

# 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9讲

## 文件系统1

# 文件管理

---

- ◎ 文件与文件系统
- ◎ 文件的存储介质
- ◎ 磁盘空间管理
- ◎ 文件控制块及文件目录
- ◎ 文件的物理结构
- ◎ 文件系统的实现
- ◎ 文件系统实例——UNIX

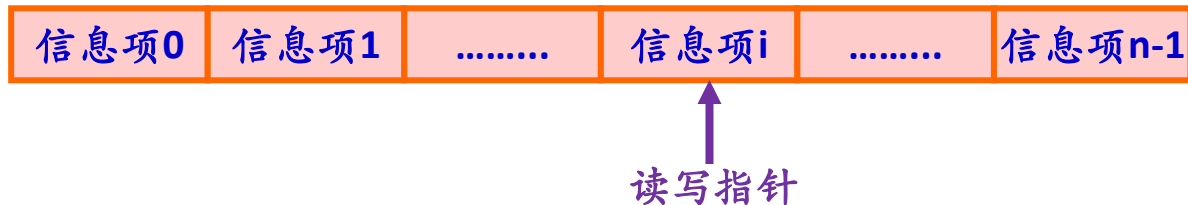
文件的分类、文件的逻辑结构

# 文件与文件系统

# 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



# 如何设计一个文件系统

---

## 操作系统角度:

怎样组织、管理文件?

- ✓ 文件的描述、分类
  - ✓ 文件目录的实现
  - ✓ 存储空间的管理
  - ✓ 文件的物理地址
  - ✓ 磁盘实际运作方式(与设备管理的接口)
  - ✓ 文件系统性能
- 等等

## 用户角度:

文件系统如何呈现在用户面前:

- ✓ 一个文件的组织
  - ✓ 如何命名?
  - ✓ 如何保护文件?
  - ✓ 可以实施的操作?
- 等等

文件管理的需求分析

# 文件系统

---

- ◎ 操作系统中统一管理信息资源的一种软件，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，并提供文件的保护、保密手段
- 向用户提供一个方便使用、易于维护的接口，并向用户提供有关统计信息
- 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

### 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)

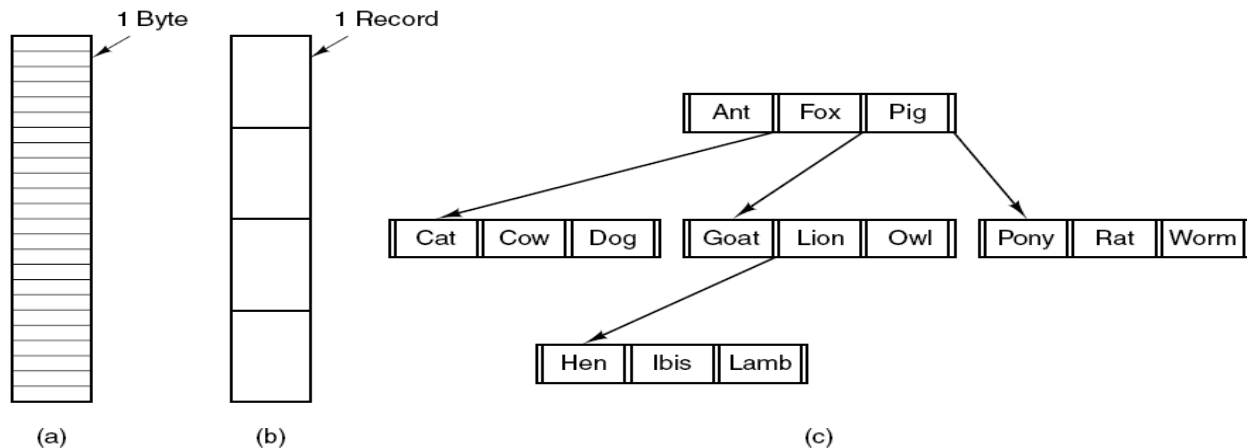
字符设备文件：和输入输出有关，用于模仿串行I/O设备，例如终端，打印机，网卡等

块设备文件：磁盘



# 文件的逻辑结构


从用户角度看文件，由用户的访问方式确定



还可以组织成堆、顺序、索引、索引顺序、散列等结构

# 典型的文件逻辑结构与文件存取

---

- ◎ **流式文件**：构成文件的基本单位是字符  
文件是有逻辑意义、无结构的一串字符的集合
- ◎ **记录式文件**：文件由若干个记录组成，可以按记录进行读、写、查找等操作  
每条记录有其内部结构 

◆ 顺序存取(访问)

◆ 随机存取(访问)

提供读写位置(当前位置)

例如：UNIX的seek操作

物理块、簇、磁盘结构、扇区

# 文件的存储介质

# 存储介质与物理块

---

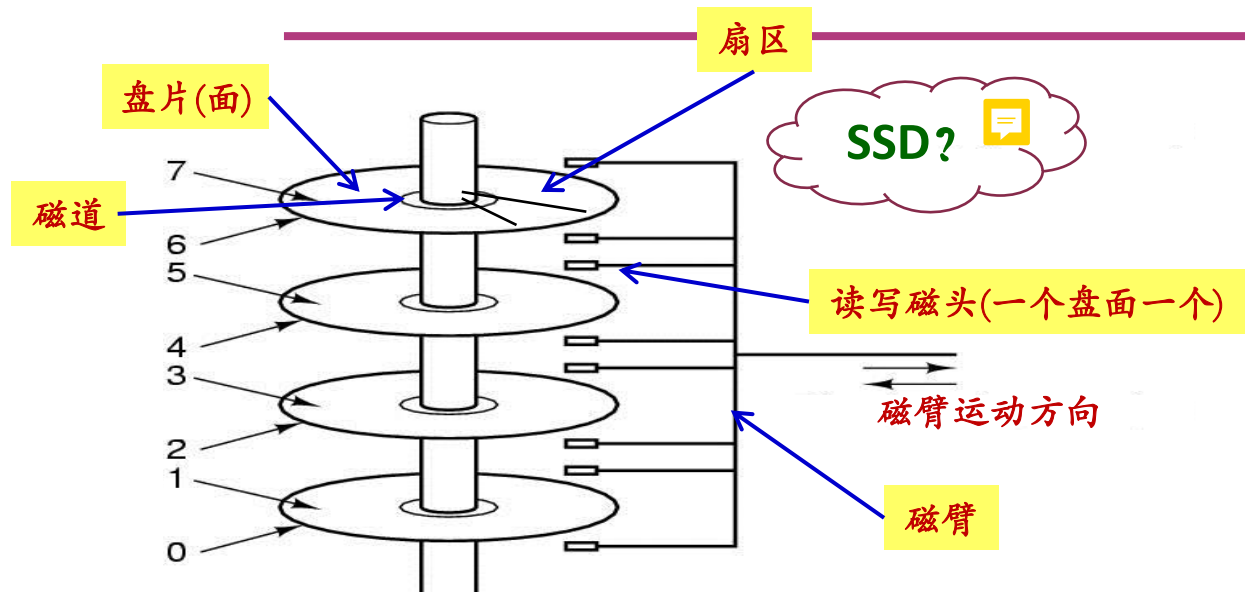
- ◎ 典型的存储介质

磁盘(包括固态硬盘SSD)、磁带、光盘、U盘、.....

- ◎ 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

# 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

SSD?



## 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

# 磁盘空间管理

# 有关数据结构

---

## ◎ 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
- 主要两项内容：起始块号，块数

## ◎ 空闲块链表

- 把所有空闲块链成一个链
- 扩展：成组链接法 ✓



# 磁盘地址与块号的转换

---

已知块号，则磁盘地址：

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

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

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：

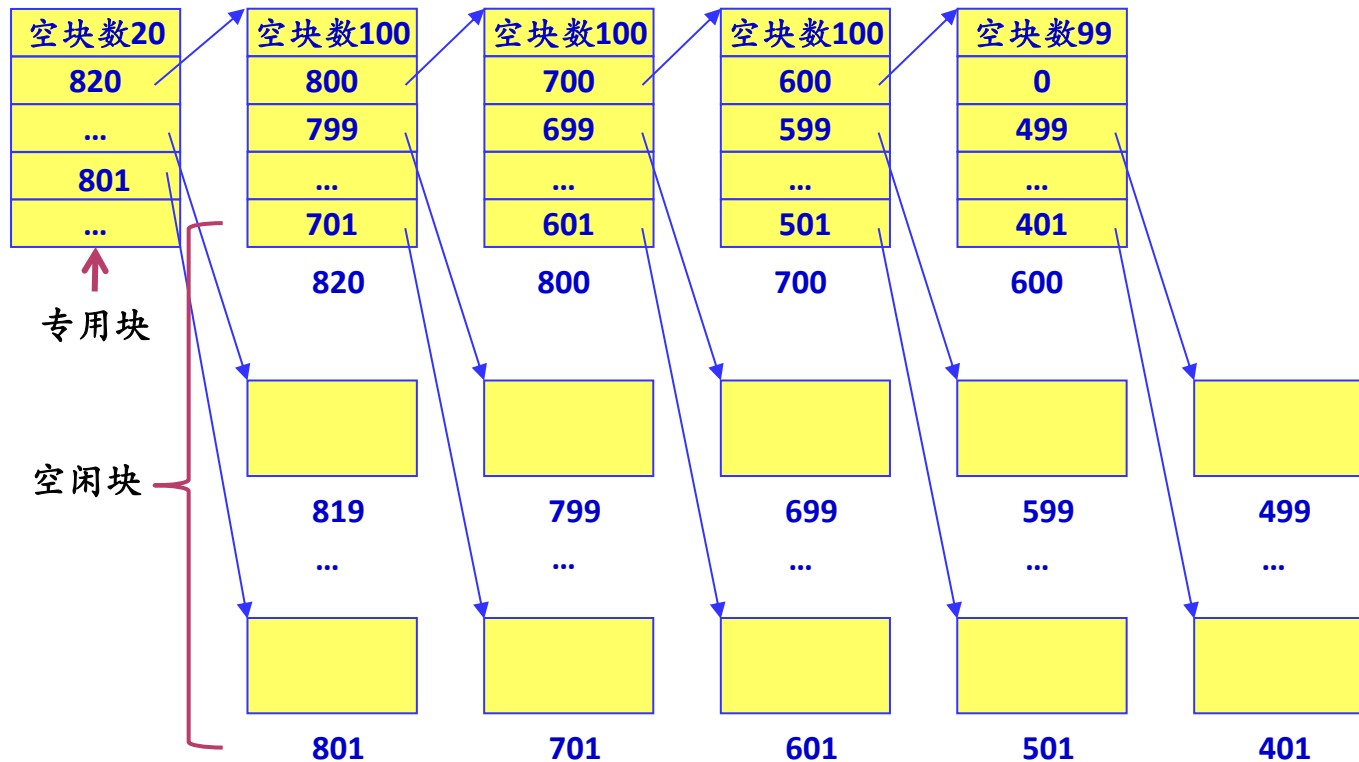
块号 = 柱面号  $\times$  (磁头数  $\times$  扇区数) + 磁头号  $\times$  扇区数 + 扇区号

位图计算公式：

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

已知块号：字号 =  $\lfloor \text{块号} / \text{字长} \rfloor$  位号 = 块号  $\bmod$  字长

# 成组链接法设计思想



# 成组链接法一分配算法

---

## 分配一个空闲块

查L单元（空闲块数）：

- 当空闲块数  $> 1$   $i = L + \text{空闲块数}$ ;  
从  $i$  单元得到一个空闲块号;  
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  
其值不等于零，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—回收算法

---

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。

文件属性、树形结构

# 文件控制块及文件目录

# 文件属性

---

- ◎ 文件控制块（File Control Block）

为管理文件而设置的数据结构，保存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，文件大小，文件地址，创建时间，最后修改时间，最后访问时间，保护，口令，创建者，当前拥有者，文件类型，共享计数，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)

# 基本文件操作

---

- **Create**
- **Delete**
- **Open**
- **Close**
- **Read**
- **Write**
- **Append**
- **Seek**
- **Get Attributes**
- **Set Attributes**
- **Rename**
- ... ..

```
/* File copy program. Error checking and reporting is minimal. */
```

```
#include <sys/types.h>
```

```
/* include necessary header files */
```

```
#include <fcntl.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

## 用基本文件操作构造其他操作示例

```
int main(int argc, char *argv[]);
```

```
/* ANSI prototype */
```

```
#define BUF_SIZE 4096
```

```
/* use a buffer size of 4096 bytes */
```

```
#define OUTPUT_MODE 0700
```

```
/* protection bits for output file */
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    int in_fd, out_fd, rd_count, wt_count;
```

```
    char buffer[BUF_SIZE];
```

```
    if (argc != 3) exit(1);
```

```
/* syntax error if argc is not 3 */
```

```
/* Open the input file and create the output file */
```

```
    in_fd = open(argv[1], O_RDONLY);
```

```
/* open the source file */
```

```
    if (in_fd < 0) exit(2);
```

```
/* if it cannot be opened, exit */
```

```
    out_fd = creat(argv[2], OUTPUT_MODE);
```

```
/* create the destination file */
```

```
    if (out_fd < 0) exit(3);
```

```
/* if it cannot be created, exit */
```



---

```
/* Copy loop */
while (TRUE) {
    rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
    if (rd_count <= 0) break; /* if end of file or error, exit loop */
    wt_count = write(out_fd, buffer, rd_count); /* write data */
    if (wt_count <= 0) exit(4); /* wt_count <= 0 is an error */
}

/* Close the files */
close(in_fd);
close(out_fd);
if (rd_count == 0) /* no error on last read */
    exit(0);
else
    exit(5); /* error on last read */
}
```

# 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元数据，以支持文件名到文件物理地址的转换
- ◉ 将所有文件的管理信息组织在一起，即构成文件目录

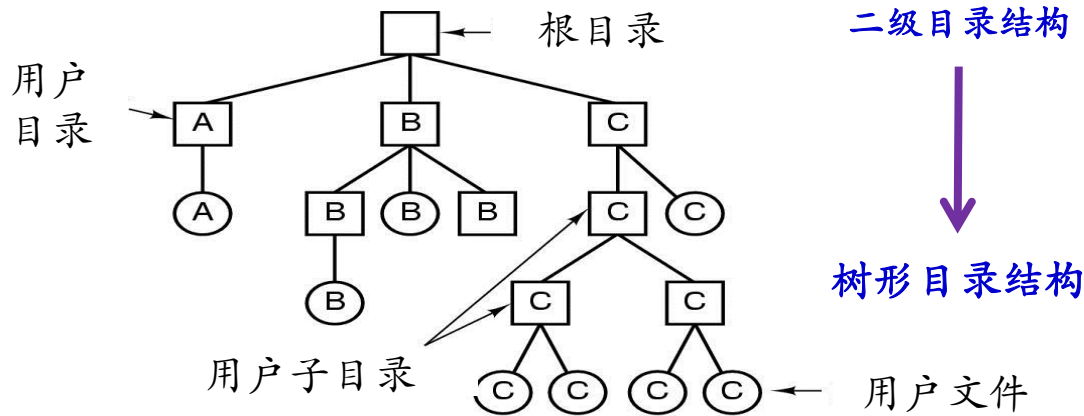
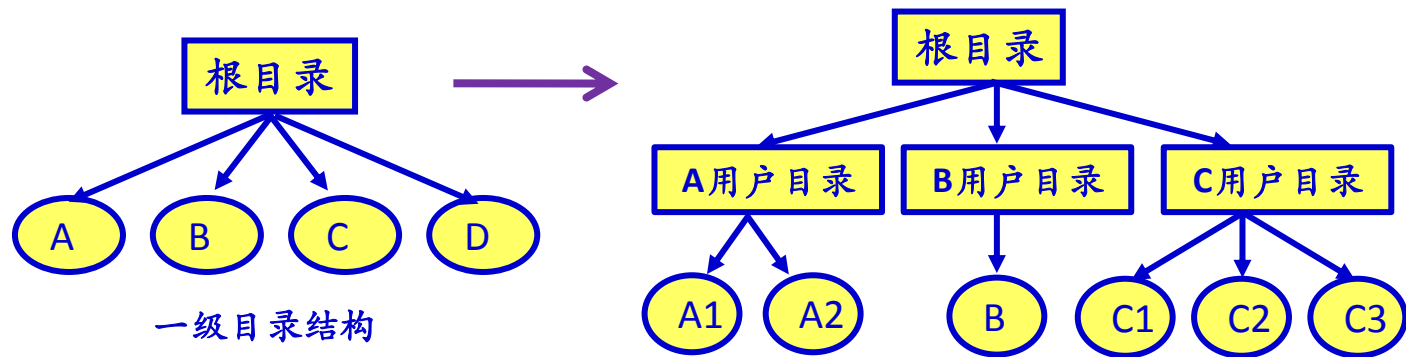
- ◉ 目录文件

- ◉ 将文件目录以文件的形式存放在磁盘上

- ◉ 目录项

- ◉ 构成文件目录的基本单元
- ◉ 目录项 可以是FCB，目录是文件控制块的有序集合

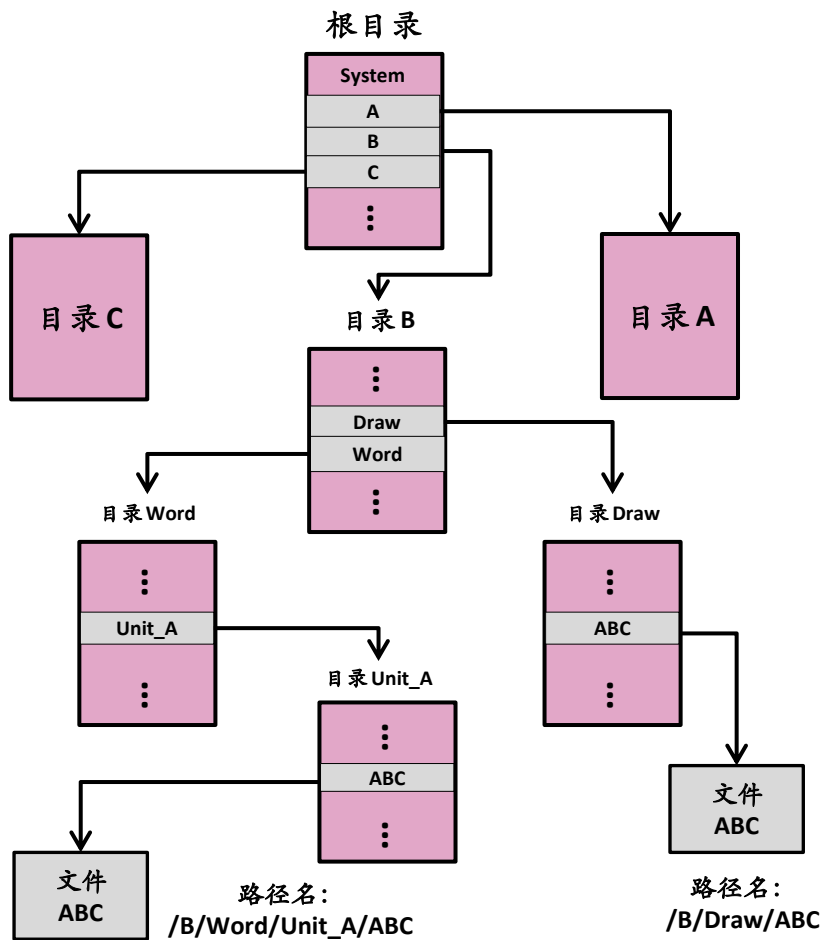
# 文件目录结构的演化



# 与目录相关的概念

---

- ◎ 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
- ◎ 目录操作
  - 创建目录、删除目录
  - 读目录、写目录、改名、复制



## 目录文件的关联

# 文件的物理结构

# 文件的物理结构

---

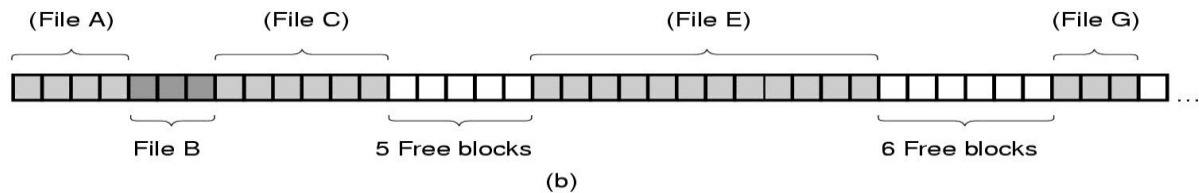
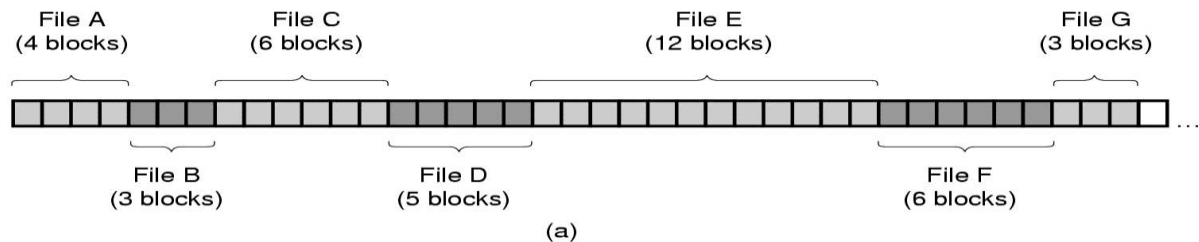
- 文件在存储介质上的存放方式

主要解决两个问题：

- 假设一个文件被划分成 $N$ 块，这 $N$ 块在磁盘上是怎么存放的？
- 其地址(块号或簇号)在FCB中是怎样记录的？

# 连续(顺序)结构

文件的的信息存放在若干连续的物理块中



在FCB中如何记录文件地址?





# 连续结构的优缺点

---

## ◎ 优点

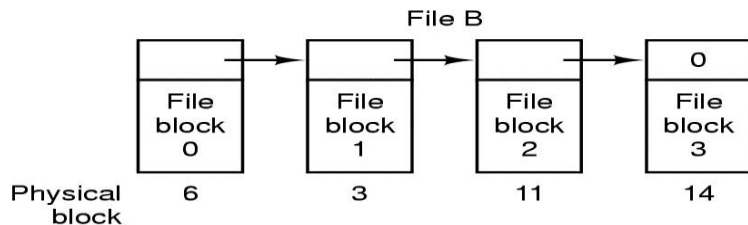
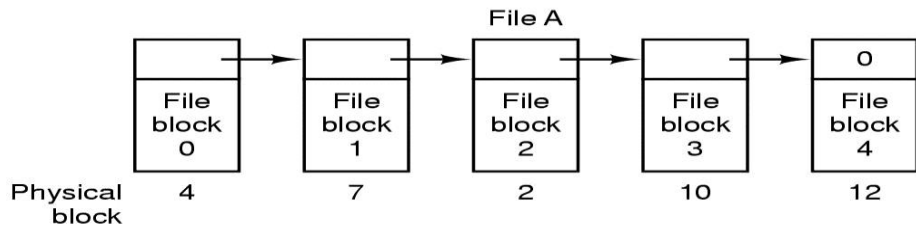
- 简单
- 支持顺序存取和随机存取
- 所需的磁盘寻道次数和寻道时间最少
- 可以同时读入多个块，检索一个块也很容易

## ◎ 缺点

- 文件不能动态增长  
预留空间：浪费 或 重新分配和移动
- 不利于文件插入和删除
- 外部碎片：紧缩技术

# 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中如何记录文件地址？

# 链接结构的优缺点

---

## ◎ 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

## ◎ 缺点

- 存取速度慢，不适于随机存取
- 可靠性问题，如指针出错
- 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**

# 文件分配表示意图

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？

# 索引结构

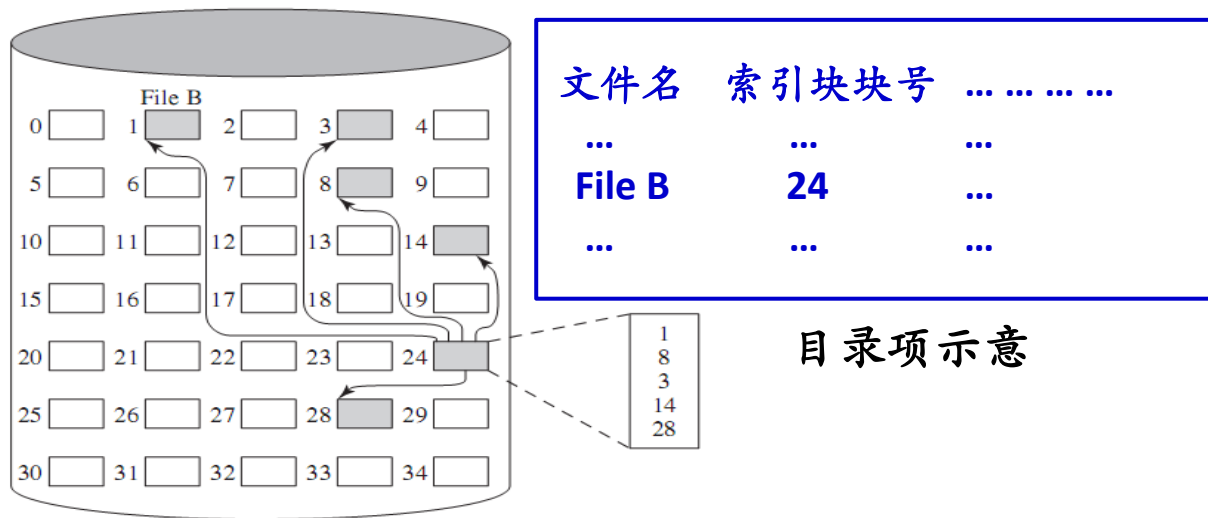
---

- 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，并将这些物理块的块号存放在该索引表中
- 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中如何记录文件地址？

索引表存放在何处？

# 索引结构示意图



# 链接结构的优缺点

---

## ◎ 优点

保持了链接结构的优点，又解决了其缺点

- 既能顺序存取，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

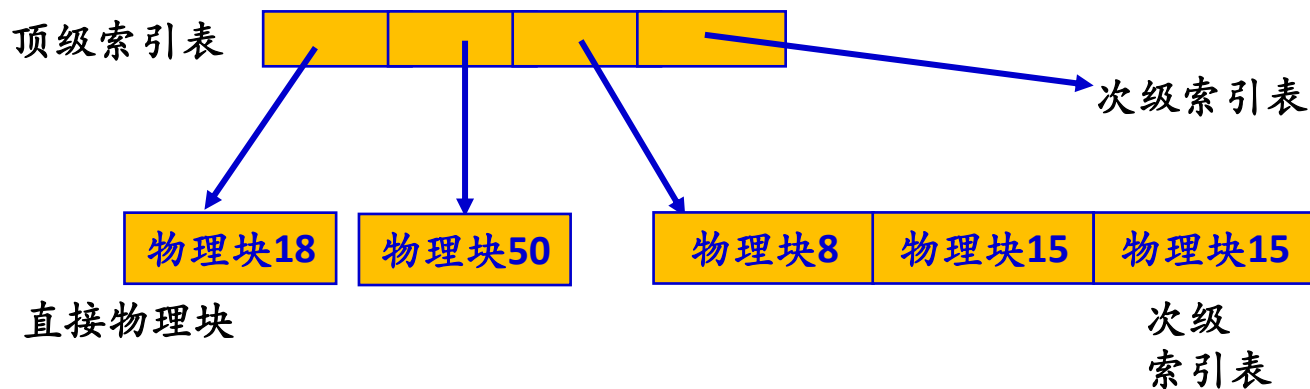
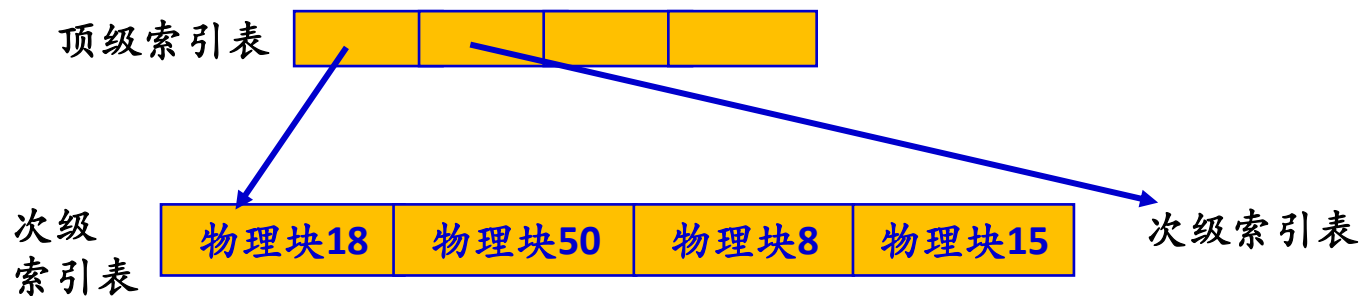
# 索引表的组织方式

---

- 问题：索引表很大，需要多个物理块存放时怎么办？
- 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  
将文件的索引表地址放在另一个索引表中
- 综合模式  
直接索引方式 与 间接索引方式 结合



# 多级索引与综合模式



# UNIX的三级索引结构

---

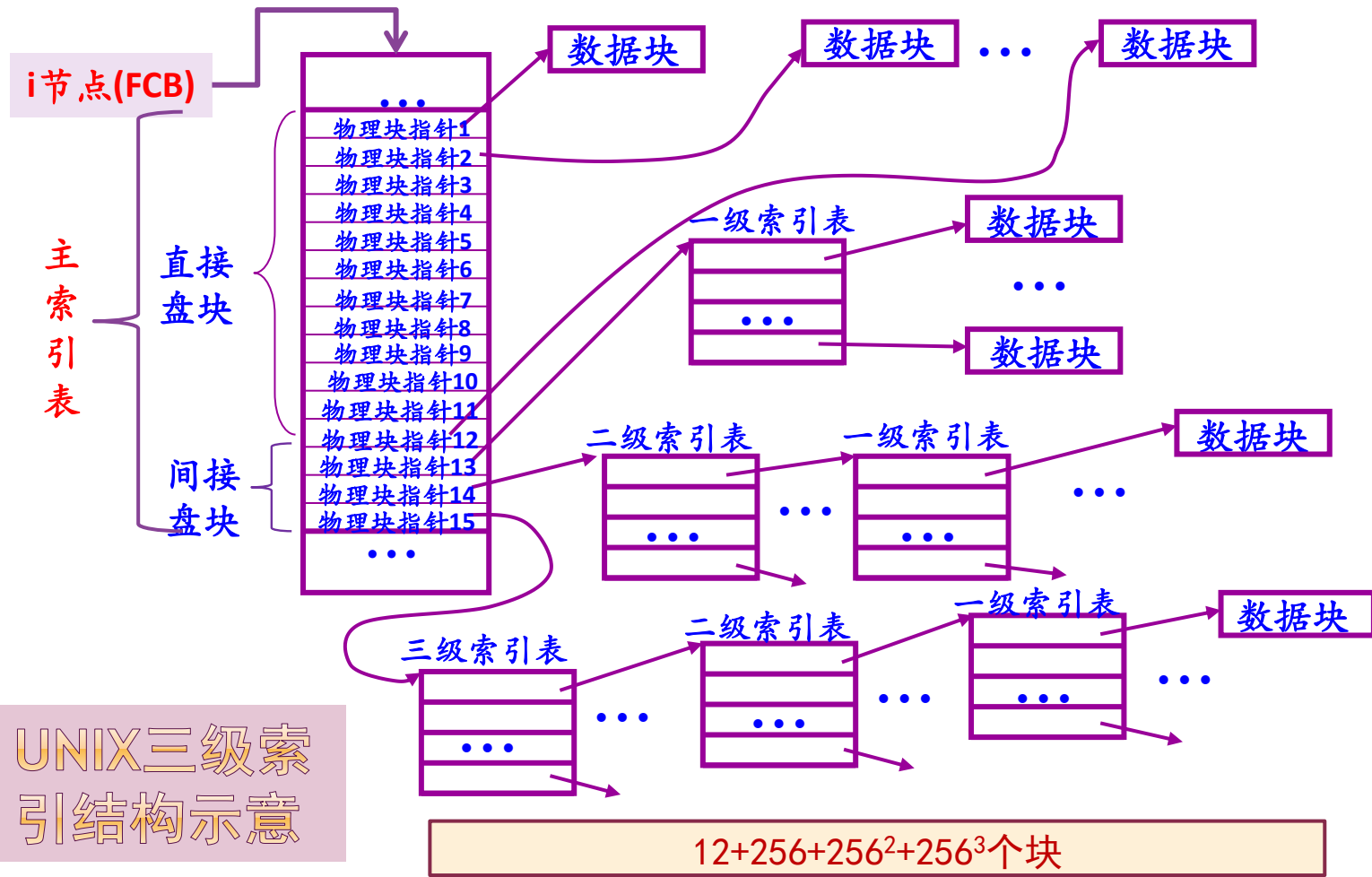
UNIX文件系统采用的是多级索引结构(综合模式)

- 每个文件的索引表有**15**个索引项，每项**2**个字节
- 前**12**项直接存放文件的物理块号（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，在该块中存放文件物理块的块号（一级索引表）

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

- 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

试问：采用这种结构，一个文件最大可达到 **?** 个物理块



磁盘上的布局、内存中的数据结构

# 文件系统的实现1

# 概述

---

- ◎ 实现文件系统需要考虑  
磁盘上 与 内存中 的内容布局
- ◎ 磁盘上
  - ✓ 如何启动操作系统？
  - ✓ 磁盘是怎样管理的？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
  - 当进程使用文件时，操作系统是如何支持的？
  - 文件系统的内存数据结构

# 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)**: 磁盘上的逻辑分区, 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
  - **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, 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

---

文件卷(分区)



.....

- ◎ 引导区

包括了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

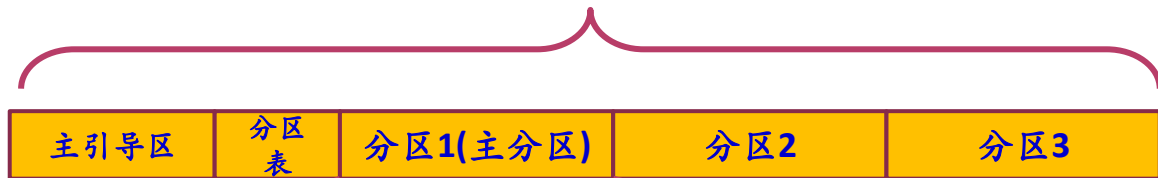
包括该卷（分区）的块（簇）数、块（簇）大小、空闲块（簇）数量和指针、空闲FCB数量和指针.....

- ◎ 目录文件(根目录文件及其他目录文件)

- ◎ 用户文件

# 磁盘上文件系统的布局

## 整块磁盘



## UNIX文件系统布局



## 文件卷(逻辑分区)



## Windows的FAT文件系统布局

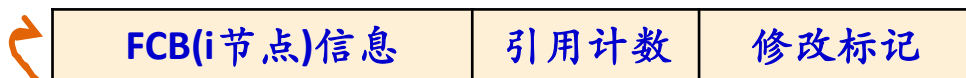


# 内存中所需的数据结构

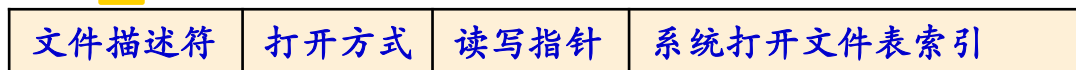
## ——以UNIX为例

---

- 系统打开文件表
  - 整个系统一张
  - 放在内存：用于保存已打开文件的FCB



- 用户打开文件表
  - 每个进程一个
  - 进程的PCB中记录了用户打开文件表的位置

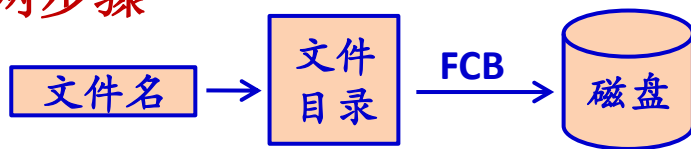


# 文件系统实例——UNIX

# 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 → 按文件名查找到目录项/FCB

根据路径名检索：

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，计算出文件中任意记录或字符在存储介质上的地址

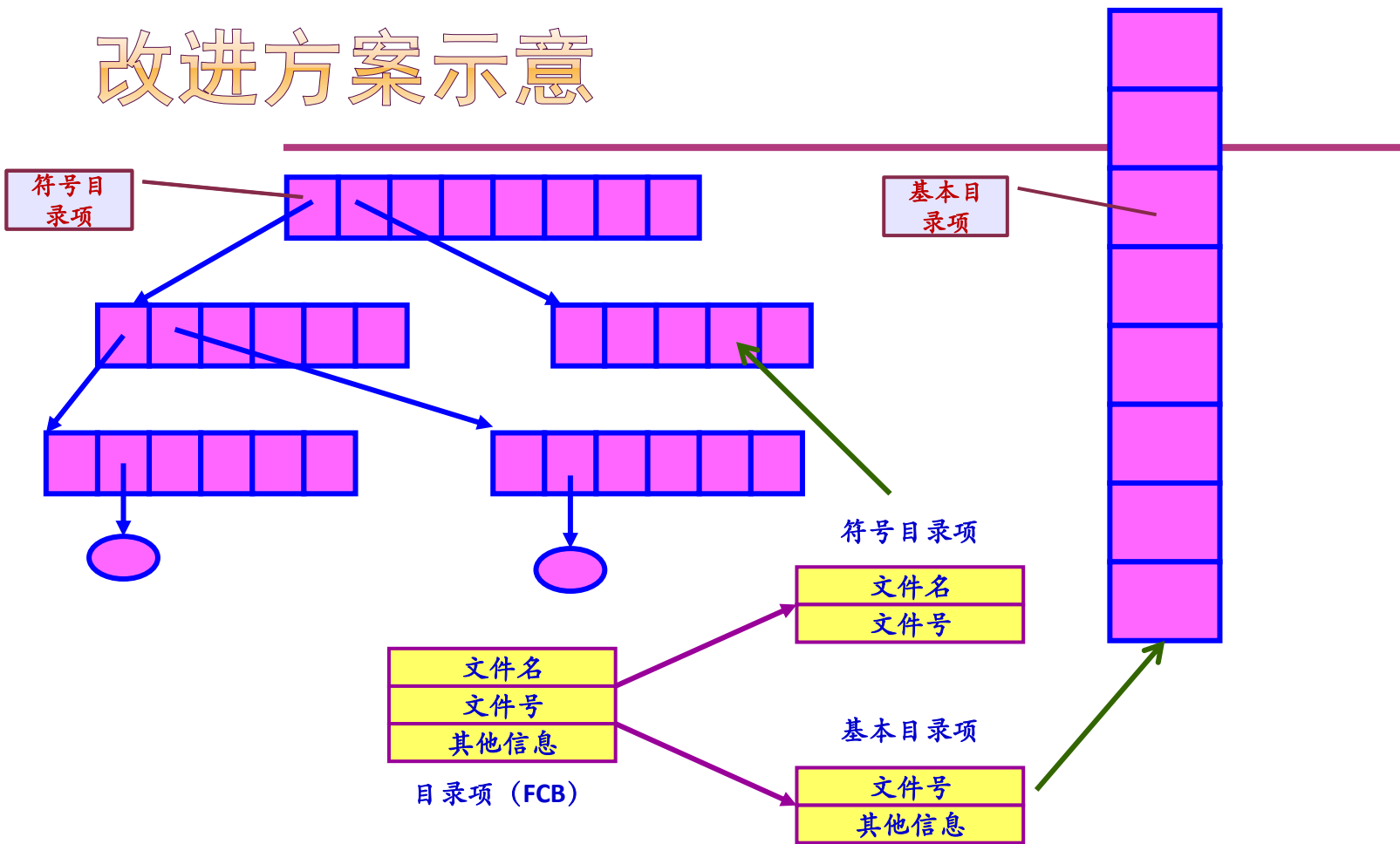
# 目录文件实现时的改进

---

- ◎ 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）

# 改进方案示意



# 改进后的好处

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 占 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

一个目录文件有128个目录项

查找一个文件的平均访盘次数：

分解前：7次

分解后：2.5次

目录文件改进后减少了访盘次数，提高了文件检索速度

# UNIX文件系统(1/3)

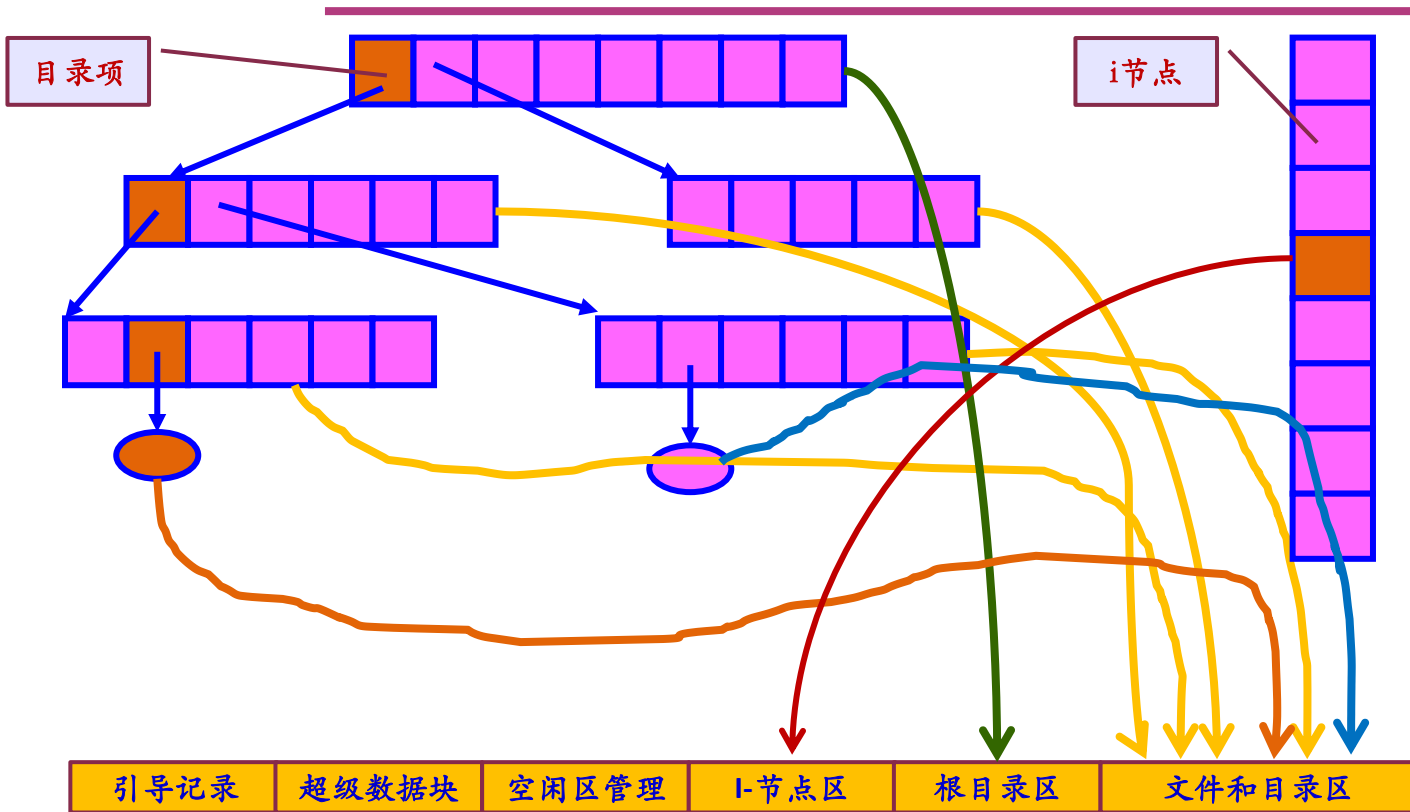
---

- ◎ **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
------	-------	-------	-------	------	--------

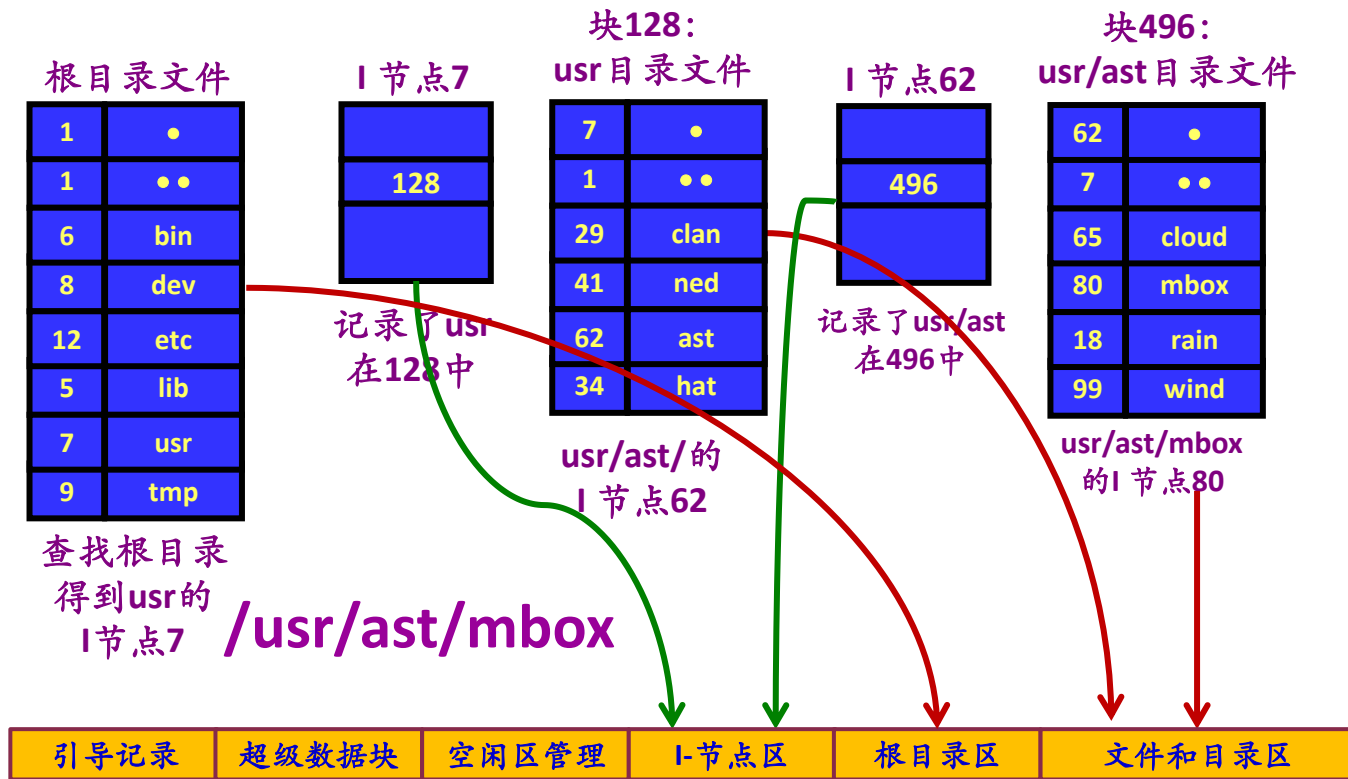
# UNIX文件系统(2/3)





# UNIX文件系统(3/3)

UNIX的I节点  
参考教材  
4.5.3



# 本讲重点

---

- 掌握文件系统涉及的相关概念  
文件、文件分类、文件的逻辑结构和物理结构
- 掌握文件目录的实现  
文件控制块、目录项、目录文件
- 掌握文件系统的实现  
磁盘布局、内存数据结构
- 了解磁盘空间的管理  
存储介质、扇区、物理块、簇

# 本周要求

---

- 重点阅读教材

第4章相关内容：4.1、4.2、4.3、4.5.3

- 重点概念

文件 文件系统 UNIX文件分类 文件逻辑结构  
存储介质(磁盘) 物理块(块、簇) 文件物理结构  
文件控制块FCB(文件属性) 文件目录 目录项 目录文件  
磁盘上文件系统的布局 内存中数据结构  
磁盘空间管理(位图、空闲块表、成组链接法)  
目录项分解法

*THANKS*

*The End*