成文件内容的基本单位（单个字节，或多个字节），各信息项之间具有顺序关系
- ▶ 文件内容的意义：由文件建立者和使用者解释



文件系统

管理持久
性数据

- ▶ 操作系统中统一管理信息资源的子系统，管理文件的存储、检索、更新，提供安全可靠的共享和保护手段，并且方便用户使用
- ▶ 统一管理磁盘空间，实施磁盘空间的分配与回收
- ▶ 实现文件的按名存取
 - 名字空间 $\xrightarrow{\text{映射}}$ 磁盘空间
- ▶ 实现文件信息的共享，并提供数据可靠性和安全保障
- ▶ 向用户提供一个方便使用、维护的接口，并向用户提供有关信息
- ▶ 提高文件系统的性能
- ▶ 提供与I/O系统的统一接口

文件的分类

按文件性质和用途分类 (UNIX)

普通文件；目录文件；特殊文件(设备文件)；管道文件；套接字；符号链接文件

普通文件(regular)

包含了用户的信息，一般为ASCII或二进制文件

目录文件(directory)

管理文件系统的系统文件

特殊文件(special file)

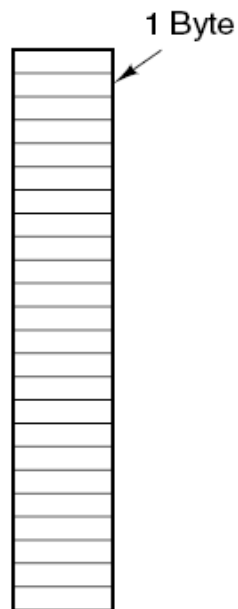
字符设备文件：和输入输出有关，用于模仿串行I/O设备，例如终端，打印机，网络等

块设备文件：模仿磁盘

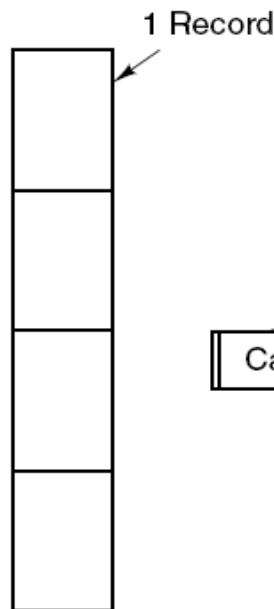
文件的逻辑结构(1/2)

文件内部结构：从用户角度看文件，由用户的访问方式确定

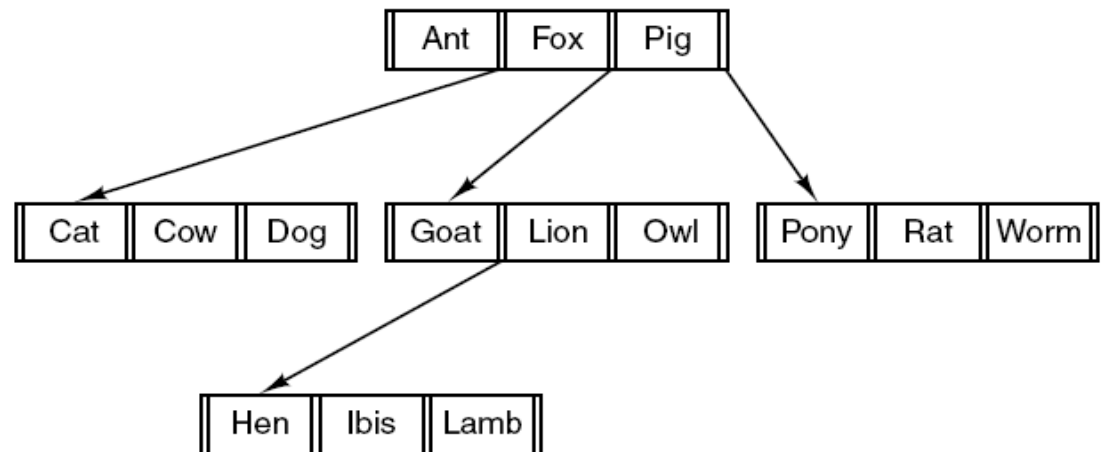
- 流式文件、记录式文件



字节序列



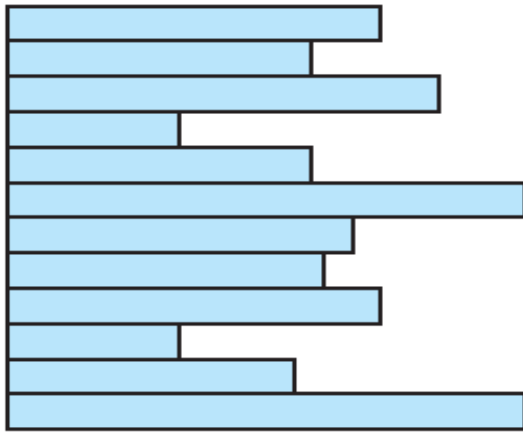
记录序列



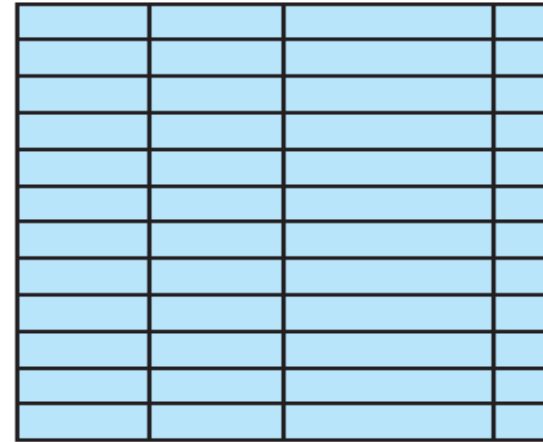
树

文件的逻辑结构(2/2)

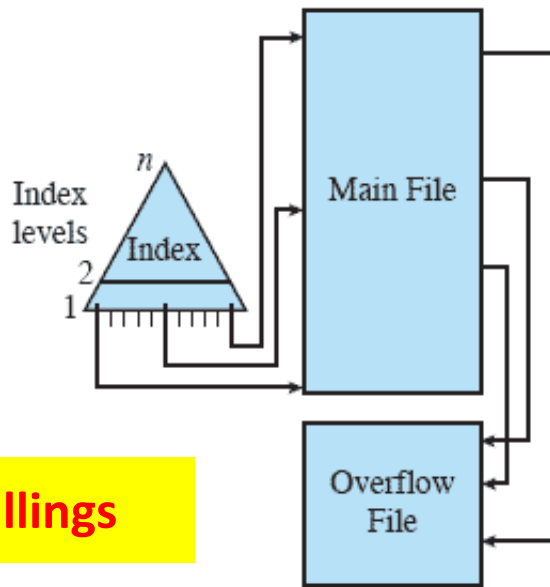
散列



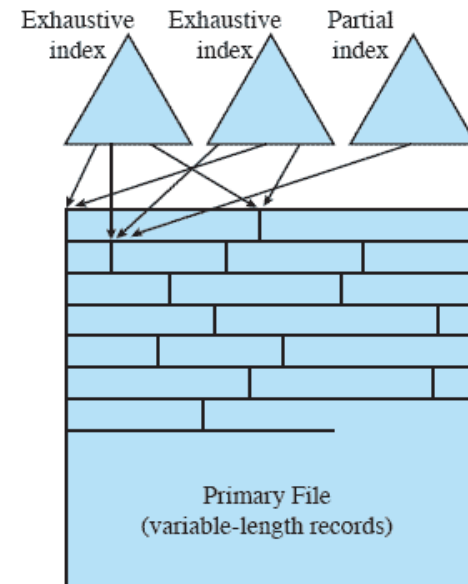
堆



顺序



索引
顺序



索引

文件的存取方式

- ◆ 顺序存取(访问)——按字节依次读取
- ◆ 随机存取(访问)——从任意位置读写
提供读写位置(当前位置)
例如：UNIX的seek操作

存储介质与物理块

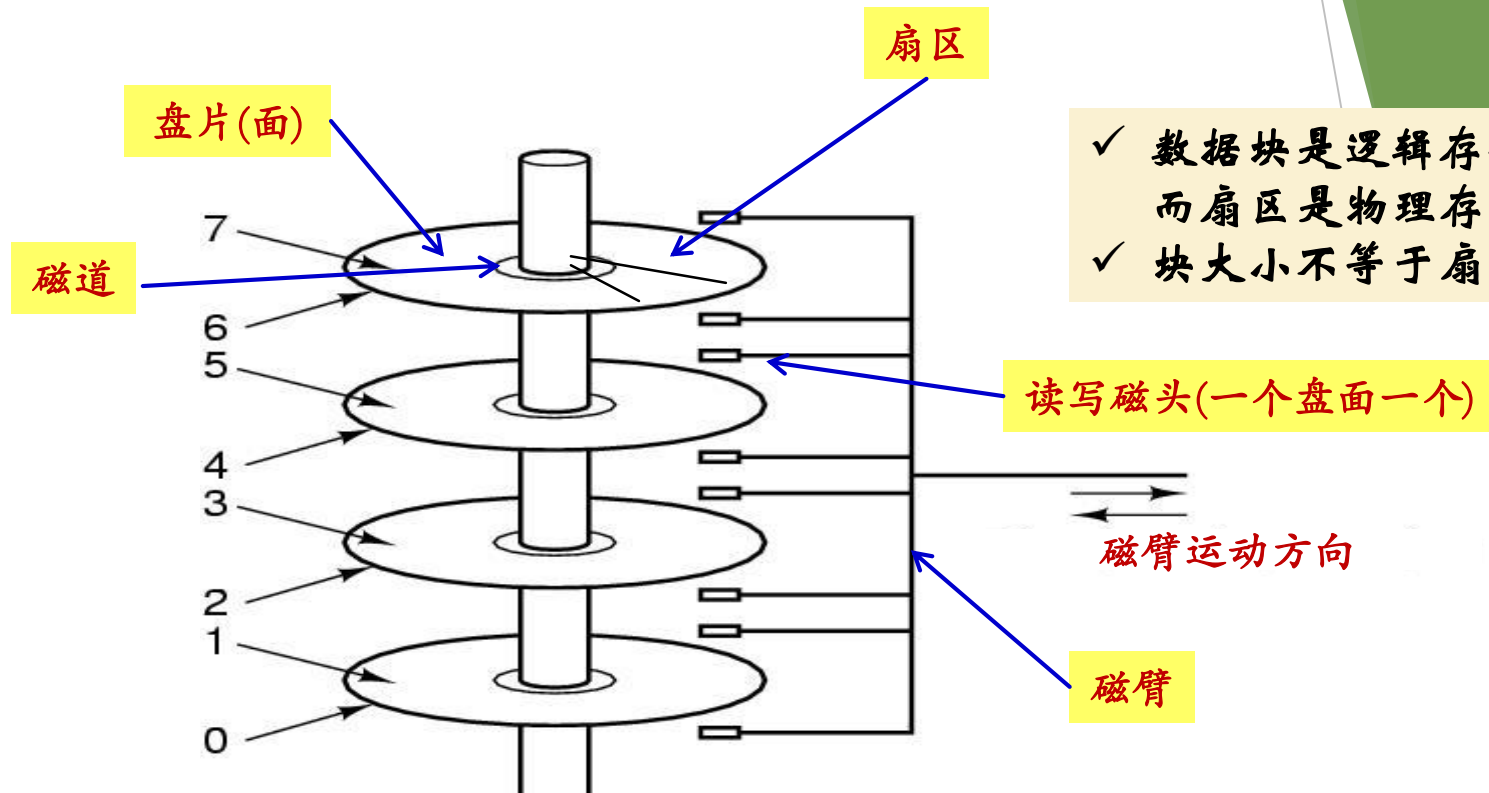
- 典型的存储介质

磁盘、**SSD**盘、磁带、光盘、U盘、.....

- 物理块（块、数据块）

- 将存储设备划分为大小相等的块，统一编号
- 块是信息存储、传输、分配的基本单位

典型的磁盘结构



任何时刻只有一个磁头处于活动状态：输入输出数据流以位串形式出现
物理地址形式：

磁头号（盘面号）、磁道号（柱面号）、扇区号

扇区：标题(10字节)、数据(512字节)、ECC纠错信息(12-16字节)

磁盘访问

一次访盘请求:

读/写，磁盘地址（设备号，柱面号，磁头号，扇区号），内存地址（源/目）

完成过程由三个动作组成:

- ▶ 寻道（时间）：磁头移动定位到指定磁道
- ▶ 旋转延迟(时间): 等待指定扇区从磁头下旋转经过
- ▶ 数据传输（时间）：数据在磁盘与内存之间的实际传输

文件属性

► 文件控制块 (File Control Block)

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

(文件属性或元数据)

► 常用属性

文件名，文件号，保护，口令，创建者，当前拥有者，文件地址，文件大小，文件类型，共享计数，创建时间，最后修改时间，最后访问时间，各种标志(只读、隐藏、系统、归档、ASCII/二进制、顺序/随机访问、临时文件、锁)

文件操作

- create
- delete
- open
- close
- read
- write
- append
- seek
- get attributes
- set attributes
- rename

文件访问模式：进程访问文件数据前必须先“打开”文件

```
f = open(name, flag);
```

```
...
```

```
read(f, ...);
```

```
...
```

```
close(f);
```

文件目录、目录项与目录文件

▶ 文件目录

- ▶ 统一管理每个文件的信息
- ▶ 将所有文件的管理信息组织在一起，即构成文件目录

文件目录是文件控制块的有序集合

▶ 目录文件

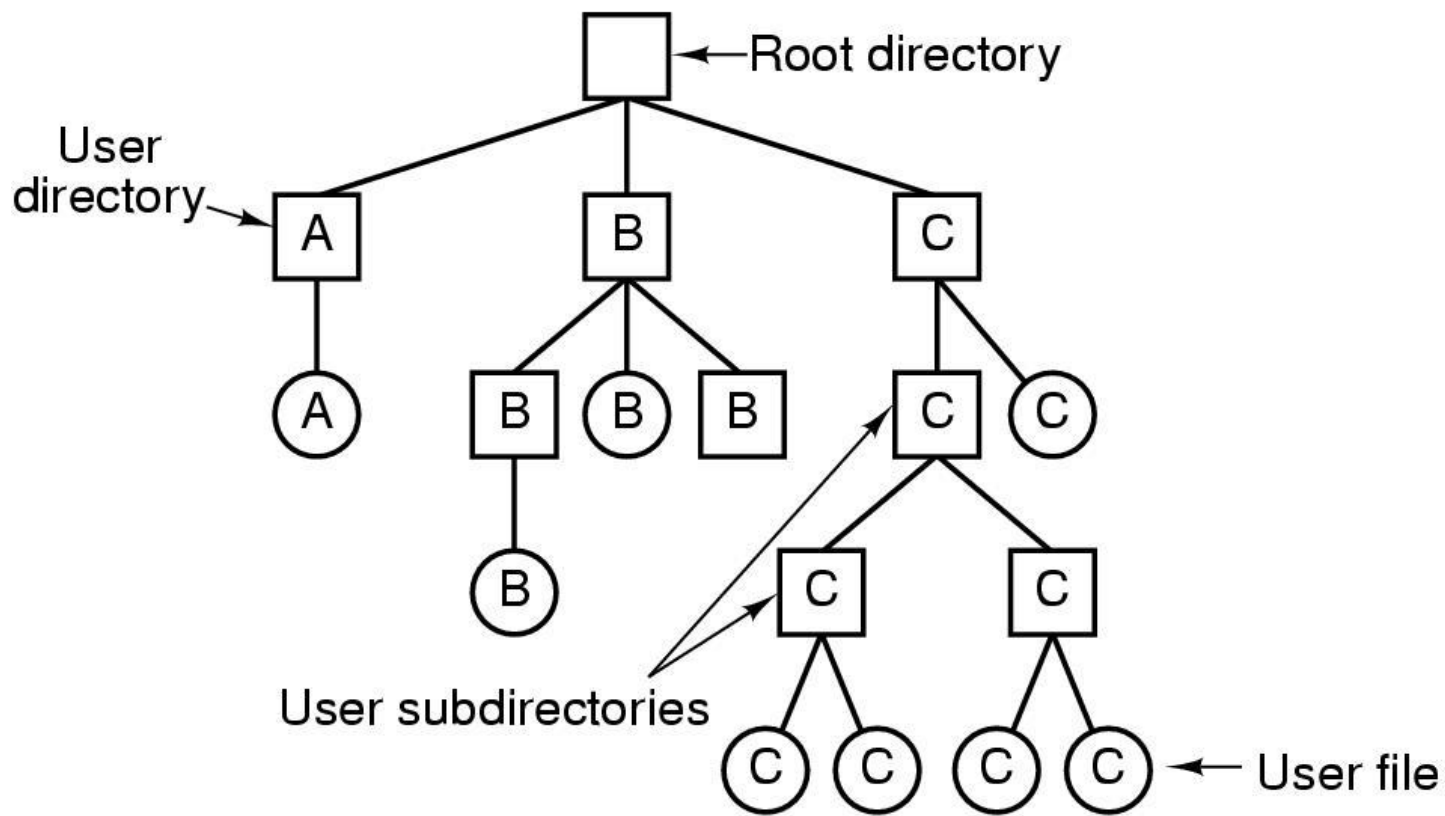
- ▶ 将文件目录以文件的形式存放在磁盘上
- ▶ 一种特殊类型的文件，其内容由目录项组成

▶ 目录项

- ▶ 构成文件目录的基本单元
- ▶ 目录项 可以是FCB

- ✓ 为确保映射的完整性，只允许内核修改目录
- ✓ 应用程序通过系统调用访问目录

树形目录结构(1/2)



树形目录结构(2/2)

▶ 路径名（文件名）

- 绝对路径名：从根目录开始
- 相对路径名：从当前目录开始

▶ 当前目录/工作目录

- 每个进程一个，可以改变，用于解析文件名，与许用户使用相对路径名代替绝对路径名

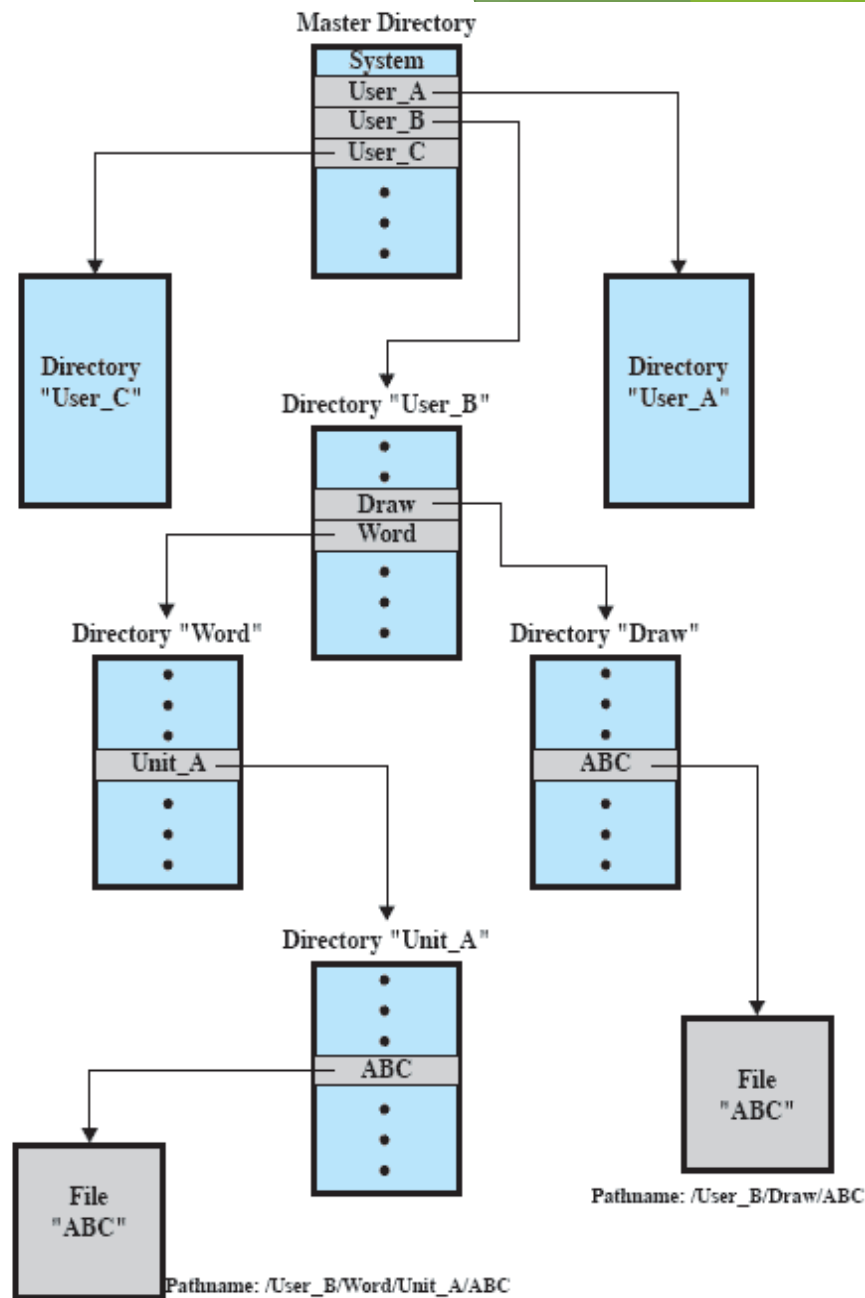
▶ 目录操作

- create、delete
- opendir、closedir
- readdir、rename
- link、unlink

典型目录操作：
创建目录文件、删除目录文件、
列目录、遍历路径等

目录文件示例

- ▶ 试画出右图中对应的文件目录的树形结构
- ▶ 右图中有多少个目录文件?
- ▶ 试给出解析文件名 /User_B/Draw/ABC 的步骤



文件系统在磁盘上的布局、内存结构、.....

文件系统的实现(1)

概述

► 实现文件系统需要考虑

磁盘 与 内存 中的内容布局

► 磁盘上

- ✓ 如何启动所存储的操作系统?
- ✓ 磁盘是怎样管理的? 即怎样获取磁盘的有关信息?
- ✓ 目录文件在磁盘上怎么存放? 普通文件在磁盘上怎么存放?

► 内存中

进程使用文件时, 操作系统如何支持与管理?

相关术语

- ◎ **磁盘分区 (partition)** : 把一个物理磁盘的存储空间划分为几个相互独立的部分, 称为**分区**
- ◎ **文件卷 (volume)** : 逻辑分区, 由一个或多个物理块(簇) 组成
 - 一个文件卷可以是整个磁盘 或 部分磁盘 或 跨盘 (RAID)
 - 同一个文件卷中使用同一份管理数据进行文件分配和磁盘空闲空间管理, 不同的文件卷中的管理数据是相互独立的
 - 一个文件卷上: 包括文件系统信息、一组文件 (用户文件、目录文件)、未分配空间
 - **物理块/块 (Block)** 或 **簇 (Cluster)** : 一个或多个 (2的幂) 连续的扇区, 可寻址数据块
- ◎ **格式化 (format)** : 在一个文件卷上建立文件系统的过程, 即建立并初始化用于文件分配和磁盘空闲空间管理的**管理数据**

元数据

磁盘上的内容

► 引导区

包括从该卷引导操作系统所需要的信息

每个卷（分区）一个，通常为第一个扇区

► 卷（分区）信息

包括该卷（分区）的块（簇）数、块（簇）大小、空闲块（簇）数量和指针、空闲FCB数量和指针

► 目录结构（目录文件）

► 用户文件

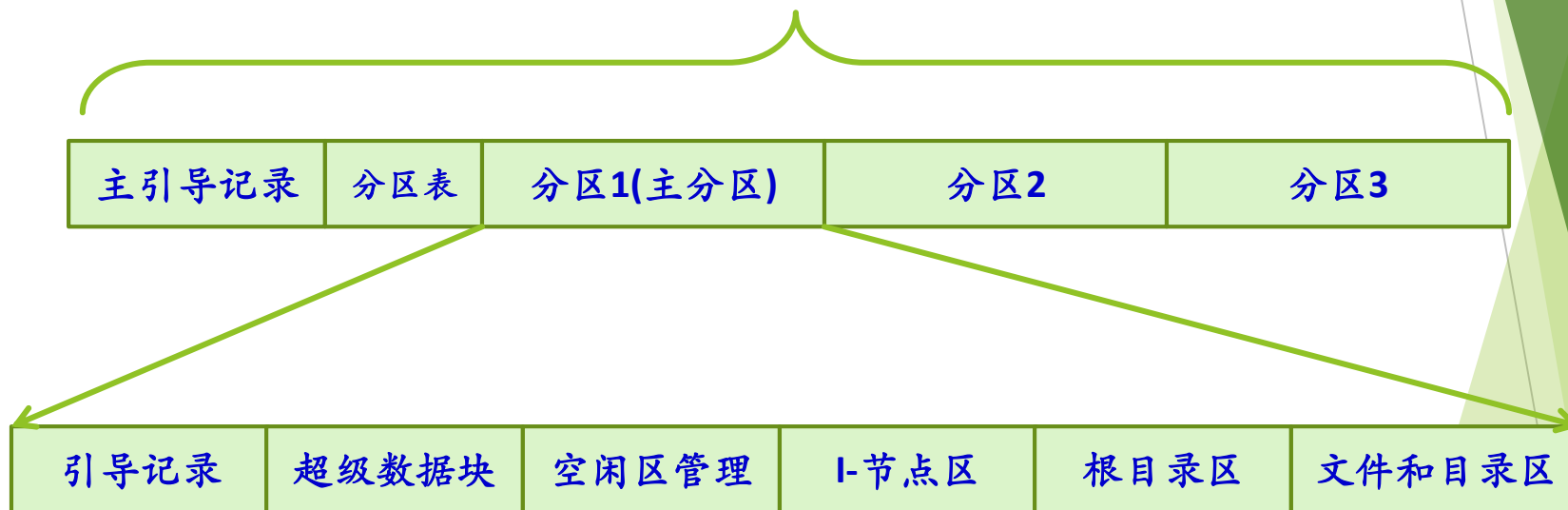
物理块（块block、簇cluster）

- 信息存储、传输、分配的独立单位
- 存储设备划分为大小相等的物理块，统一编号

1.磁 盘 上 文 件 系 统 的 布 局(1/2)

UNIX文件系统布局

整块磁盘



某一分区

磁盘上文件系统的布局(2/2)

FAT文件系统布局



物理结构



一个NTFS卷的布局

2. 文件的物理结构 (1/13)

文件在物理介质上的存放方式，也是系统分配给文件的物理块的位置和顺序

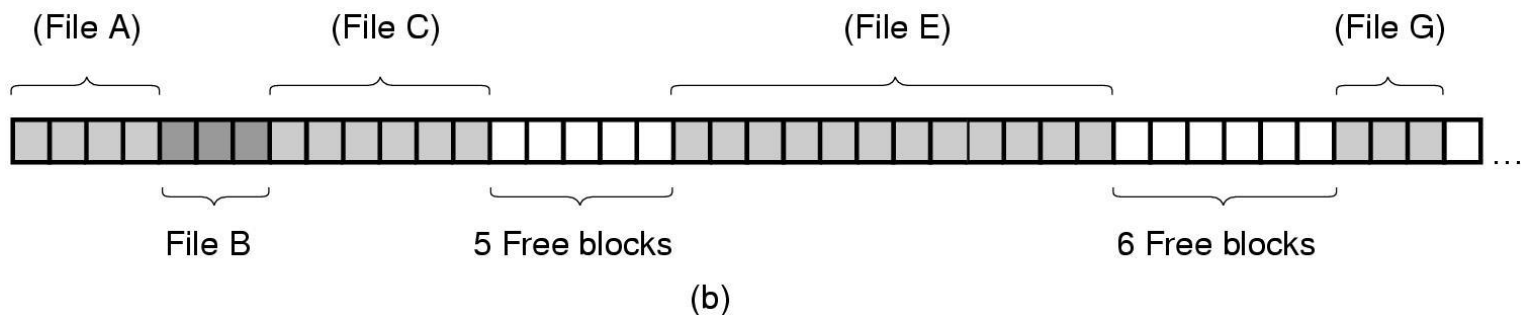
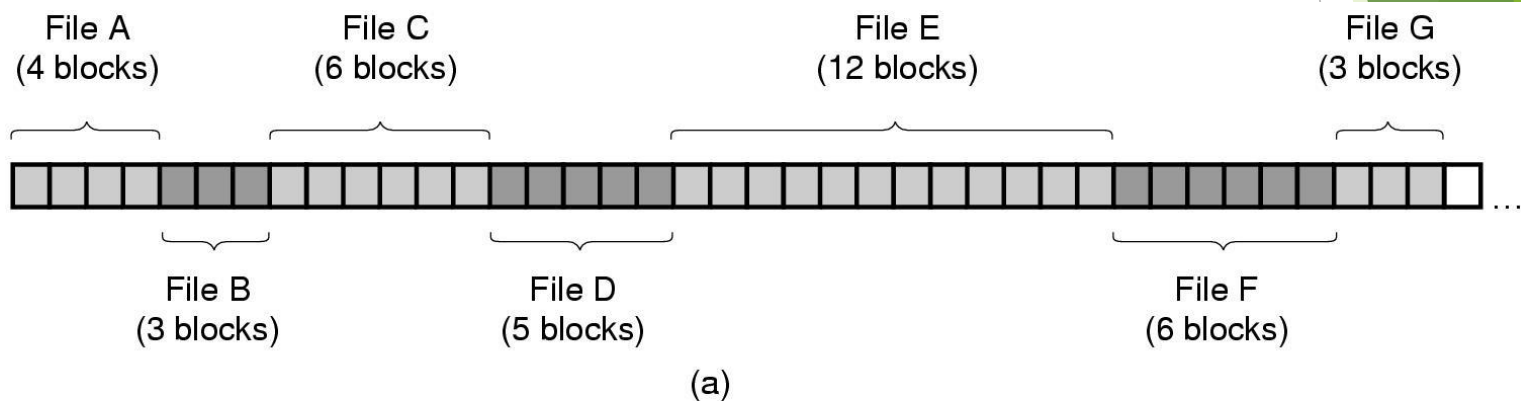
要考虑的问题：

- ▶ 存储效率：外部碎片等
- ▶ 读写性能：访问速度

文件的物理结构(2/13)

(1) 连续结构 (顺序)

文件的信息存放在若干连续的物理块中



文件的物理结构(3/13)

在FCB中
如何记录
磁盘地址?

► 优点

- 简单、高效
- 支持顺序存取和随机存取
- 所需的磁盘寻道次数和寻道时间最少
- 可以同时读入多个块，检索一个块也很容易

► 缺点

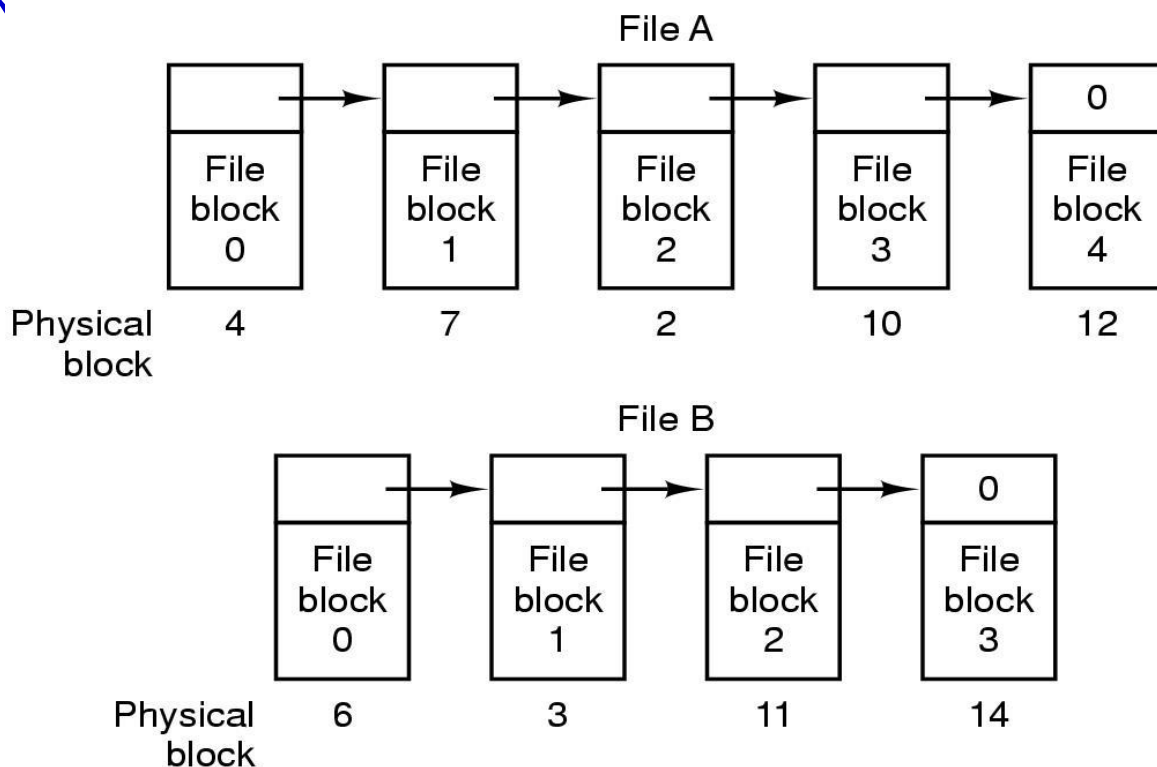
- 文件不能动态增长
预留空间：浪费 或 重新分配和移动
- 不利于文件内容的插入和删除
- 外部碎片问题，存储紧缩技术

文件的物理结构(4/13)

(2) 链接结构

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

在FCB中如何记录磁盘地址？



文件的物理结构(5/13)

► 优点

- 提高了磁盘空间利用率，不存在外部碎片问题
- 有利于文件内容的插入和删除
- 有利于文件动态扩充

► 缺点

- 存取速度慢，不适于随机存取
- 可靠性问题，如链接指针出错
- 更多的寻道次数和寻道时间
- 链接指针占用一定的空间

链接结构的一个变形：
文件分配表FAT

文件的物理结构(6/13)

文件分配表	
Physical block	
0	
1	
2	10
3	11
4	7
5	
6	3
7	2
8	
9	
10	12
11	14
12	-1
13	
14	-1
15	

表项的值有三种：
0, 下一块块号, -1

File A starts here

File B starts here

Unused block

某文件的起始块号从何处得到?

文件的物理结构(7/13)

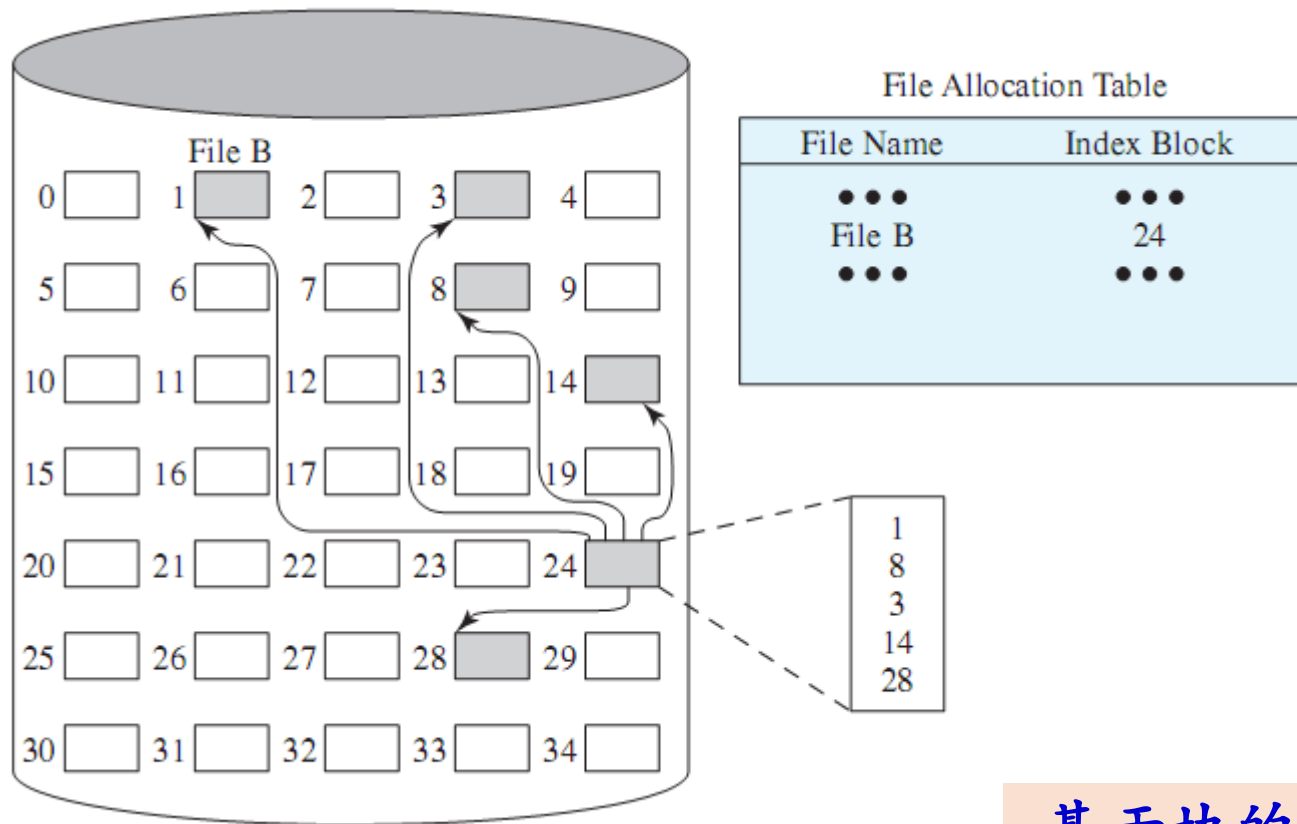
(3) 索引结构

- ✓ 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个专用数据结构——索引表，并将这些块的块号存放在一个索引表中
- ✓ 一个索引表就是磁盘块地址数组，其中第 i 个条目指向文件的第 i 块

在FCB中
如何记录
磁盘地址？

索引表
存放在
何处？

文件的物理结构(8/13)



基于块的索引分配

文件的物理结构(9/13)

► 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
- 能充分利用磁盘空间

► 缺点

- 较多的寻道次数和寻道时间
- 索引表本身带来了系统开销
如：内存、磁盘空间，存取时间

文件的物理结构(10/13)

问题：索引表很大，需要多个物理块存放时怎么办？

索引表组织：

- ▶ 链接方式

一个盘块存一个索引表，多个索引表链接起来

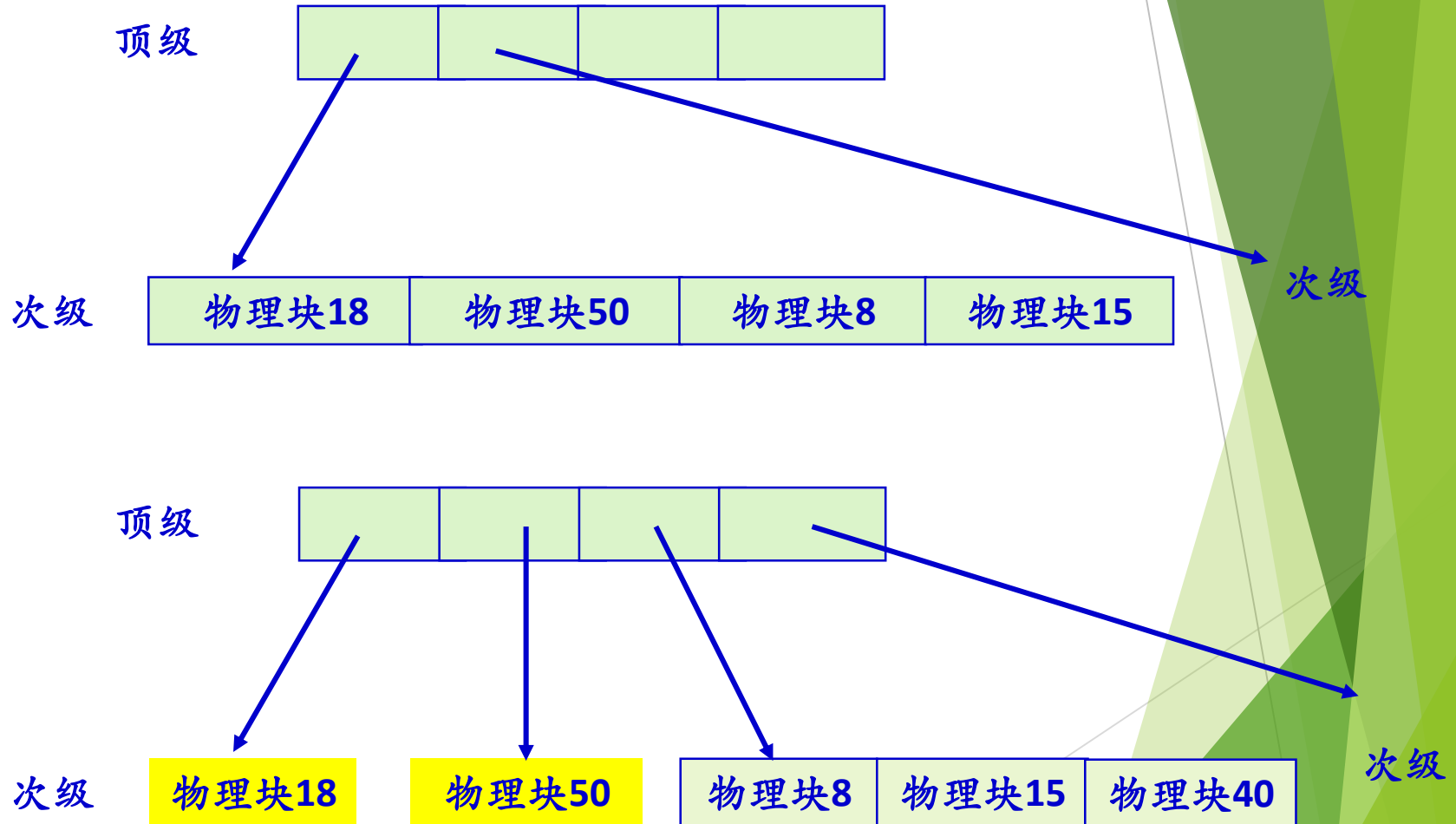
- ▶ 多级索引方式

将文件的索引表（二级索引）的地址放在另一个索引表（一级索引）中

- ▶ 综合模式

直接索引方式 与 间接索引方式 结合

文件的物理结构(11/13)



文件的物理结构(12/13)

UNIX文件系统采用的是多级索引结构(综合模式)

- ▶ 每个文件的索引表有15个索引项，每项2个字节
- ▶ 最前面12项直接登记存放文件信息的物理块号（直接寻址）
- ▶ 如果文件大于12块，则利用第13项指向一个物理块，该块中最多可放256个文件物理块的块号（一级索引表）
- ▶ 对于更大的文件还可利用第14和第15项作为二级和三级索引表

UNIX中采用了三级索引结构后，文件最大可达到？个物理块

假设扇区大小为512字节，物理块等于扇区块大小，一级索引表可以存放256个物理块号

主索引表

直接
盘块

间接
盘块

1节点



数据块

数据块

数据块

一级索引表

数据块

数据块

二级索引表

一级索引表

数据块

三级索引表

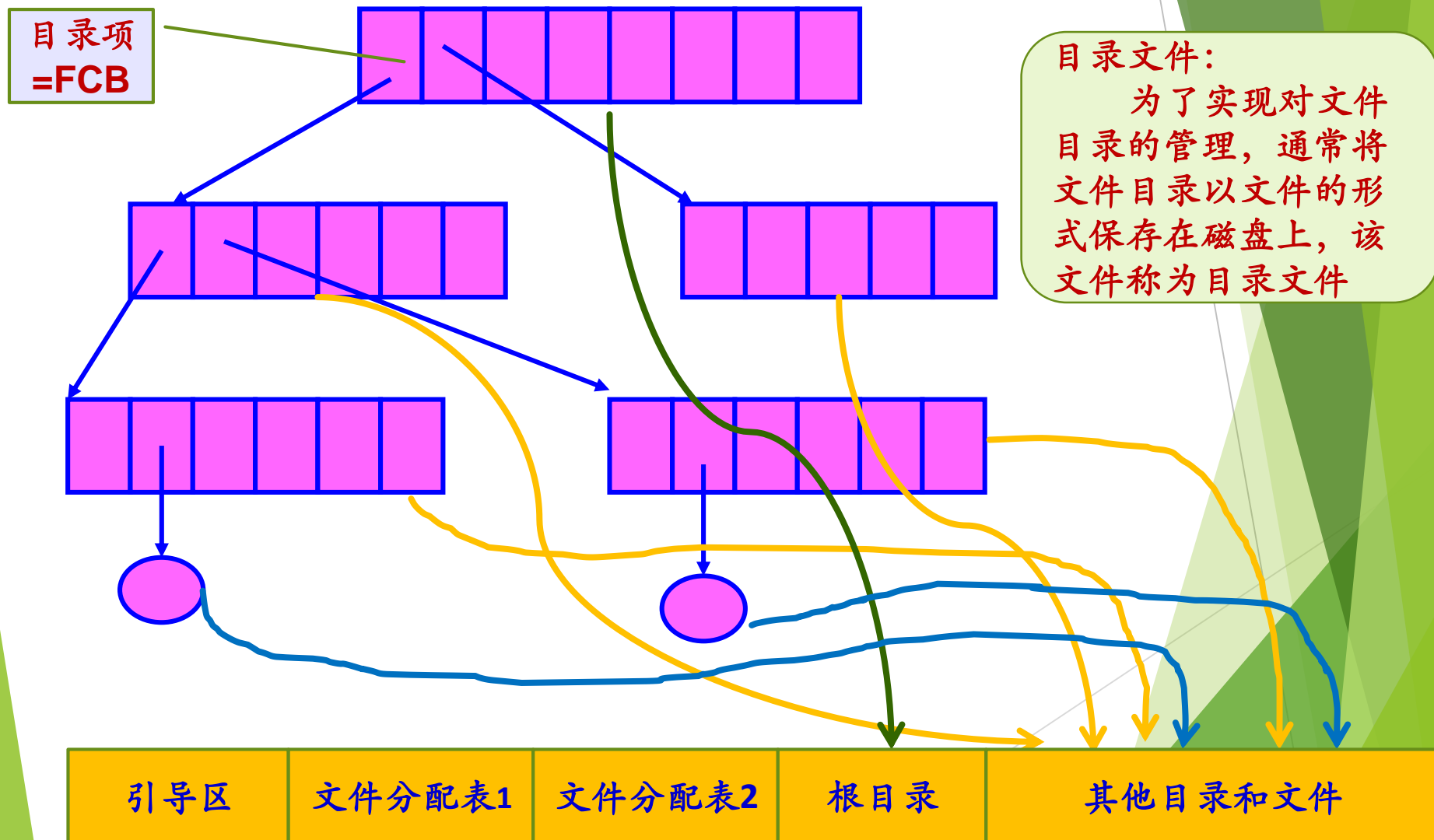
二级索引表

一级索引表

数据块

文件的物理
结构 (13/13)

3. 目录文件的组织方式(1/2)



目录文件的组织方式(2/2)

目录文件中目录项的组织方式:

- ▶ 顺序表

- ▶ 散列表

将目录项组织成一散列表

根据文件名计算散列值, 得到一个指向表中文件的指针(冲突值用链表组织)

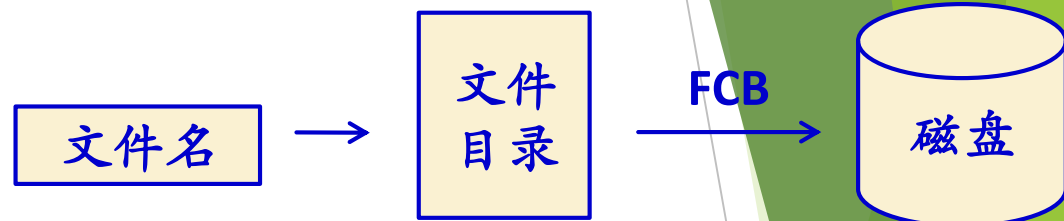
优点: 查找速度快

- ▶ B树 或 B+树

NTFS文件系统中, 目录文件采用B+树组织目录项

4. 文件目录检索

访问一个文件 → 两步骤



➤ 目录检索

用户给出文件名 → 按文件名查找目录项/FCB

文件名解析: 把逻辑名字转换成物理资源
路径名

全路径名: 从根开始 **\A\B\C\File1**

相对路径: 从当前目录开始 **C\File1**

➤ 文件寻址

根据**目录项/FCB**中文件物理地址等信息, 计算文件中任意记录或字符在存取介质上的地址

5. 目录文件的改进(1/3)

► 提问：如何加快目录检索？

► 一种解决方案：

目录项分解法：即把FCB分成两部分

► 符号目录项

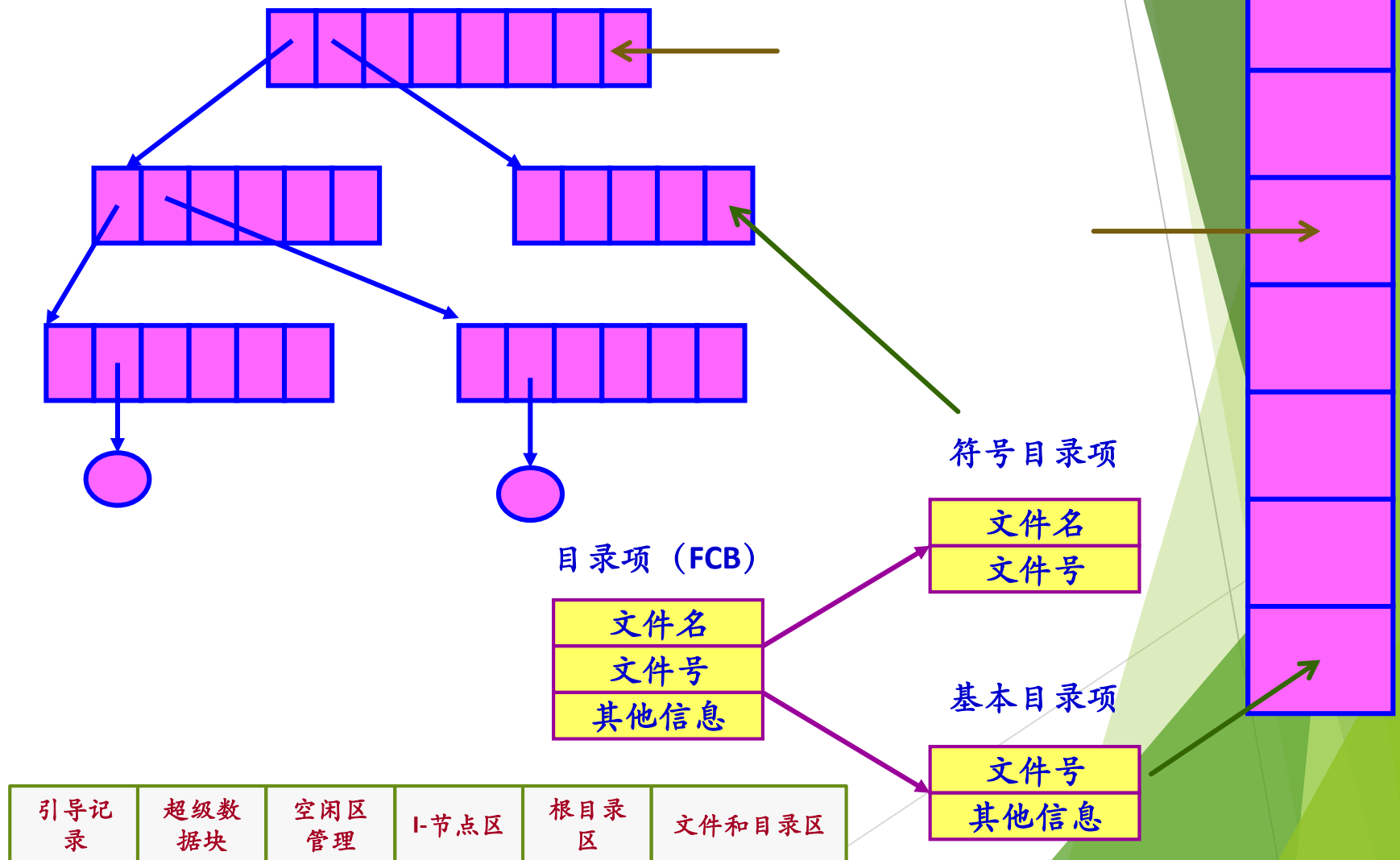
文件名，文件号

► 基本目录项

除**文件名**外的所有字段

例子：UNIX的I节点（索引节点 或 inode）

目录文件的改进(2/3)



目录文件的改进(3/3)

分解前：占13块
分解后：符号文件占2块
基本文件占11块

例子：

假设一个FCB有 48 个字节

物理块大小 512 字节

符号目录项占 8 字节（文件名6字节，文件号2字节）

基本目录项占 $48 - 6 = 42$ 字节

假设一个目录文件有128个目录项，计算查找一个文件的平均访盘次数？

分解前：7次

分解后：2.5次

文件目录改进后减少了访问硬盘的次数，提高了检索速度

UNIX、FAT16、FAT32.....

文件系统实例

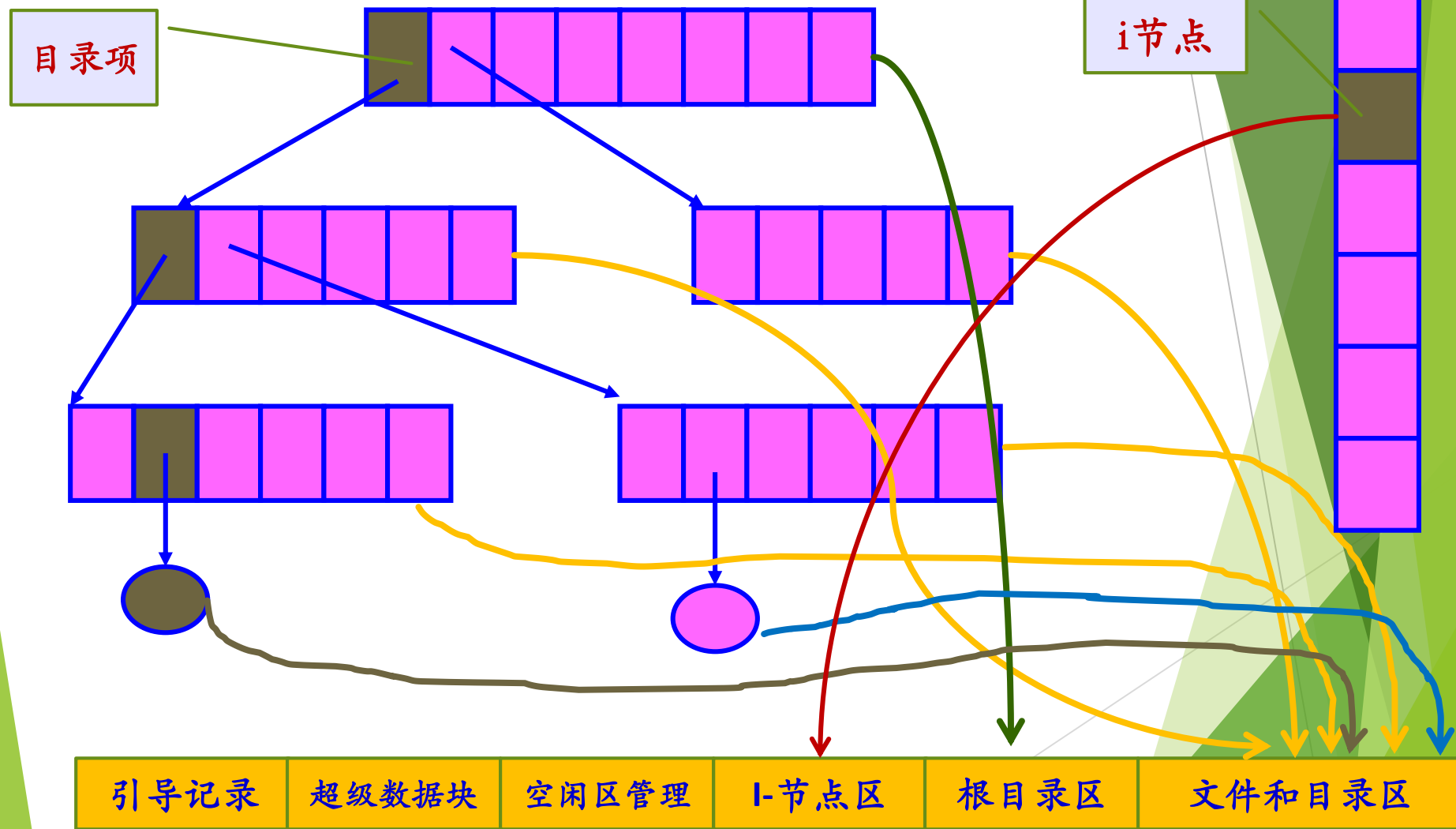
UNIX文件系统(1/3)

- ▶ FCB = 目录项 + i节点
- ▶ 目录项：文件名 + i节点号
- ▶ 目录文件由目录项构成
- ▶ i节点：描述文件的相关信息
- ▶ 每个文件由 一个目录项、一个i节点和若干磁盘块构成



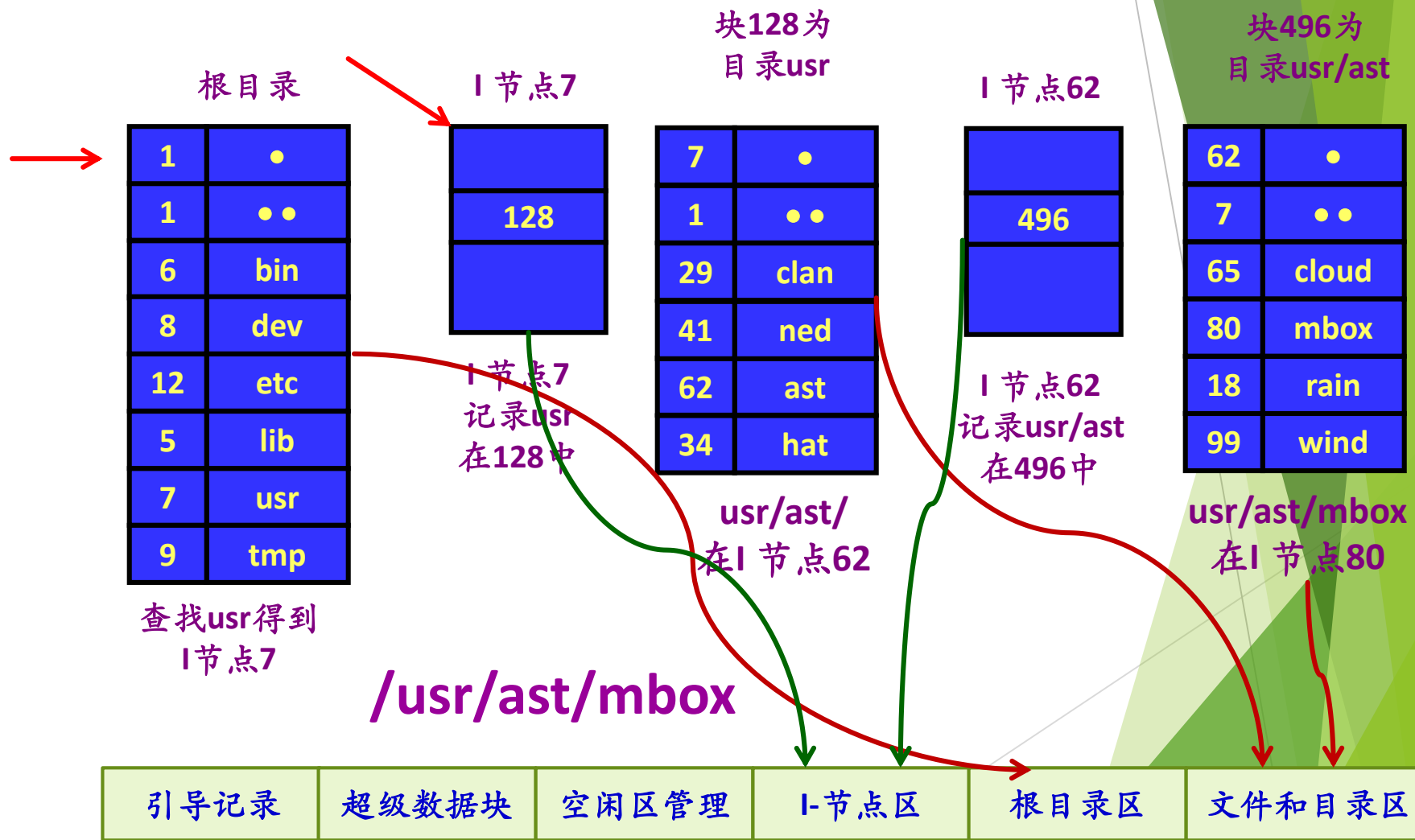
引导记录	超级数据块	空闲区管理	i-节点区	根目录区	文件和目录区
------	-------	-------	-------	------	--------

UNIX文件系统(2/3)



UNIX文件系统(3/3)

UNIX的I节点
参考教材4.5.3



Windows — FAT16文件系统

- ▶ **簇**：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
- ▶ **FAT表项**：占2字节
- ▶ **目录项**：占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 文件	其他目录文件和 普通文件
-----	------------	------------	-----------	-----------------

FA7文件系统——MBR

► 主引导记录扇区 — 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)

MBR

分区表

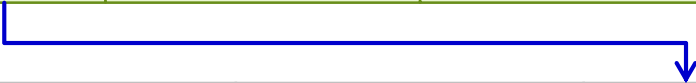
分区1(主分区)

分区2

分区3

FAT文件系统——DBR

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

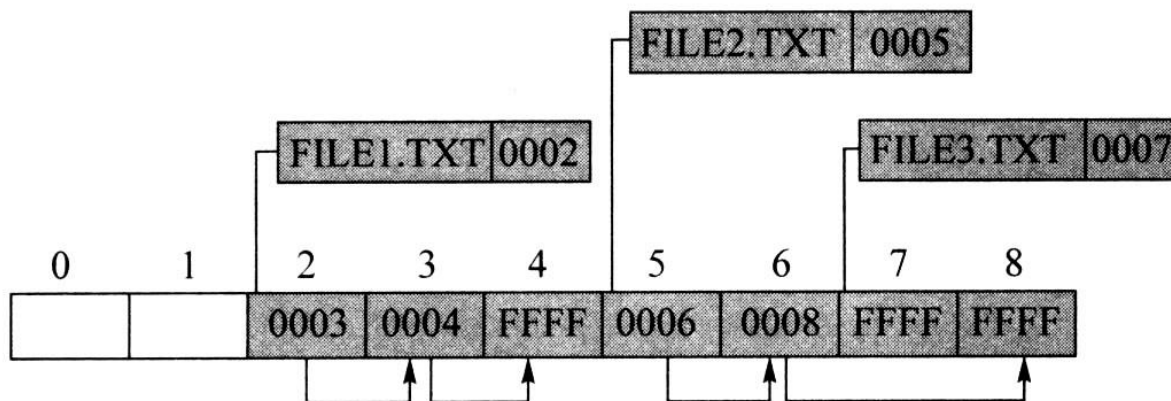
引导扇区 - 扩展BIOS参数块(EBPB)

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

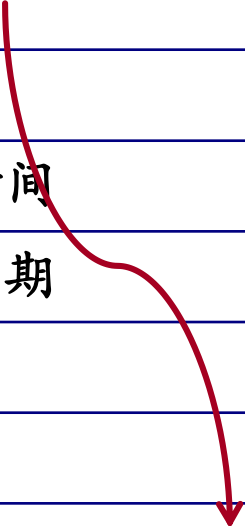
文件分配表 FAT

- ▶ 可以看成是一个整数数组，每个整数代表数据区的一个簇号
- ▶ 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



FA716目录项

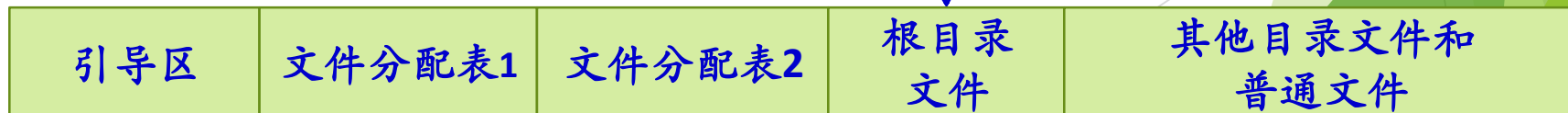
偏移	域长	含义
00h	8	文件名
08h	3	文件扩展名
0Bh	1	文件属性字节
0Ch	10	保留
16h	2	最后一次修改的时间
18h	2	最后一次修改的日期
1Ah	2	起始簇号
1Ch	4	文件大小



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、卷标项（根目录）、已删除目录项（第一字节为0xE5）、长文件名目录项等）
- ▶ 支持长文件名格式
- ▶ 支持Unicode
- ▶ 无法支持高级容错特性，不具有内部安全特性

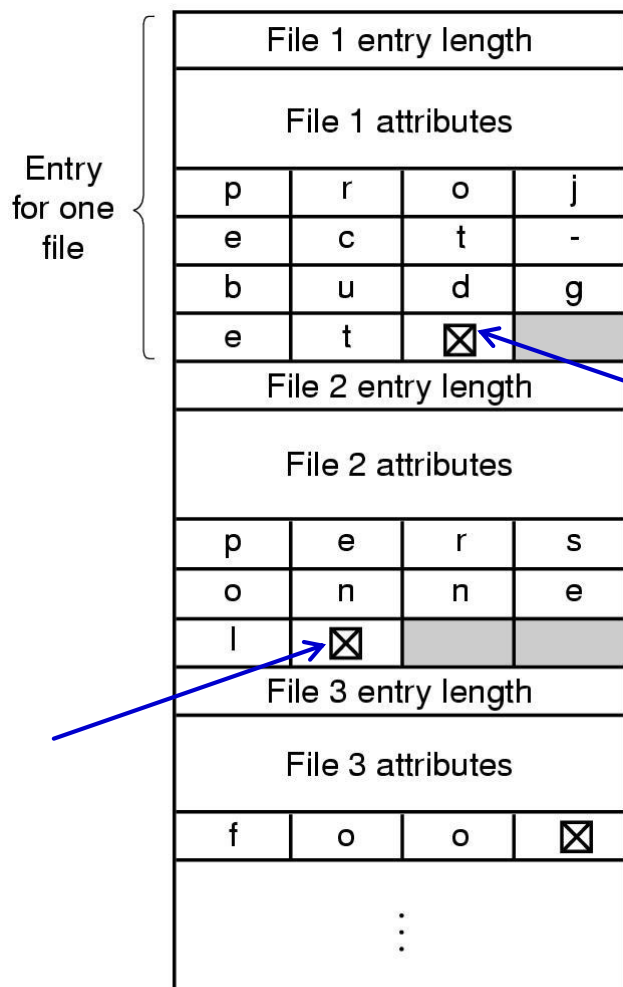


7A732目录项

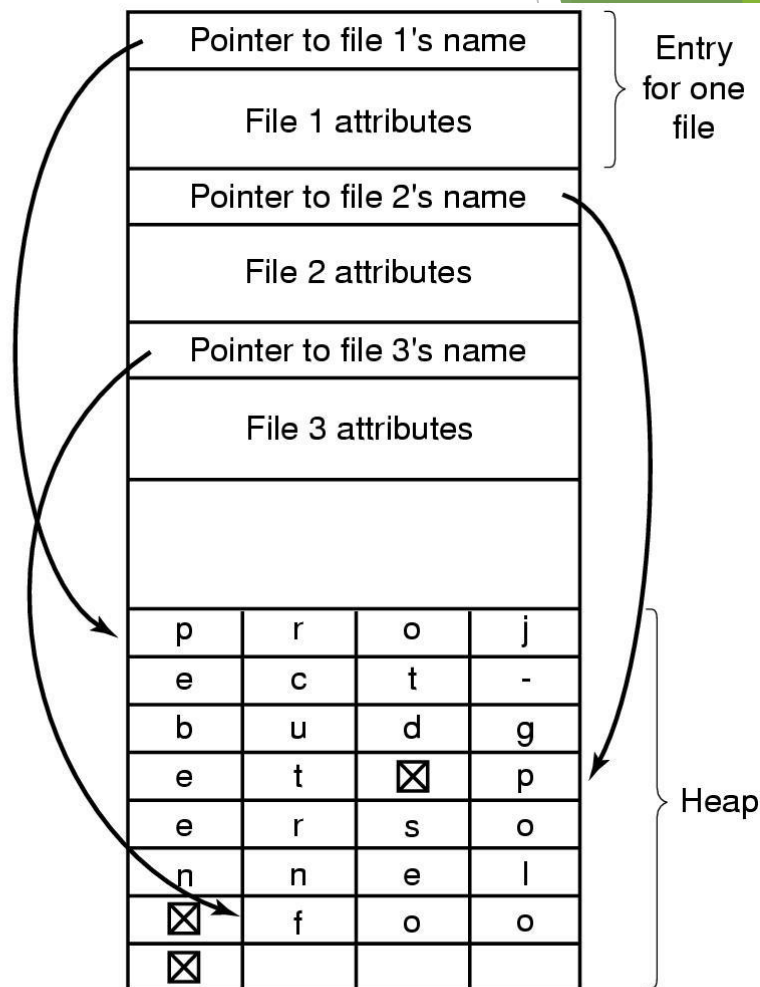
7	6	5	4	3	2	1	0
		归档	目录	卷标	系统	隐藏	只读

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



方案1



方案2

FA732-长文件名目录项格式

方案3

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)

0x42	w	n	.	f	o	0x0F	0x00	check sum	x
0x000000FFFF0xFFFF0xFFFF0xFFFF0xFFFF						0x0000	0xFFFF0xFFFF		
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

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

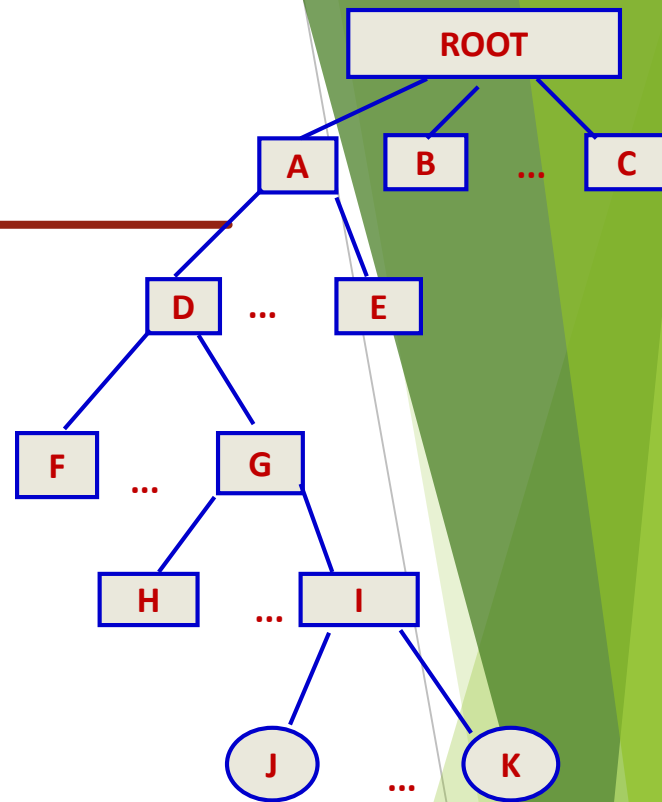
练习1

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60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，...表示有若干内容未显示。

(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

- ▶ 假设一块刚格式化好的磁盘大小为 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)

磁盘空间管理、内存结构、文件操作.....

文件系统的实现(2)

6. 磁盘空间管理(1/2)

► 位图法

- 用一串二进制位反映磁盘空间中分配使用情况，每个物理块对应一位，分配物理块为**0**，否则为**1**
- 申请物理块时，可以在位示图中查找为**1**的位，返回对应物理块号
- 归还时，将对应位转置**1**

► 空闲块表

将所有空闲块记录在一个表中，即空闲块表，有两项

► 空闲块链表

- 把所有空闲块链成一个链
- 扩展：成组链接法 ✓

磁盘空间管理(2/2)

已知块号，则磁盘地址：

$$\text{柱面号} = [\text{块号} / (\text{磁头数} \times \text{扇区数})]$$

$$\text{磁头号} = [(\text{块号} \bmod (\text{磁头数} \times \text{扇区数})) / \text{扇区数}]$$

$$\text{扇区号} = (\text{块号} \bmod (\text{磁头数} \times \text{扇区数})) \bmod \text{扇区数}$$

已知磁盘地址：

$$\begin{aligned} \text{块号} = & \text{柱面号} \times (\text{磁头数} \times \text{扇区数}) + \text{磁头号} \times \text{扇区数} \\ & + \text{扇区号} \end{aligned}$$

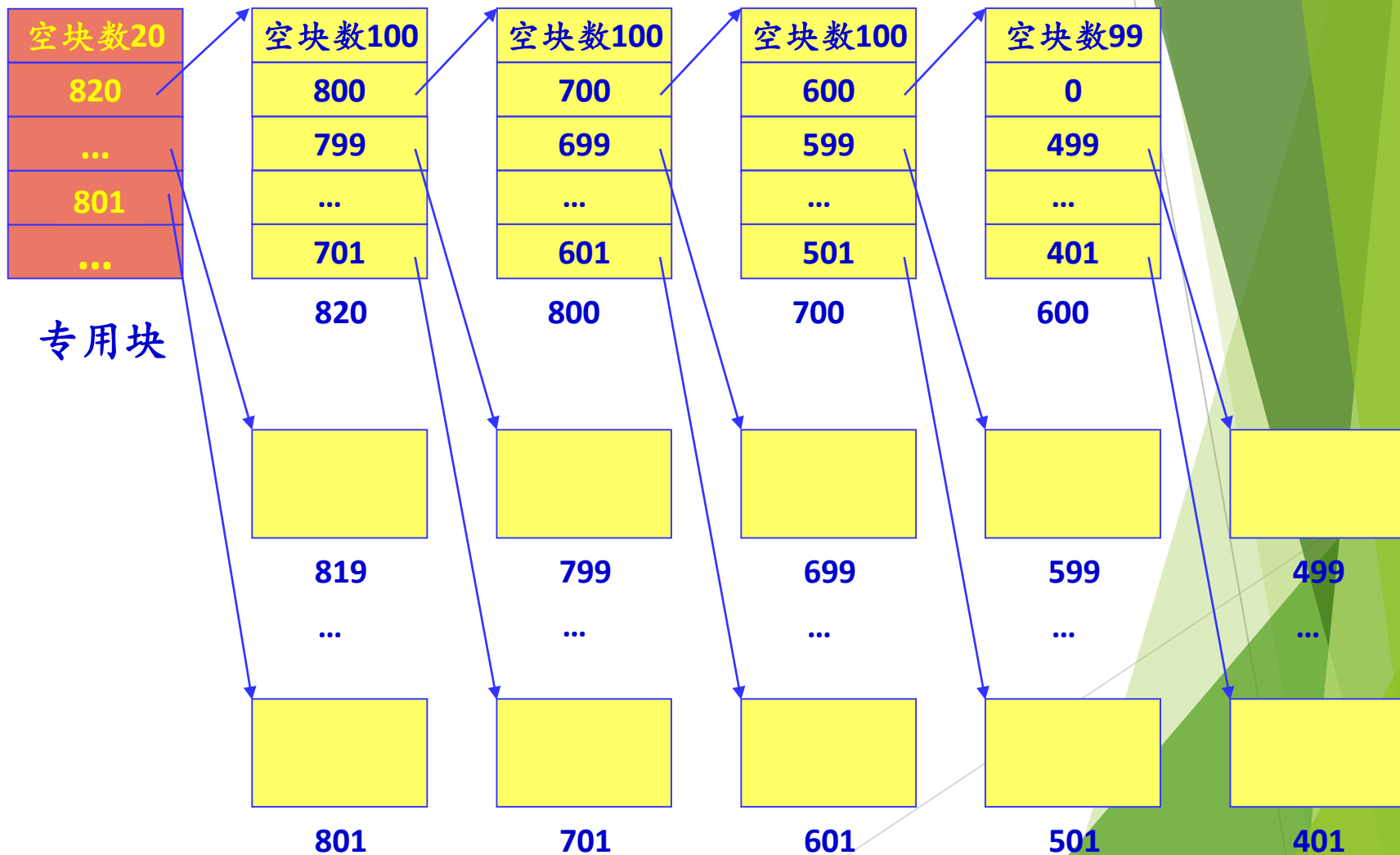
位图计算公式：

$$\text{已知字号} i, \text{位号} j: \text{块号} = i \times \text{字长} + j$$

$$\text{已知块号}: \text{字号} = [\text{块号} / \text{字长}] \quad \text{位号} = \text{块号} \bmod \text{字长}$$

成组链接法 (1/3)

空闲块



成组链接法 (2/3)

分配和回收的算法如下：

1. 分配一个空闲块

查L单元（空闲块数）内容：

当空闲块数 >1 $i := L + \text{空闲块数}$ ；

从i单元得到一空闲块号；

把该块分配给申请者；

空闲块数减1；

当空闲块数 $=1$ 取出 $L+1$ 单元内容（一组的第一块块号或0）；

其值 $=0$ 无空闲块，申请者等待

不等于零，把该块内容复制到专用块；

该块分配给申请者；

把专用块内容读到内存L开始的区域。

成组链接法 (3/3)

2. 归还一块

查L单元的空闲块数;

当空闲块数<100 空闲块数加1;

$j := L + \text{空闲块数};$

归还块号填入j单元。

当空闲块数=100, 则把内存中登记的信息写入归还块中;

把归还块号填入L+1单元;

将L单元置成1。

7. 内存中所需的数据结构——UNIX

运行时文件结构

文件描述符

(1) 用户打开文件表

- 每个进程一个
- 维护打开文件的状态和信息
- 进程的PCB中记录了用户打开文件表的位置

文件描述符	打开方式	读写指针	系统打开文件表入口
...

(2) 系统打开文件表

- 整个系统一个，放在内存
- 用于保存已打开文件的FCB

FCB(i节点)信息	引用计数	修改标志
...

8. 文件操作的实现

创建文件：实质是建立文件的FCB

(1) 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有关内容

(2) 分配必要的存储空间

目的：建立系统与文件的联系

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（FCB）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→错误返回

② 申请空目录项，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 (FCB) 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)

系统为每个打开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，找到对应的入口；
- 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 (文件句柄或文件描述符, 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

练习题

- ▶ 怎样实现系统调用rename（给文件重命名）？
- ▶ 怎样实现系统调用copy（复制文件）？
- ▶

9. 文件共享

文件共享 是指 一个文件被多个用户或进程使用
多用户系统中的文件共享是很必要的

一种实现——文件别名：
硬链接 和 软链接

目的：节省
时间和存储
空间；交换
信息

硬链接

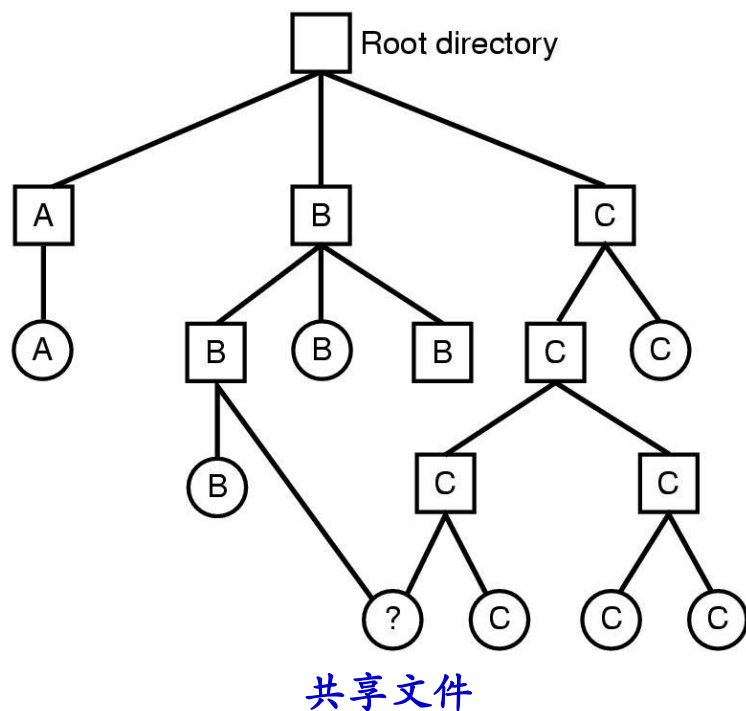
- 利用多个路径名描述同一共享文件，多个目录项指向一个文件

软链接

- 建立一种特殊类型的文件，通过文件内容与另一个文件连接
例子：“快捷方式”指向其他文件

硬链接(1/2)

例如：通过“**连接 (Link)**”命令，在用户自己的目录中对要共享的文件建立起相应的表目，即**建立两个文件的等价关系**



- FCB中直接给出文件地址?
- 目录项指向I节点

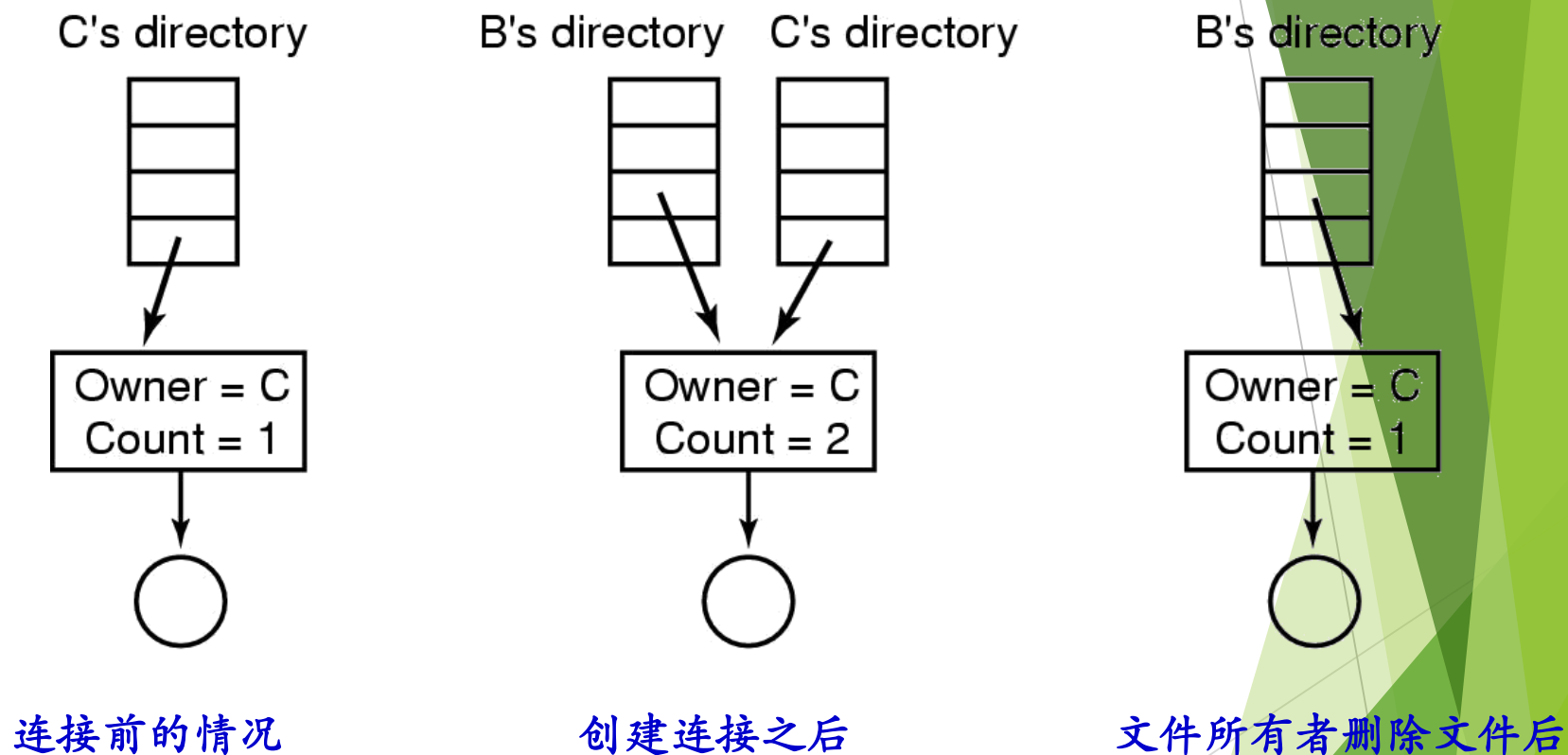
以Linux为例：目录项分为两部分：文件名和索引结点；

通过多个文件名链接(link)到同一个索引结点,可建立同一个文件的多个彼此平等的别名;

别名的数目记录在索引结点的链接计数中，若其减至0，则文件被删除

问题：删除文件时怎样考虑？

硬链接(2/2)



软链接

又称 符号连接、快捷方式

- ▶ 建立一种特殊类型 (Link) 的文件，其内容是要共享的文件路径名
- ▶ 只有真正的文件所有者才有指向i节点的指针
- ▶ 可以建立任意的别名关系，甚至原文件是在其他计算机上

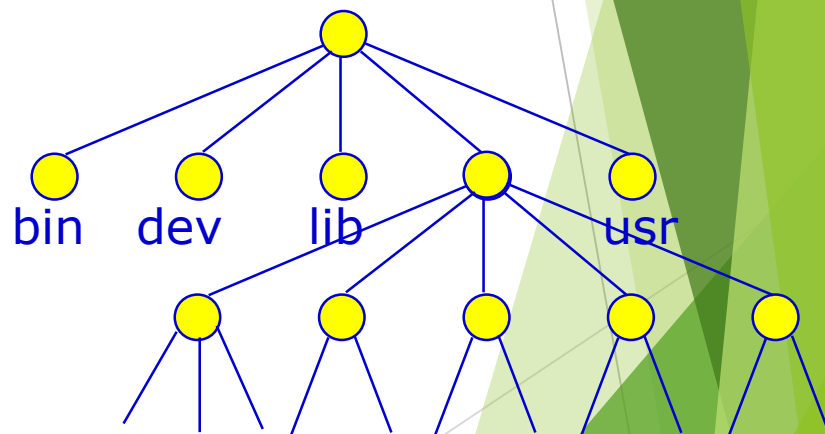
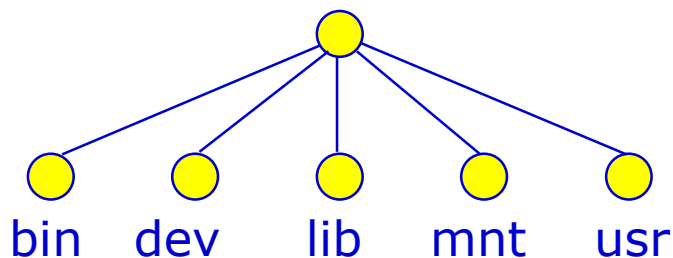
问题：系统开销大；目录结构可能形成环状

优势：计算机网络环境下可用

10. 挂载(mount)和卸载(unmount)

挂载：将一个文件系统加入到另一个文件系统

用户提供：被挂载的文件系统的根目录/挂载点



安装

NTFS文件系统

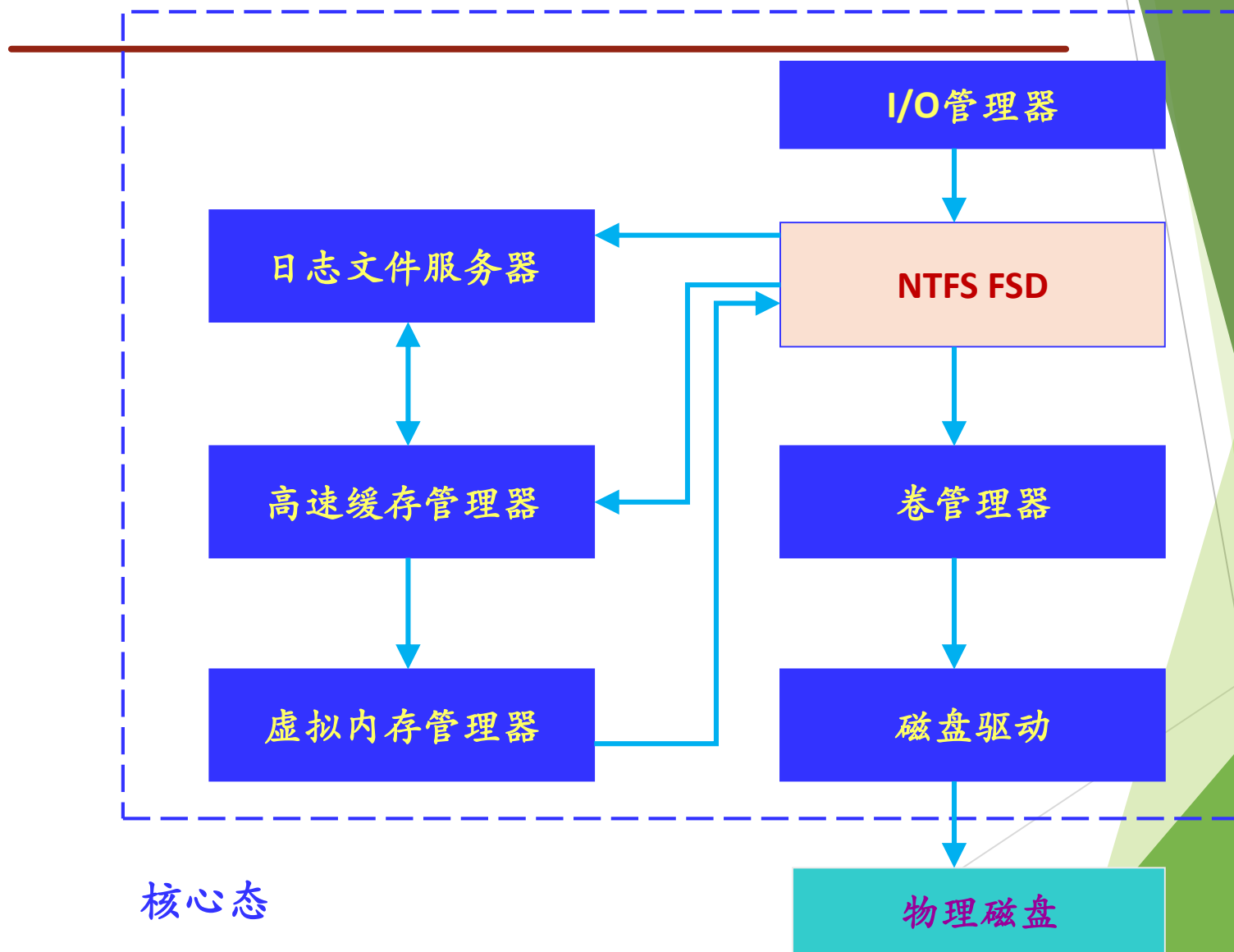
(1) 概述

设计目标:

可靠、高效、安全

- ▶ 为了满足可靠数据存储和访问的需求，NTFS提供了**基于原子事务（Atomic transaction）**概念的文件系统可恢复性
- ▶ NTFS对关键文件系统信息采用冗余存储
- ▶ NTFS提供了综合的安全模型（对象模型）以及支持加密文件系统（EFS, Encrypted File System），可以阻止非授权用户访问加密文件
- ▶ NTFS为改进的多级目录结构，支持文件别名
- ▶ NTFS文件由多个文件属性构成，每个属性由属性名和属性流（stream，简单字节队列）组成
- ▶ NTFS文件支持数据压缩功能
- ▶ NTFS卷结构支持容错功能

(2) NTFS FSD的作用



(3) NTFS的磁盘结构

► 卷

- 卷建立在磁盘分区上
- 一个磁盘可以有多个卷，一个卷也可以有多个磁盘组成
- （已格式化的）卷上的数据可分为：元数据和用户数据

► 簇

- 簇是磁盘空间分配和回收的基本单位
- 簇大小是用户在使用Format命令或其他格式化程序格式化卷时确定的
- **NTFS使用LCN（Logical Cluster Number，逻辑簇号）和VCN（Virtual Cluster Number，虚拟簇号）来进行簇的定位**
- ✓ **LCN是对整个卷中所有的簇从头到尾所进行的简单编号**
- ✓ **VCN是对属于特定文件的簇从头到尾进行编号，以便于引用文件中的数据**

NTFS分区和簇大小

卷大小	每簇的扇区数	簇大小
<=512MB	1	512B
512MB~1GB	2	1KB
1GB~2GB	4	2KB
2GB~4GB	8	4KB
4GB~8GB	16	8KB
8GB~16GB	32	16KB
16GB~32GB	64	32KB
>32GB	128	64KB

NTFS一卷最大为 2^{64} 字节

(4) NTFS文件组织

- ▶ 文件名称
- ▶ 主控文件表
- ▶ 文件记录
- ▶ 常驻属性与非常驻属性

文件名称

- ▶ NTFS路径名中的每个文件名/目录名的长度可达255个字节，可以包含Unicode字符、多个空格及句点
- ▶ NTFS卷上的每个文件都有一个64位的，称为文件引用号的唯一标识
- ▶ 文件引用号的组成
 - ▶ 文件号（48位，该文件在MFT中的位置）
 - ▶ 文件顺序号

主控文件表

► MFT (Master File Table, 主控文件表)

NTFS卷结构的核心，是NTFS中最重要的系统文件，包含了卷中所有文件的信息

► MFT是以文件记录数组来实现的

- 每一个记录描述了卷中的一个文件或目录（例如：1K大小）
- MFT自身也是一个文件
- 文件系统中每一个文件或目录都至少有一个MFT项
- 一般：每个MFT项的前几十个字节有固定的头结构，用以描述本MFT项的相关信息；之后的字节用于存放属性；文件或目录的其余信息可放到卷中其他可用簇中
- 访问文件：首先在MFT中找到对应的MFT项，再根据MFT项记录的信息找到文件内容

属性

- ▶ NTFS中，所有信息都被称为属性
 - ▶ NTFS文件是属性/属性值的集合，通常所说的文件内容是指未命名数据属性流
 - ▶ 每个属性由单个的流(stream)组成
- ▶ NTFS提供对属性的各种操作
 - ▶ 读/写操作一般是针对文件的未命名属性的
 - ▶ 创建、删除、读取（字节范围）以及写入（字节范围）
- ▶ 常驻属性与非常驻属性
 - ▶ 标准信息、文件名、索引根是常驻属性
 - ▶ 延展(run)或延伸(extent)：MFT之外的区域，可存放变长的非常驻属性

NTFS卷上文件的常用属性

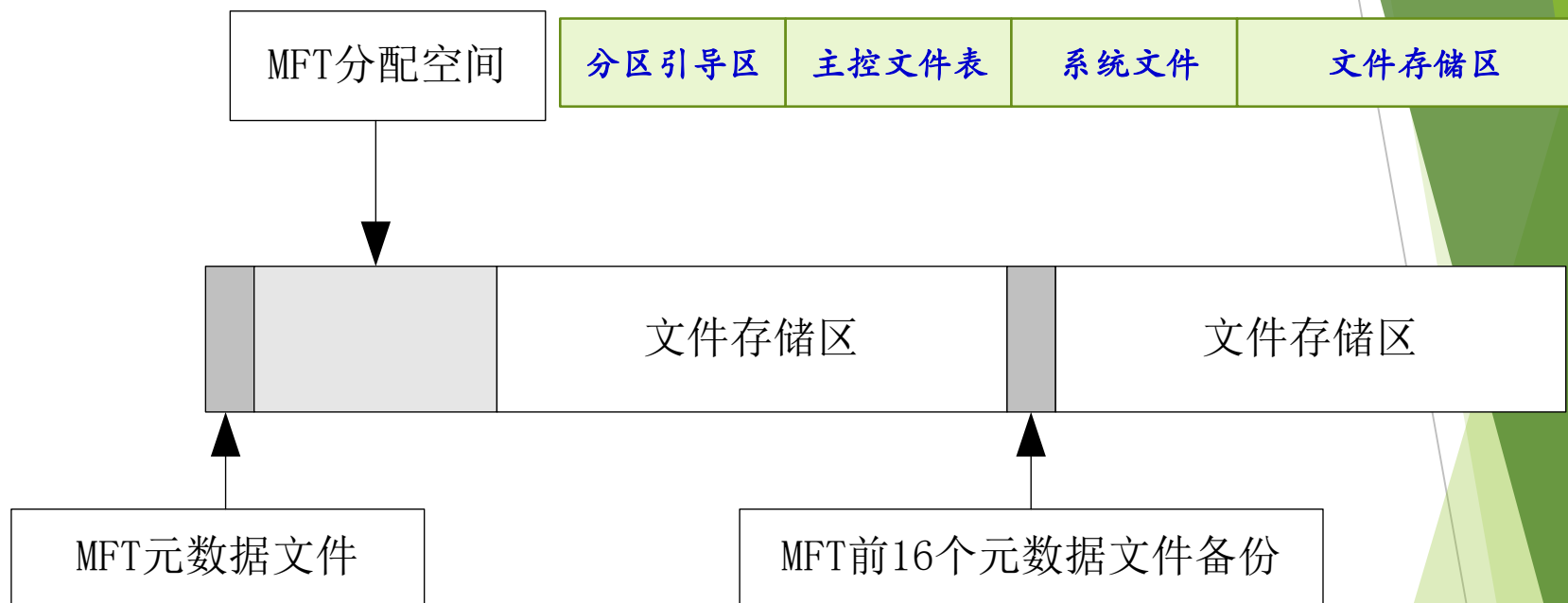
属性	描述
标准信息	标志位，时间戳等
文件名	Unicode文件名
安全描述符	废弃。安全信息现在用\$Extend \$Secure表示
属性列表	额外的MFT记录的位置，如果需要的话
对象ID	对此卷唯一的64位文件标识符
再解析点	用于加载和符号链接
卷名	当前卷的名字（仅用于\$Volume）
卷信息	卷版本（仅用于\$Volume）
索引根	用于目录
索引分配	用于很大的目录
位图	\$MFT文件及索引的位图
日志工具流	控制记录日志到\$LogFile
数据	数据流；可以重复

NF7S元数据文件

MFT的前16个项
是为元数据文件
保留的，之后则
是普通的用户文
件和目录

- 0: \$Mft: MFT本身
- 1: \$MftMirr: MFT镜像
- 2: \$LogFile: 日志文件
- 3: \$Volume: 卷文件
- 4: \$AttrDef: 属性定义表
- 5: \$\: 根目录文件
- 6: \$Bitmap: 位图文件
- 7: \$Boot: 引导文件
- 8: \$BadClus: 坏簇文件
- 9: \$Secure: 安全文件
- 10: \$UpCase: 大小写字符转换表文件
- 11: \$Extended metadata directory: 扩展元数据目录
- 12, 13, 14, 15:
- >15: 其他用户文件和目录

主控文件表空间分配



NTFS把磁盘分成了两大部分，其中大约12%分配给了MFT，余下的88%的空间被分配用来存储文件

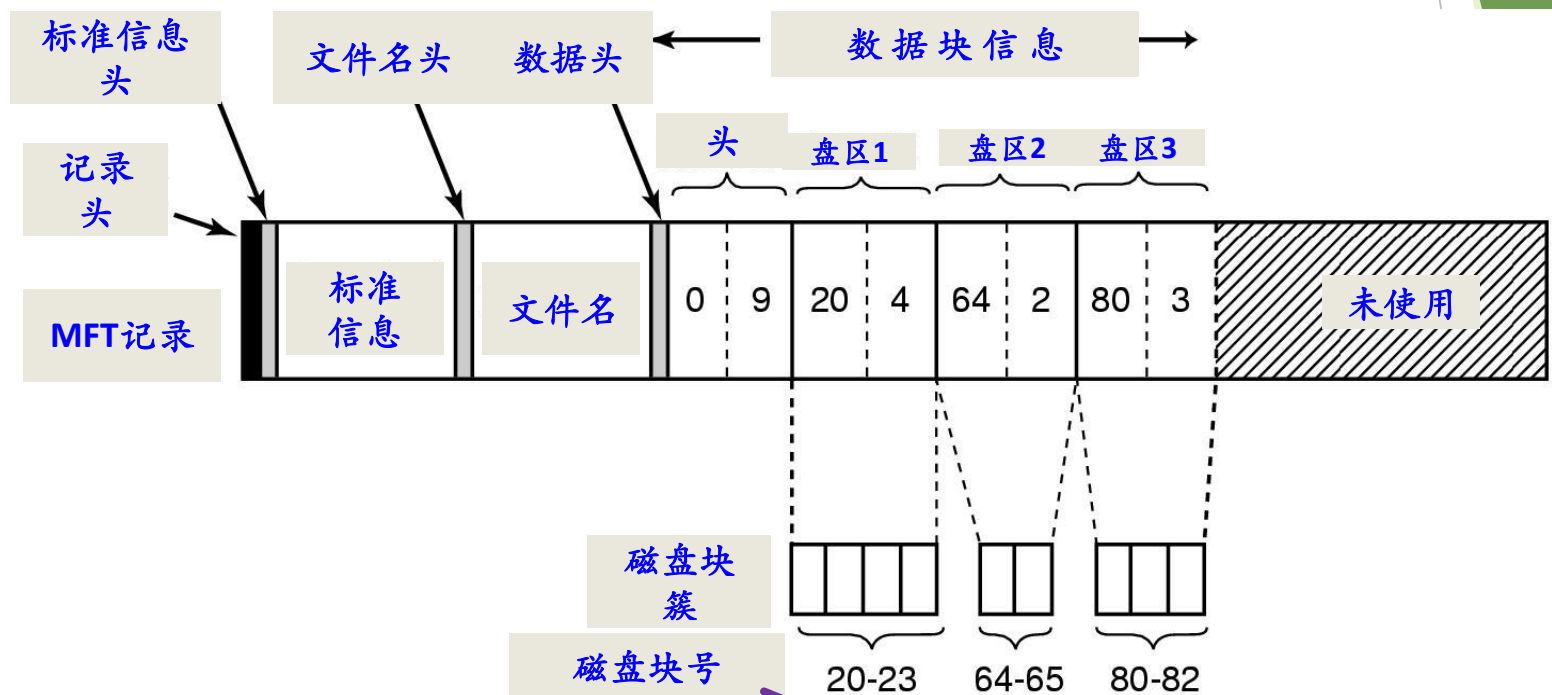
MFT例子1

标准信息	文件名	文件数据
------	-----	------

小文件的MFT

MFT例子2

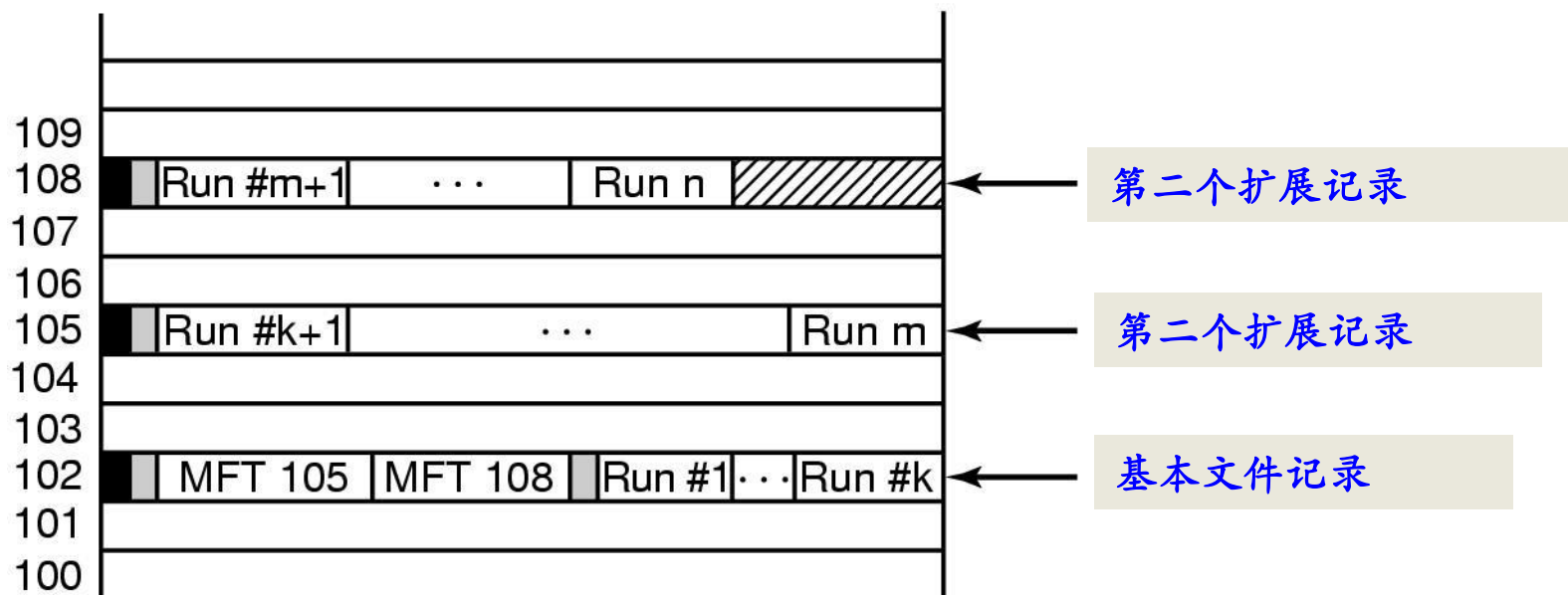
大文件的MFT



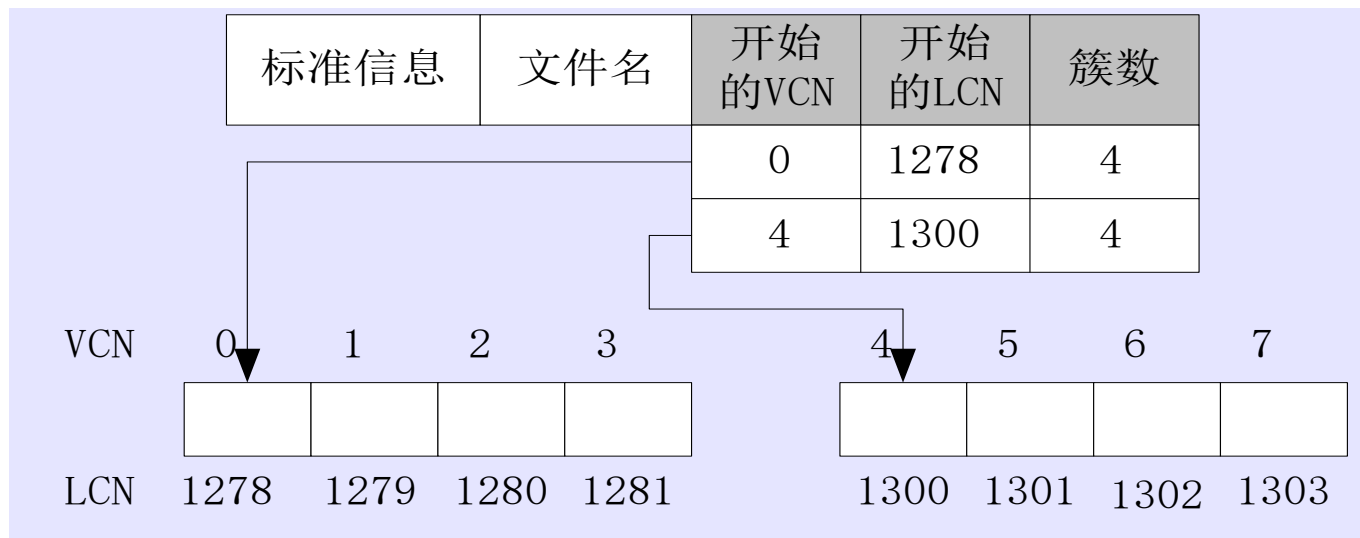
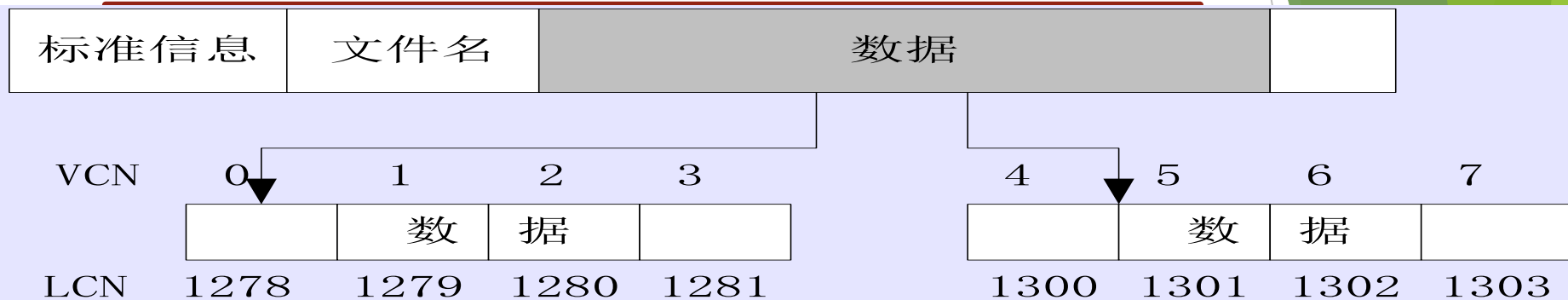
一个run或extent

MFT记录扩展

- 如果一个文件的属性很多，一个MFT项不能容下全部属性，需要使用多个MFT项

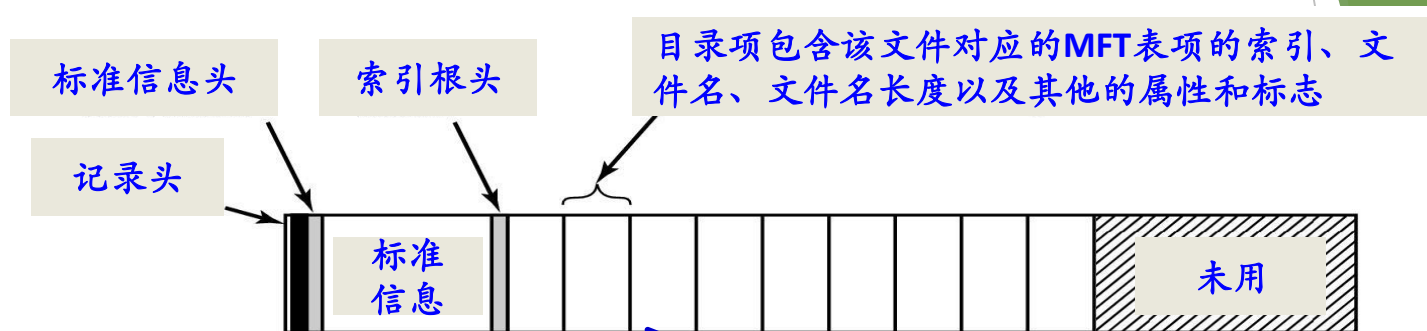


一般文件的常驻属性与非常驻属性



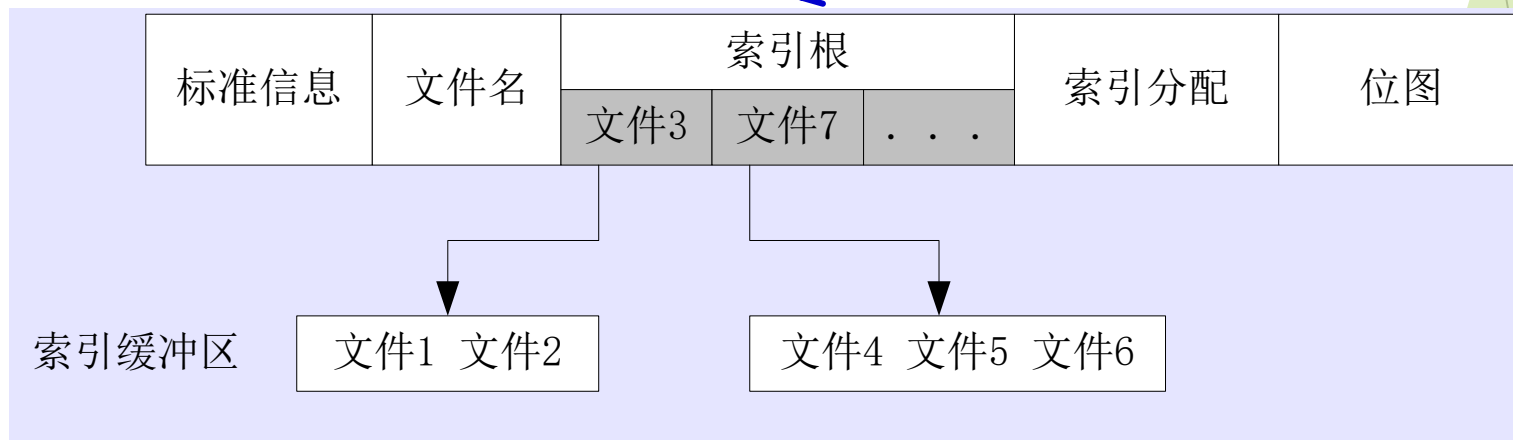
为了便于NTFS快速查找，具有多个run的常驻数据属性头中包含了VCN-LCN映射关系

目录的常驻属性与非常驻属性



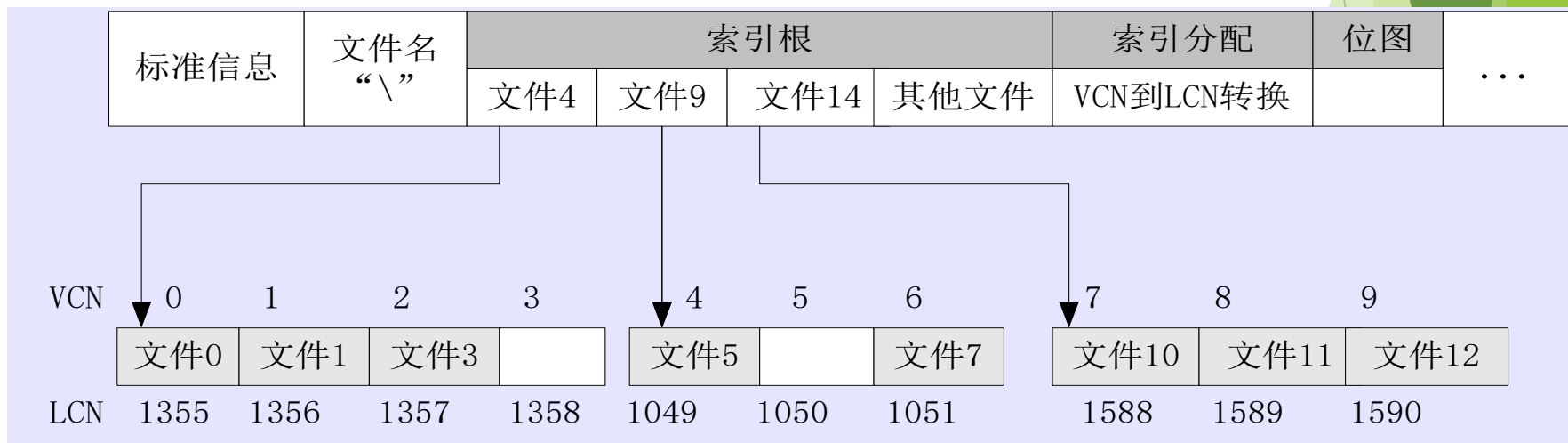
小目录的MFT

大目录的MFT



NTFS目录组织与索引

- ▶ 将目录中的文件名和子目录名进行排序，保存在**索引根属性**中(文件目录是文件名的索引，还包括文件引用、文件时间和大小等信息)
- ▶ 对于一个大目录，经排序的文件(目录)名实际存储在4KB大小的**索引缓冲区**中(索引缓冲区是通过B+树组织的)
- ▶ **索引分配属性**包含了索引缓冲区的VCN到LCN映射
- ▶ **位图属性**跟踪在索引缓冲区中哪些VCN是已使用而哪些是空闲的



NTFS数据压缩 (1/3)

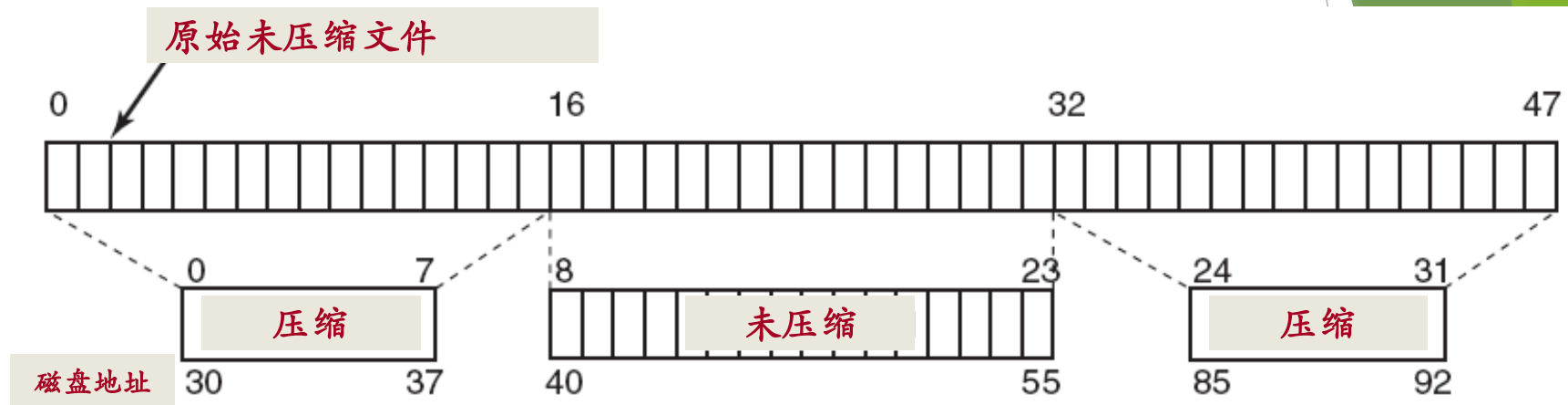
- ▶ 数据压缩是NTFS文件系统的一个重要特征
- ▶ NTFS压缩功能可以对单个文件、整个目录或NTFS卷上的整个目录树进行压缩
- ▶ NTFS压缩只在用户数据上进行，而不能在文件系统元数据上进行
- ▶ 数据压缩可以减少磁盘使用空间，但是由于每次解压缩需要大量的数据运算，使用压缩功能将会导致NTFS卷的性能下降

(如果要拷贝一个压缩文件，其过程是：解压缩、拷贝、重新对拷贝的文件进行压缩，这些都大大增加了CPU的处理时间)

NTFS数据压缩 (2/3)

- ▶ NTFS是以16个簇为压缩单元进行一般文件的压缩
- ▶ NTFS决定压缩这16个簇后是否能腾出一个簇，如果能则只分配所需的簇数，并将数据写到磁盘上；否则直接将数据写到磁盘上
- ▶ 当NTFS向压缩文件写数据时，它确保每个run都以一个虚拟16簇边界开始
- ▶ 每个run中VCN都是以16的倍数开始的，并且run的长度不大于16

NTFS数据压缩 (3/3) — 示例



(a)



(b)

NTFS可恢复性

- ▶ 通过日志记录来实现
- ▶ 子操作在磁盘运行之前，记录在日志文件中
- ▶ 系统恢复阶段，NTFS根据日志文件中的文件操作信息，对部分完成的事务进行重做或者撤销，保证磁盘文件系统的一致性
(预写日志记录)

重点小结

- 基本概念
 - ▶ 文件、文件系统、文件目录
 - ▶ 文件类型
 - ▶ 文件的逻辑结构
- 文件系统实现
 - ▶ 文件系统布局
 - ▶ 文件的物理结构
 - ▶ 文件目录的实现
 - ▶ 内存结构
 - ▶ 文件操作
 - ▶ 磁盘空间管理
 - ▶ 文件共享
- 文件系统实例
 - ▶ UNIX, FAT16/32, NTFS

作业提交时间：

2020年12月13日 晚23:30

作业8

1、某虚拟文件系统空间的最大容量为4TB，磁盘块（基本分配单位）大小为1KB。文件控制块（FCB）包含一个512B主索引表。请回答下列问题。

（1）假设主索引表只采用直接索引结构，主索引表中保存文件占用的磁盘块号。主索引表表项最少要占多少字节？可支持的单个文件最大长度是多少字节？

（2）假设主索引表采用如下结构：第0~7字节采用<起始块号，块数>的格式，表示文件创建时连续分配给文件的磁盘空间。其中起始块号占6B、块数占2B，剩余504B采用直接索引结构，一个索引项占6B。试问，采用这种结构可支持的单个文件的最大长度是多少字节？为了使单个文件的长度达到最大，起始块号和块数分别所占的合理字节数是多少？请说明理由。

作业提交时间：

2020年12月13日 晚23:30

作业8

2、假设某文件系统是一级目录结构，文件的数据一次性写入磁盘，已写入的文件不可修改，但可多次创建新文件。

请回答如下问题：

(1) 在顺序、链接、索引三种文件的物理结构方式中，哪一种更合适？请说明理由。为定位文件的数据块，需要在FCB中设计哪些相关描述字段？

(2) 为快速找到文件，是集中存储FCB好，还是与对应的文件数据块一起存储好？请说明理由。

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic look.

Thanks

The End