Computer

操作系统原理与实践

第五章 文件系统

高等教育出版社

第五章　文件系统

- 目的与要求：掌握文件、文件系统、目录的定义，掌握文件系统的实现、文件的分配及空闲空间的管理等关键技术。了解具体操作系统的文件系统的实现原理。
- 重点与难点：文件系统、目录的定义,文件的分配，磁盘空闲空间的管理。
- 作业：　2、3、4、5。

主要内容

5.1 概述

5.2 文件

5.3 文件的访问

5.4 文件保护

5.5 目录结构

5.6 文件系统的实现

5.7 空闲空间管理

5.8 性能和可靠性

5.9 文件系统实例

5.1 概述

文件是操作系统中最重要的概念之一。它是操作系统对底层硬件（即存储介质）最典型的抽象之一。有了文件以后，人们总是考虑数据保存在哪一个文件中，而不是某个存储介质的具体位置。

本章首先介绍文件系统中一些基本概念，接着再着重介绍文件系统的实现，最后结合两个具体的例子（**FAT**和**ext2**）来讲解文件系统的具体实现。

5.2 文件

5.2.1 文件的概念

5.2.2 文件的属性

5.2.3 文件的操作

5.2.4 文件的类型

5.2.5 文件的结构

5.2 文件

5.2.1文件的概念

- 文件是一组或者多组相关信息的集合。
- 对于普通用户来说，文件是最小的数据存储单元。文件中保存的内容可以是数值、文本或者二进制的机器代码。有些文件没有特定的格式，例如常用的ASCII文本文件；有些则包含很复杂的格式，例如可执行程序文件。

5.2 文件

5.2.2 文件的属性

- 名称
- 类型
- 大小
- 保护
- 拥有者
- 时间信息

5.2 文件

5.2.3 文件的操作

文件本质上是一种抽象数据类型。除了文件属性以外，我们还需要定义对文件各种允许的操作。

- 创建文件
- 打开文件
- 关闭文件
- 读取文件
- 写入文件
- 删除文件
- 属性修改

5.2 文件

5.2.4 文件类型

文件类型	常用扩展名	意义
可执行文件	exe, bin, com或无扩展名	包含可以在系统上执行的机器代码，一般由编译器生成
源程序文件	c, cpp, asm	一般由程序员编写的源代码，不同的编程语言使用的扩展名也不相同
目标文件	obj, o	已经完成编译但尚未被链接的机器代码，一般由编译器生成
库文件	lib, a, dll, so	由库函数组成的文件，一般由编译器生成
头文件	h, hpp	C/C++语言程序的头文件，包含宏，常数和函数的定义
文本文件	txt,	包含用户写的文字的文件，由编辑器生成
图像文件	gif, jpg, bmp, png	包含图像的文件，扩展名代表不同的图像编码方式
影片文件	mpg, avi	可连续播放的影片文件，扩展名代表编码格式
流媒体文件	rm, asf, wmf	通过网络播放的影片文件，不需要完整下载整个影片文件
压缩文件	zip, rar, gz, Z	由多个文件或目录经压缩而成，扩展名代表不同的压缩算法
音频文件	wav, mp3, mid, wma, aac	包含音乐数据的文件，扩展名代表不同的声音压缩格式
网页文件	html, htm, xml	包含网页内容和格式的文件

文件分类

- 按文件性质与用途分类
- 按操作保护分类
- 按使用情况分类
- 按用户观点分类(UNIX或Linux操作系统)
- 按存取的物理结构分类
- 按文件中的数据形式分类

1）按性质和用途分类

系统文件

- 由系统软件构成的文件，只允许用户通过系统调用或系统提供的专用命令来执行它们，不允许对其进行读写和修改
- 主要有操作系统核心和各种系统应用程序或实用工具程序和数据组成
- 例如：

  ibmbio.com, ibmdos.com, \comand.com, /unix

1）按性质和用途分类

库文件

- 文件允许用户对其进行读取和执行，但不允许对其进行修改
- 主要由各种标准子程序库组成
- 例如：C语言、FORTRAN子程序库存放在子目录下 *.LIB,/lib/,/usr/lib/

1）按性质和用途分类

用户文件

- 是用户通过操作系统保存的用户文件，由文件的所有者或所有者授权的用户才能使用
- 主要由用户的源程序源代码、可执行目标程序的文件和用户数据库数据等组成
- 例如：*.c,*.for,*.f,*DBF,*.OBJ

2）按操作保护分类

只读文件

- 只允许文件主及被核准的用户去读文件，而不允许写文件。标记为：-r------

可读可写文件

- 允许文件主及被核准的用户去读和写文件。标记为： -rw-----

可执行文件

- 允许文件主及被核准的用户去调用执行该文件而不允许读和写文件，标记为： ---x---

2）按操作保护分类

各个操作系统的保护方法和级别有所不同

- DOS操作系统三种保护：系统、隐藏、可写
- UNIX或Linux操作系统有九个级别的保护

3）按使用情况分类

临时文件

- 用于系统在工作过程中产生的中间文件，一般有暂存的目录，正常工作情况下，工作完毕会自动删除，一旦有异常情况往往会残留不少临时文件

永久文件

- 指一般受系统管理的各种系统和用户文件，经过安装或编辑、编译生成的文件，存放在软盘、硬盘或光盘等外存上

档案文件

- 系统或一些实用工具软件包在工作过程中记录在案的文挡资料文件，以便查阅历史挡案

4）按用户观点分类

普通文件(常规文件)

- 是指系统中最一般组织格式的文件，一般是字符流组成的无结构文件

目录文件

- 是由文件的目录信息构成的特殊文件，操作系统将目录也做成文件，便于统一管理

4）按用户观点分类

特殊文件（设备文件）

- 在UNIX或Linux操作系统中，所有的输入输出外部设备都被看作特殊文件便于统一管理

- 操作系统会把对特殊文件的操作转接指向相应的设备操作，真正的设备驱动程序不包含在这特殊文件中，而是指向与链接到操作系统核心中存放在内存高端部分

5）按存取的物理结构分类

顺序（连续）文件

- 文件中的纪录，顺序地存储到连续的物理盘块中，顺序文件中所记录的次序，与它们存储在物理介质上存放的次序是一致的

链接文件

- 文件中的纪录可存储在并不相邻接的各个物理块中，通过物理块中的链接指针组成一个链表管理，形成一个完整的文件，又称指针串连文件或直接存取文件

5）按存取的物理结构分类

索引文件

- 文件中的纪录可存储在并不相邻接的各个物理块中，记录和物理块之间通过索引表项按关键字存取文件，通过物理块中的索引表管理，形成一个完整的文件

6）按文件的逻辑存储结构分类

有结构文件

- 由若干个记录所构成的文件，故又称为记录式文件

无结构文件

- 这是直接由字符序列所构成的文件，故又祢为流式文件

7）按文件中的数据形式分类

源文件

- 由源程序和数据构成的文件
- 一般是由美国信息交换标准码（ASCII）、EBCD码或汉字编码组成

目标文件

- 由源程序经过相应的计算机语言编译程序编译，但尚未经过链接程序链接的目标代码所形成的文件
- 后缀名为“.OBJ”（DOS系统）或“.o”（UNIX或Linux操作系统）

5.3 文件的访问

- 顺序访问
- 随机访问
- 索引访问

文件的逻辑结构

1. 顺序文件

- 定长记录：

  读指针rptr——指向下一次读出的记录地址；

  写指针wptr——指向下一次写入的记录地址。

  读完指针做相应修改:rptr+m=>rptr

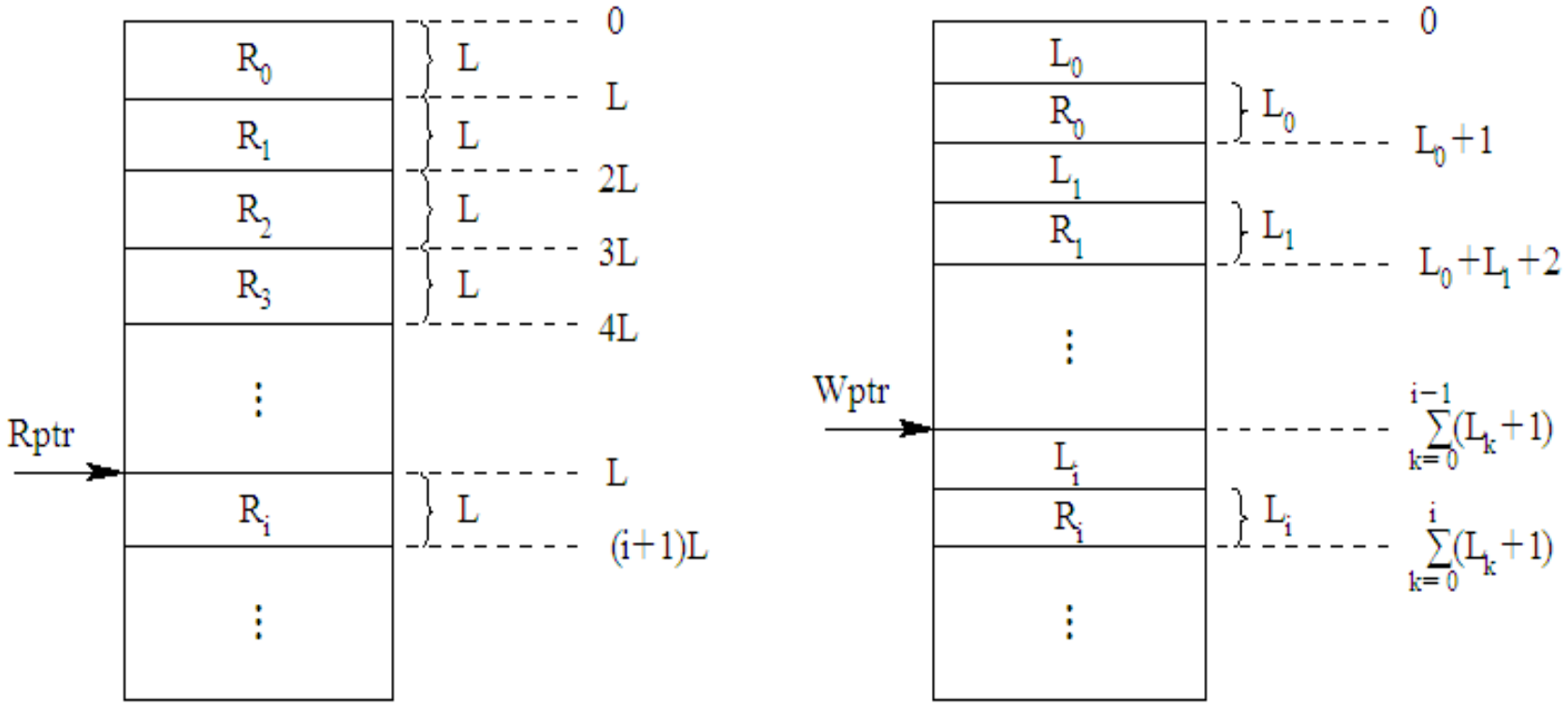
  写完指针做相应修改:wptr+m=>wptr

- 变长记录：

  每个记录长度存于记录前的单元中。

  读完rptr+mi+1=>rptr

2. 对顺序文件的读/写操作



(a) 定长记录文件　　(b) 变长记录文件

定长和变长记录文件

3. 顺序文件的优缺点

- 顺序文件的最佳应用是对记录进行批量存取时，即每次要读或写一大批记录时，对顺序文件的存取效率是所有逻辑文件中最高的；此外，也只有顺序文件才能存储在磁带上，并能有效地工作。
- 在交互应用的场合，如果用户要求查找或修改单个记录，系统要逐个地查找诸记录。这时，顺序文件所表现出来的性能就可能很差，尤其是当文件较大时，情况更为严重。
- 增加或删除一个记录较困难。

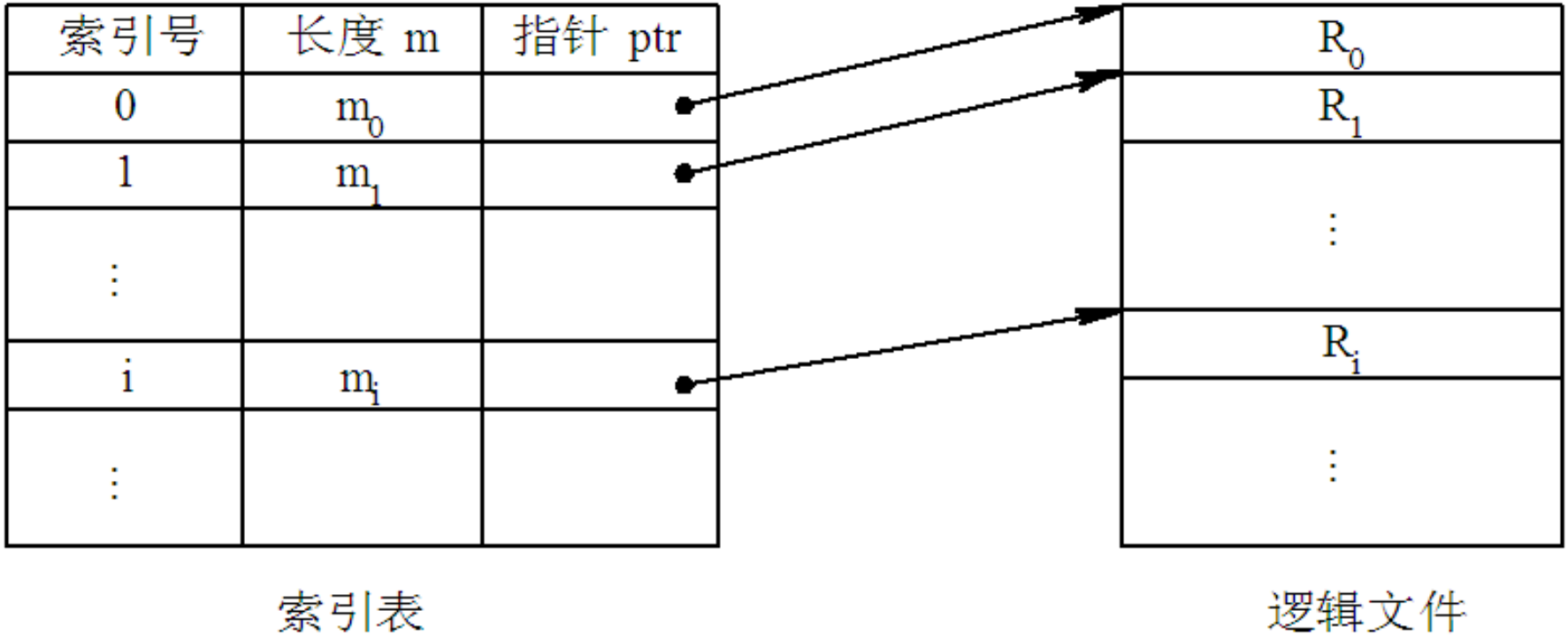
索引文件

对于定长记录文件，如果要查找第i个记录，可直接根据下式计算来获得第i个记录相对于第一个记录首址的地址：　$A_i=i\times L$

对于可变长度记录的文件，要查找其第i个记录时，须首先计算出该记录的首地址。为此，须顺序地查找每个记录，从中获得相应记录的长度L_i，然后才能按下式计算出第i个记录的首址。假定在每个记录前用一个字节指明该记录的长度，则

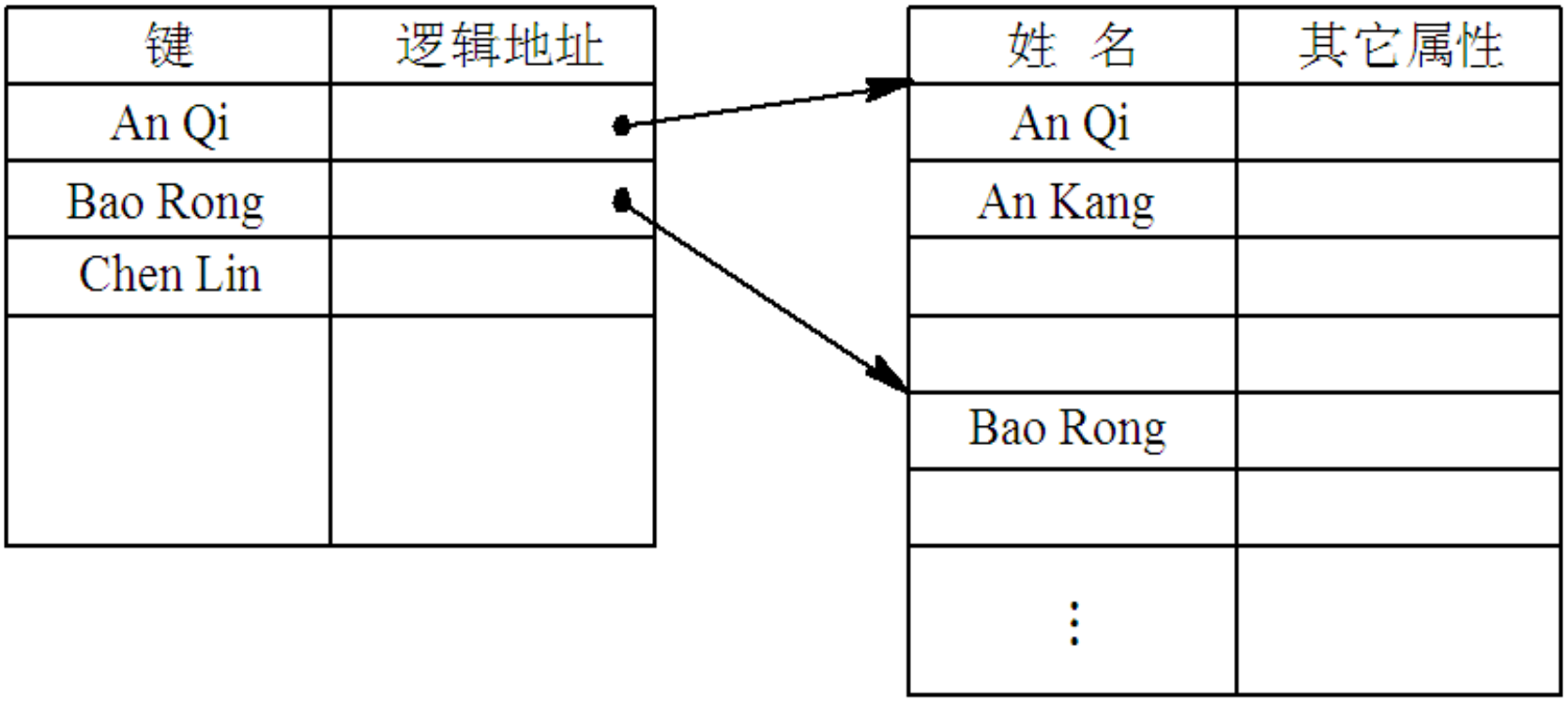
$$A_i=\sum_{i=0}^{i-1}L_i+i$$

索引文件



索引文件的组织

索引顺序文件



索引顺序文件

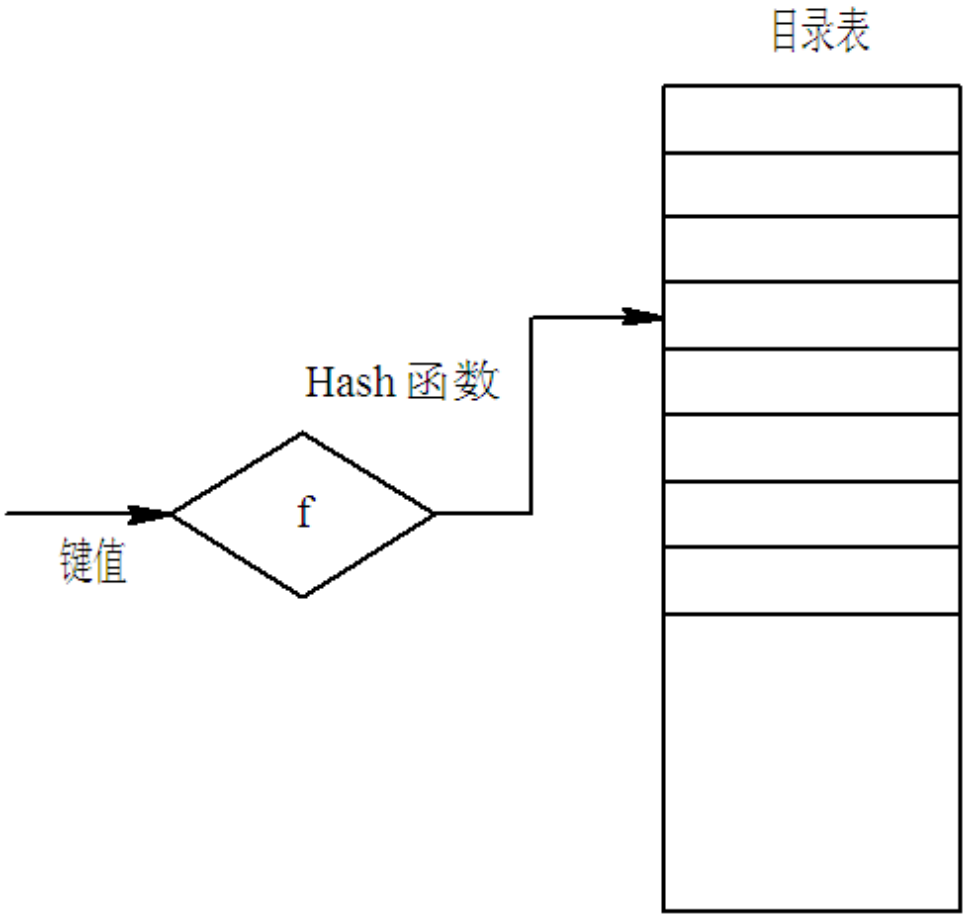
直接文件和哈希文件

1. 直接文件

对于直接文件，则可根据给定的记录键值，直接获得指定记录的物理地址。换言之，记录键值本身就决定了记录的物理地址。这种由记录键值到记录物理地址的转换被称为键值转换。组织直接文件的关键， 在于用什么方法进行从记录值到物理地址的转换。

直接文件和哈希文件

2. 哈希(Hash)文件



Hash文件的逻辑结构

5.4文件保护

1.访问类型

- 读文件
- 写文件
- 执行文件
- 追加文件
- 列表文件
- 删除文件

5.4文件保护

2. 访问控制（控制访问类型，控制用户）

- 基于用户和组的访问控制

文件名	文件拥有者	所属组	其他用户
example.c	**RW**	**RW**	**R**

- 访问控制列表（**Access Control List，ACL**）

ACL所采用的方法是给系统中的每一个文件附加一个访问控制列表信息，这个表中包含了哪些用户对这个文件有哪种类型的访问。这样系统打开这个文件之前，都要搜索访问控制列表来判断当前用户是否允许执行相应的操作。

5.5文件目录

- 文件目录：是文件系统中主要数据结构之一，通常通过文件目录实现文件的组织。

- 对目录管理的要求如下：

(1)实现“按名存取”

(2) 提高对目录的检索速度

(3) 文件共享

(4) 允许文件重名

文件控制块(FCB)

- 文件控制块是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相关信息。
- 文件控制块是文件存在的标志

文件控制块的内容

(1)基本信息类

① 文件名；　　② 文件物理位置；

③ 文件逻辑结构；　④ 文件的物理结构

(2) 存取控制信息类

(3) 使用信息类

文件名	扩展名	属性	备用	时间	日期	第一块号	盘块数

MS-DOS的文件控制块

2. 文件控制块包括的内容

文件名	文件设备	指向下一个PCB的指针
文件标识符	物理位置	当前共享的状态
文件结构	存取控制	共享访问时等待状态
文件类型	口令	进程访问文件所用的逻辑单元号
记录长度	文件建立时间	当前的逻辑位置
当前文件大小	上次存取时间	访问元素的当前物理位置
缓冲区大小	缓冲区地址	文件创建者
当前存取方式	文件拥有者	临时\永久文件
最大文件尺寸	上次修改时间	下一个元素的物理位置

文件目录

- 把所有的FCB组织在一起，就构成了文件目录，即文件控制块的有序集合
- 目录项：构成文件目录的项目（目录项就是FCB）
- 目录文件：为了实现对文件目录的管理，通常将文件目录以文件的形式保存在外存，这个文件就叫目录文件

索引结点

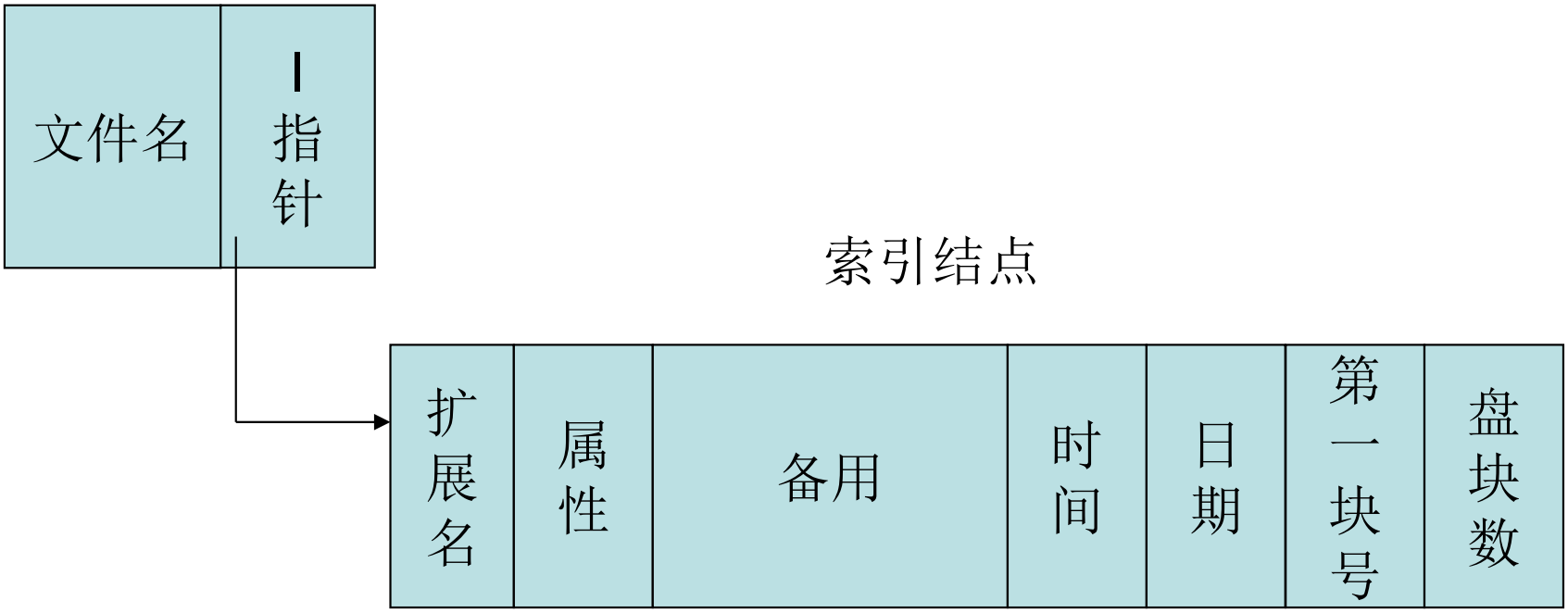
1.索引结点的引入

- 文件目录放在外存上，当检索一个文件时需要启动磁盘，将不同磁盘块中的目录项逐一调入内存。设目录文件占用**N**块盘块，则查找目录项平均调入盘块（**N+1**）**/2**。
- 改进：在查找目录时，目录项中只有文件名是必要的。所以可以采取文件名和其他文件描述信息分开存储的办法。文件描述信息单独形成一个称为索引结点的数据结构，简称i结点。目录项的内容变成文件名和指向该文件所对应的i结点的指针所构成。因此目录文件的存储盘块数大大降低。

索引结点

文件控制块

文件名	扩展名	属性	备用	时间	日期	第一块号	盘块数



索引结点

文件名	索引结点编号
文件名1	
文件名2	
⋮	⋮

UNIX的文件目录

2. 磁盘索引结点

(1) 文件主标识符

(2) 文件类型

(3) 文件存取权限

(4) 文件物理地址

(5) 文件长度

(6) 文件连接计数

(7) 文件存取时间

3. 内存索引结点

文件被打开时，将磁盘索引结点拷贝到内存的索引结点中，内存索引结点增加了以下内容:

(1) 索引结点编号。 用于标识内存索引结点。

(2) 状态。 指示i结点是否上锁或被修改。

(3) 访问计数。 每当有一进程要访问此i结点时，将该访问计数加1， 访问完再减1。

(4) 文件所属文件系统的逻辑设备号。

(5) 链接指针。 设置有分别指向空闲链表和散列队列的指针。

1. 单级目录结构

为所有文件建立一个目录文件。单级目录的优点是简单且能实现目录管理的基本功能—按名存取。

缺点：(1) 查找速度慢 ；(2) 不允许重名

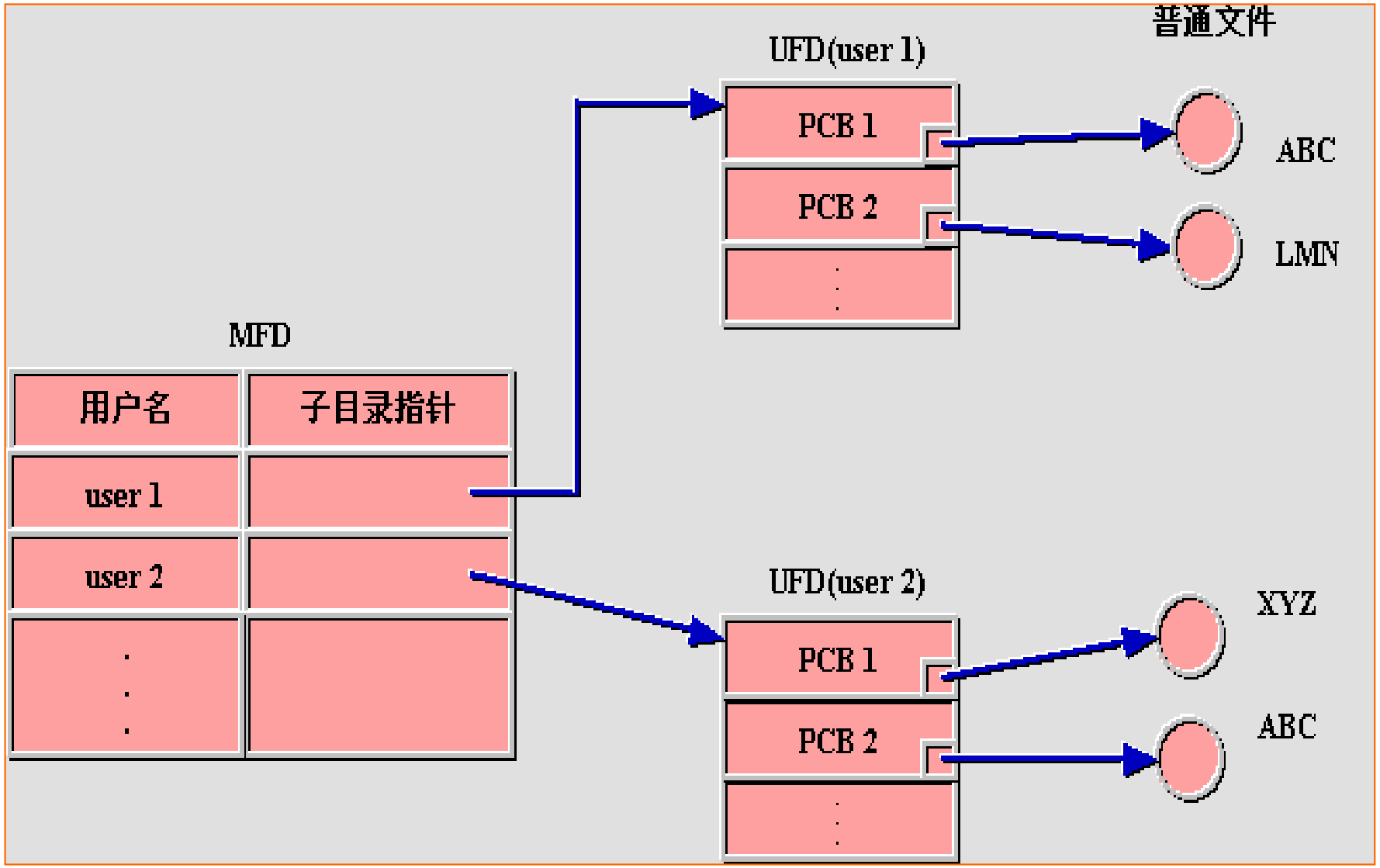
(3) 不便于实现文件共享

文件名	物理地址	文件说明	状态位
文件名1			
文件名2			
⋮			

2. 二级目录结构

- 为改变一级目录文件目录命名冲突，并提高对目录文件检索速度而将目录分为两级：一级称为主文件目录，给出用户名，用户子目录所在的物理位置；二级称为用户文件目录，给出该用户所有文件的FCB
- 产生于多用户分时系统，DOS2.0版本以上采用，文件主目录（MFD）的表目按用户分，每个用户有一个用户文件目录（UFD）
- 优点：解决了文件的重名问题和文件共享问题，提高搜索速度，查找时间降低
- 缺点：缺点是不太适合大量用户和大量文件的大系统，增加了系统开销，

两级目录结构


普通文件
UFD(user 1)
PCB 1
PCB 2
ABC
LMN
MFD
用户名
子目录指针
user 1
user 2
UFD(user 2)
PCB 1
PCB 2
XYZ
ABC

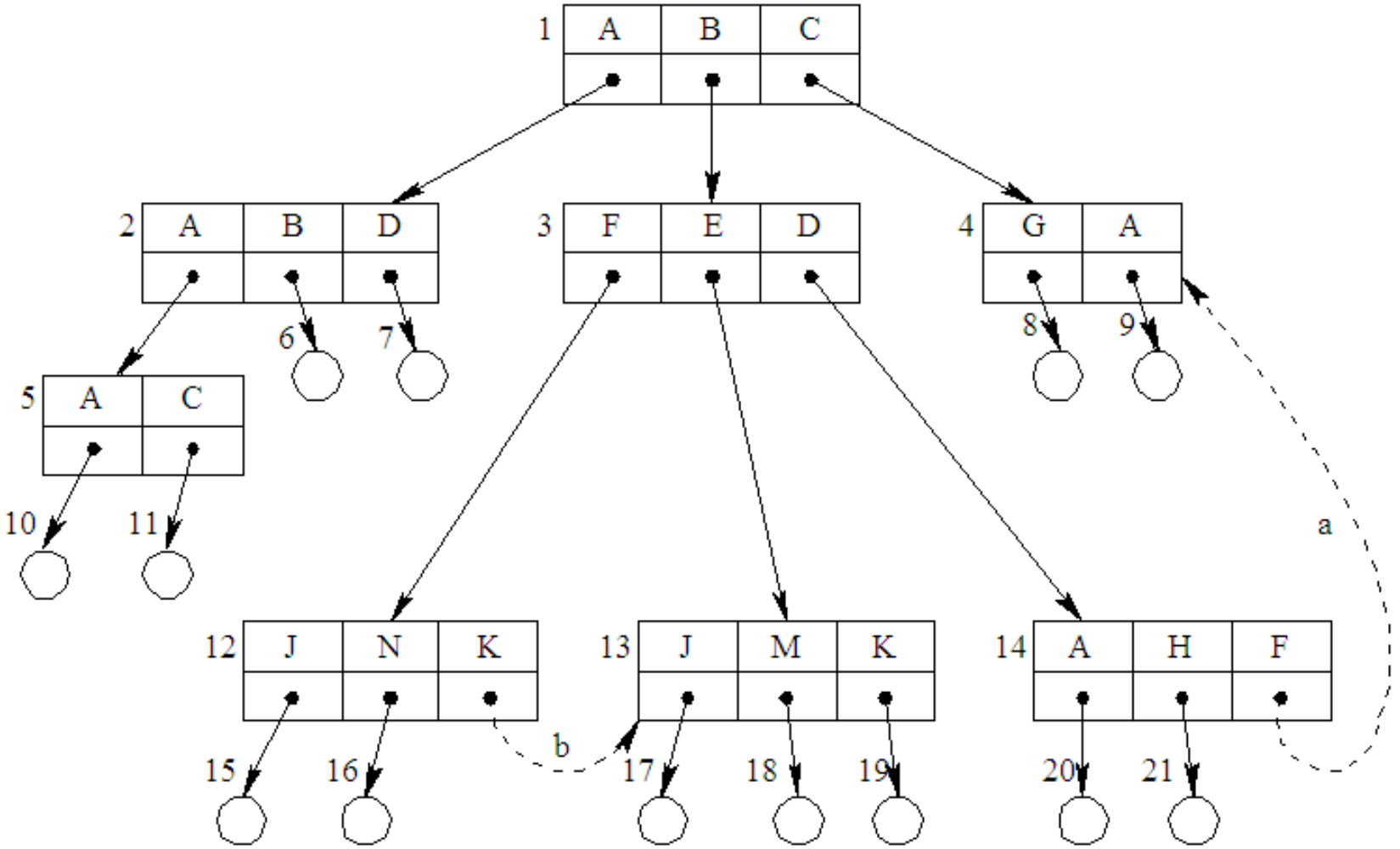
3. 多级目录结构

- 多级目录结构也称树型目录
- 产生于UNIX操作系统，已被现代操作系统广泛采用。目录与文件在一起，目录也做成文件
- 优点：

  层次结构清晰，便于管理和保护；有利于文件分类；解决重名问题；提高文件检索速度；能进行存取权限的控制
- 缺点：

  查找一个文件按路径名逐层检查，由于每个文件都放在外存，多次访问磁盘，影响速度

1）多级目录结构



多级目录结构

2）路径名

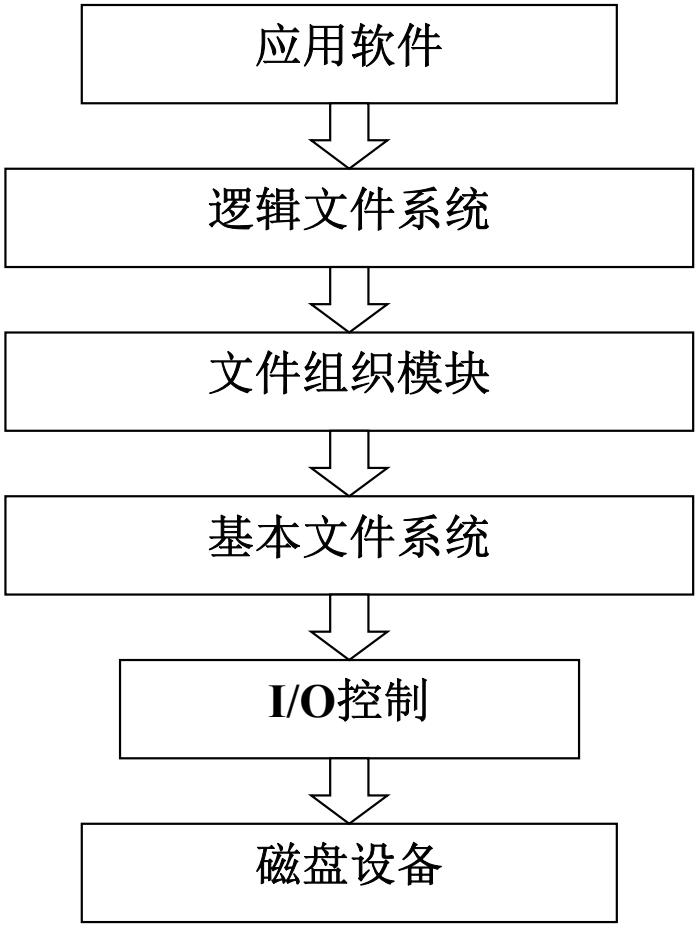
在树形目录结构中，从根目录到任何数据文件，都只有一条惟一的通路。在该路径上从树的根(即主目录)开始，把全部目录文件名与数据文件名，依次地用“/”连接起来，即构成该数据文件的路径名。系统中的每一个文件都有惟一的路径名。例如，在上图中用户B为访问文件J，应使用其路径名/B/F/J来访问。

3）当前目录

- 为了提高文件检索速度，文件系统向用户提供了一个当前正在使用的目录，称为当前目录（也称工作目录或值班目录）。查找一个文件可从当前目录开始，使用部分路径名
- 当前目录可根据需要任意改变。
- 当前目录一般存放在内存

5.6文件系统的实现

5.6.1 文件系统的结构



5.6文件系统的实现

- 应用软件：当应用软件需要对某个文件进行操作时，它通过操作系统的系统调用将文件的路径信息传递进来；
- 逻辑文件系统：逻辑文件系统根据所提供的路径找到指定的文件，进行权限的检查。如果这是一个合法的请求，它提取文件的逻辑位置信息；

5.6文件系统的实现

- 文件组织模块：将逻辑位置信息转换成磁盘上属于该文件的磁盘块的位置；
- 基本文件系统：它负责把磁盘块的位置信息通过指令发送给I/O系统，请求磁盘操作；
- I/O控制：磁盘控制器接收到命令后，发送一系列命令给磁盘设备并等待磁盘操作结束，然后发出中断命令等操作；

5.6文件系统的实现

5.6.3 目录实现

我们可以把目录看成是一张表，其中的每一行记录了这个目录下的某个文件或子目录的名称、逻辑位置等信息。因此，一个基本的实现问题就是，如何组织这张表才能获得最好效率和性能呢？目前，最常用的两种方式是线性表（**Linear List**）和哈希表（**Hash table**）。

5.6文件系统的实现

- 线性表

线性表是最简单的数据结构了。目录中的文件和子目录信息依次顺序存储在这个特殊的“文件”中就形成了这种结构。

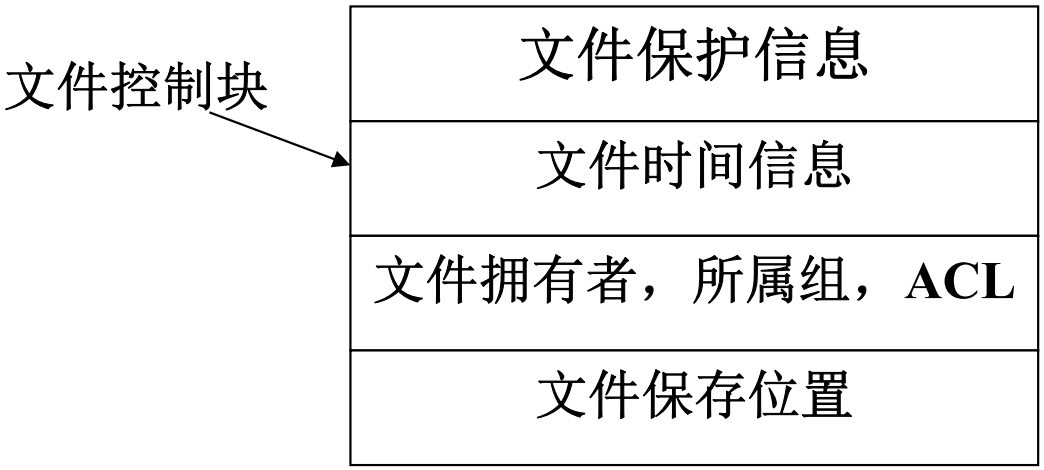
- 哈希表

为了更快地访问目录中的文件，我们可以将目录表项组织成哈希结构，采用文件名作为哈希函数的输入值。

5.6文件系统的实现

5.6.4 文件的实现

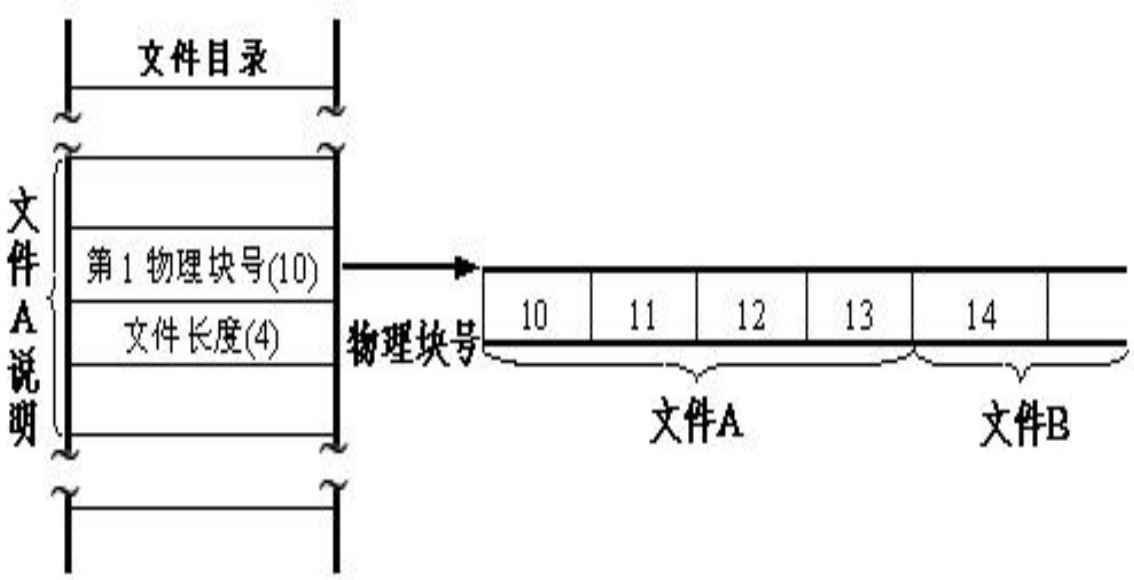
一个文件除了数据本身外，还有很多的属性信息。为了方便描述和实现，一般称文件的所有属性信息为文件控制块（**File Control Block， FCB**）。



5.6.5 文件分配方法

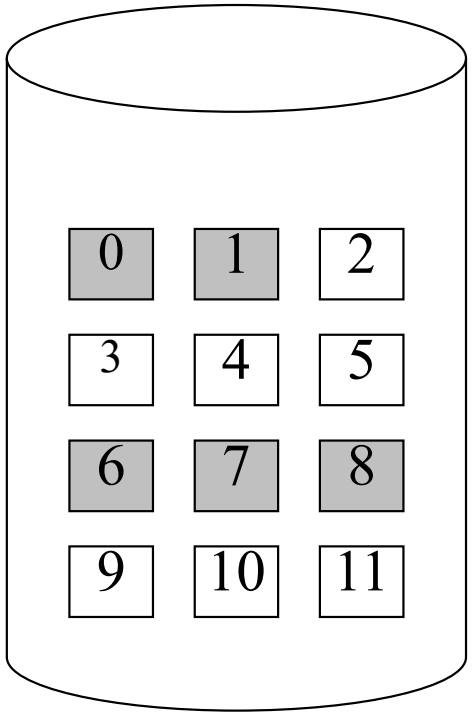
1. 连续分配

- 为每个文件分配一组相邻接的盘块。一组盘块的地址定义了磁盘上的一段线性地址。
- 对于这种方式，文件的逻辑组织与物理组织一致。为使系统能找到文件存放的地址，应在目录项的“文件物理地址”字段存放该文件的第一个记录所在的盘块号和文件长度（盘块数）如图：



1. 连续分配

- 有碎片问题
- 优缺点：
  - 顺序访问容易
  - 顺序访问速度快
  - 碎片
  - 必须知道文件长度



文件	起始位置	大小
foo	**0**	**2**
bar	**6**	**3**

2. 链接分配

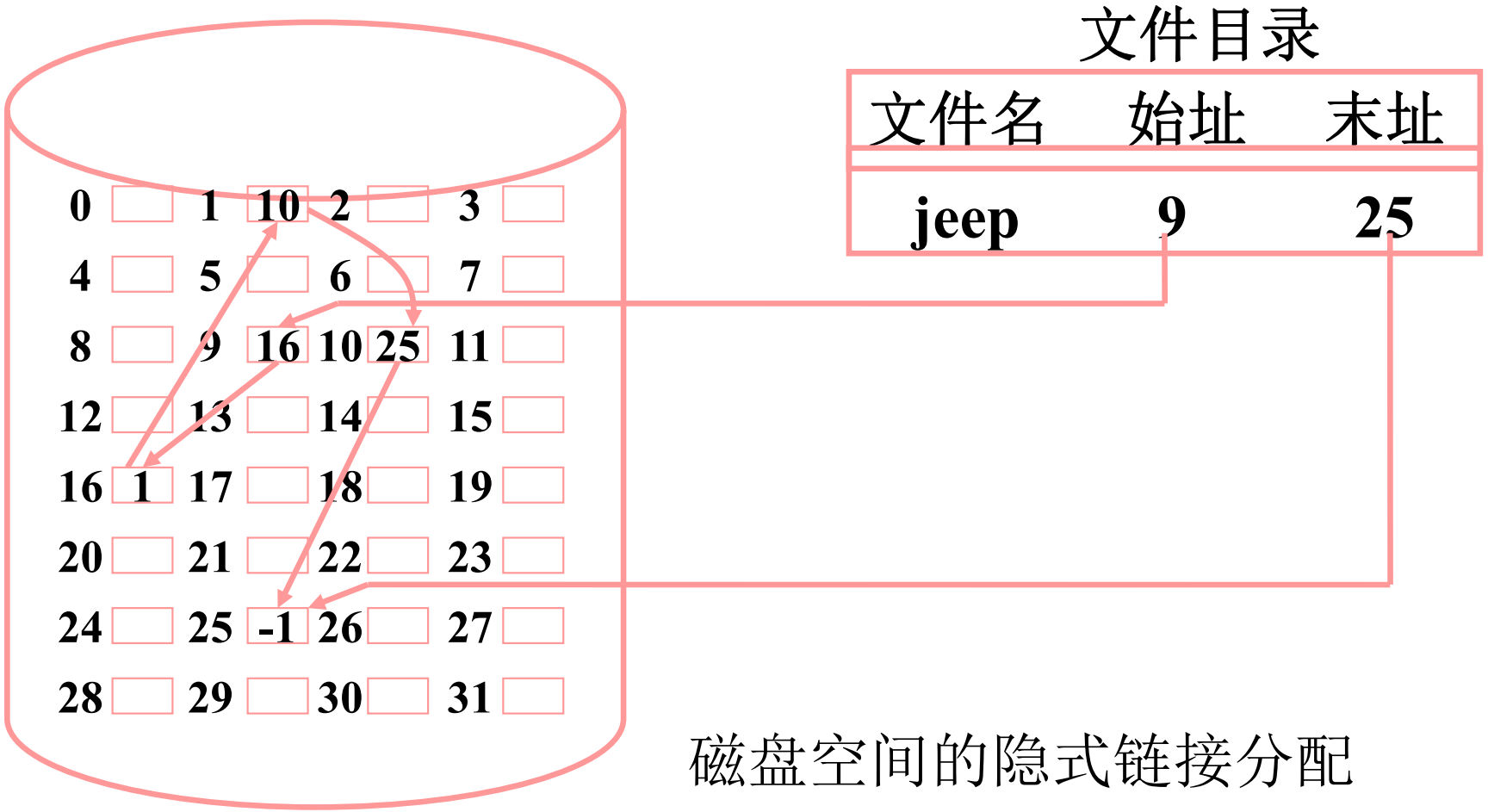
- 一个文件的信息存放在若干不连续的物理块中，各块之间通过指针连接，前一个物理块指向下一个物理块

- 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

- 分为隐式连接和显式连接

2. 链接分配

1） 隐式链接



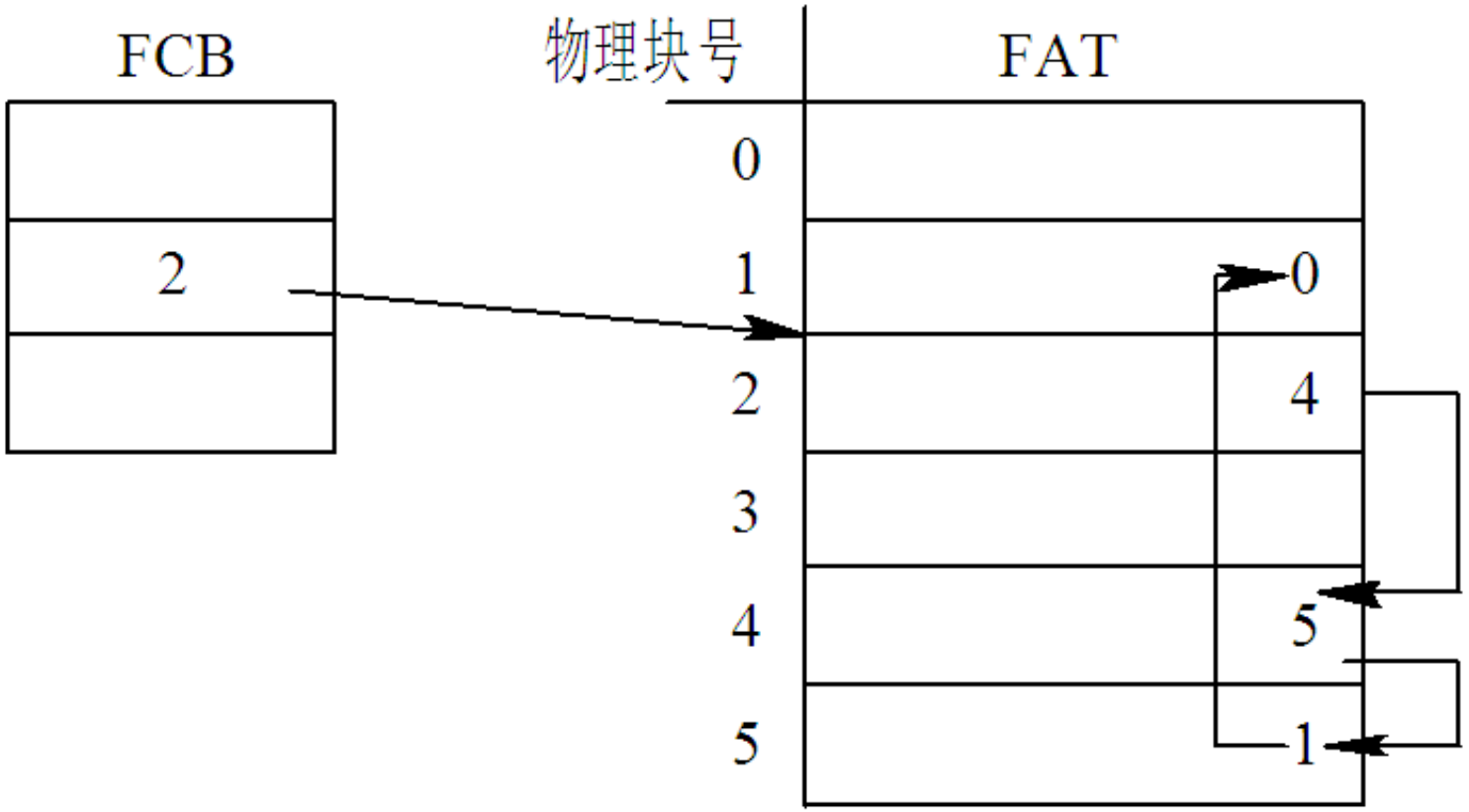
磁盘空间的隐式链接分配

2. 链接分配

- 在文件目录的每个目录项中须含有指向链接文件第一个盘块和最后一个盘块的指针。每个盘块含有指向下一个盘块的指针
- 问题:只适合顺序访问，因为在盘块中含下一个盘块的指针，所以必须启动磁盘。如果随机读取第100块，则要启动磁盘100次。速度很低。改进的方法是“簇”，但改进有限，且增大碎片

2. 链接分配

2）显式链接



显式链接结构

2. 链接分配

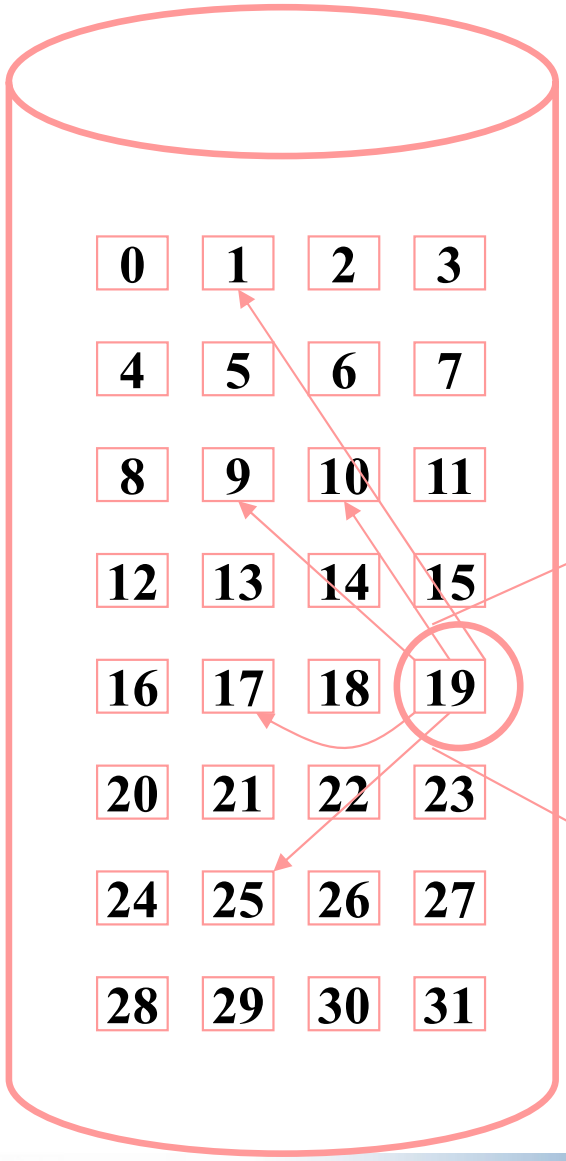
- 计算：**1.2M**软盘，块把用于链接文件各物理块的指针，显式地存放在内存的一张链接表中，在整个磁盘仅设置一张。表的序号是物理盘块号，从**0**开始到**N-1**.每个表项存放链接指针。文件的首个盘块号存放在相应文件的**FCB**中。
- 由于在内存中检索，所以显著提高检索速度。—**FAT**
- 问题：可能占用较大内存
- **200M**硬盘，块**1K**，**FAT**有多大？
- **1.2M**软盘，块**1K**，**FAT**有多大？

3. 索引分配

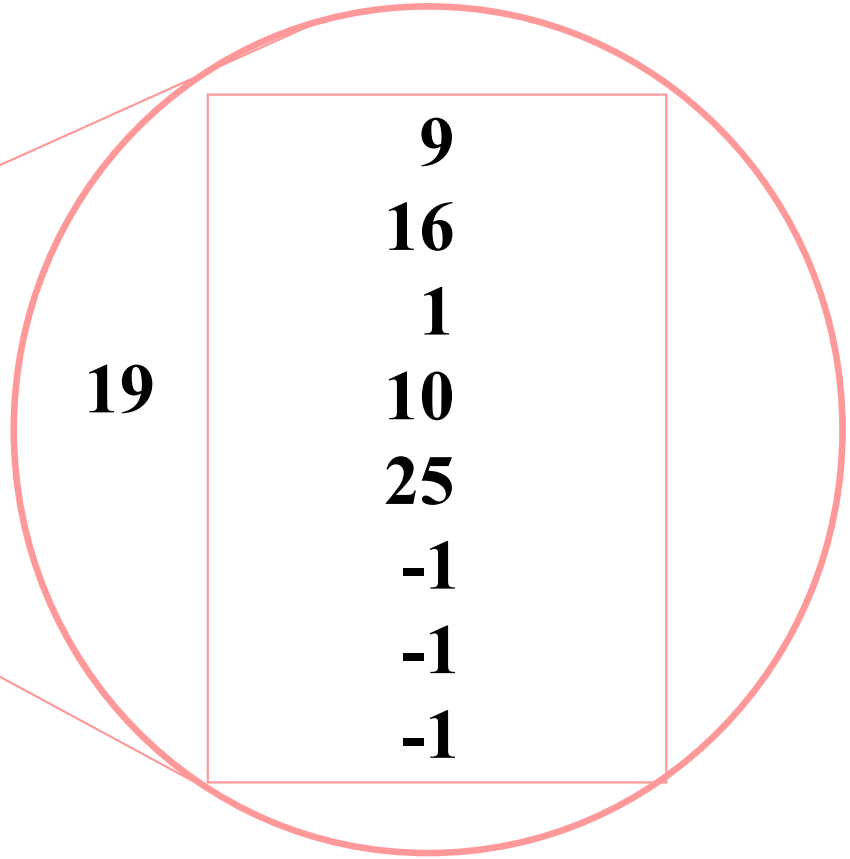
1）单级索引分配

- 链接带来的问题
  - 不能高效直接存取，对较大文件必须首先在FAT中查找许多盘块号
  - FAT占用较大内存
- 解决：不将整个FAT调入内存，将每个文件对应的盘块号，存入文件的索引块，该索引块就是一个含有许多盘块号的数组。
- 问题：花费较多外存，小文件索引块利用率极低。

文件目录



文件名	索引表地址
Jeep	19

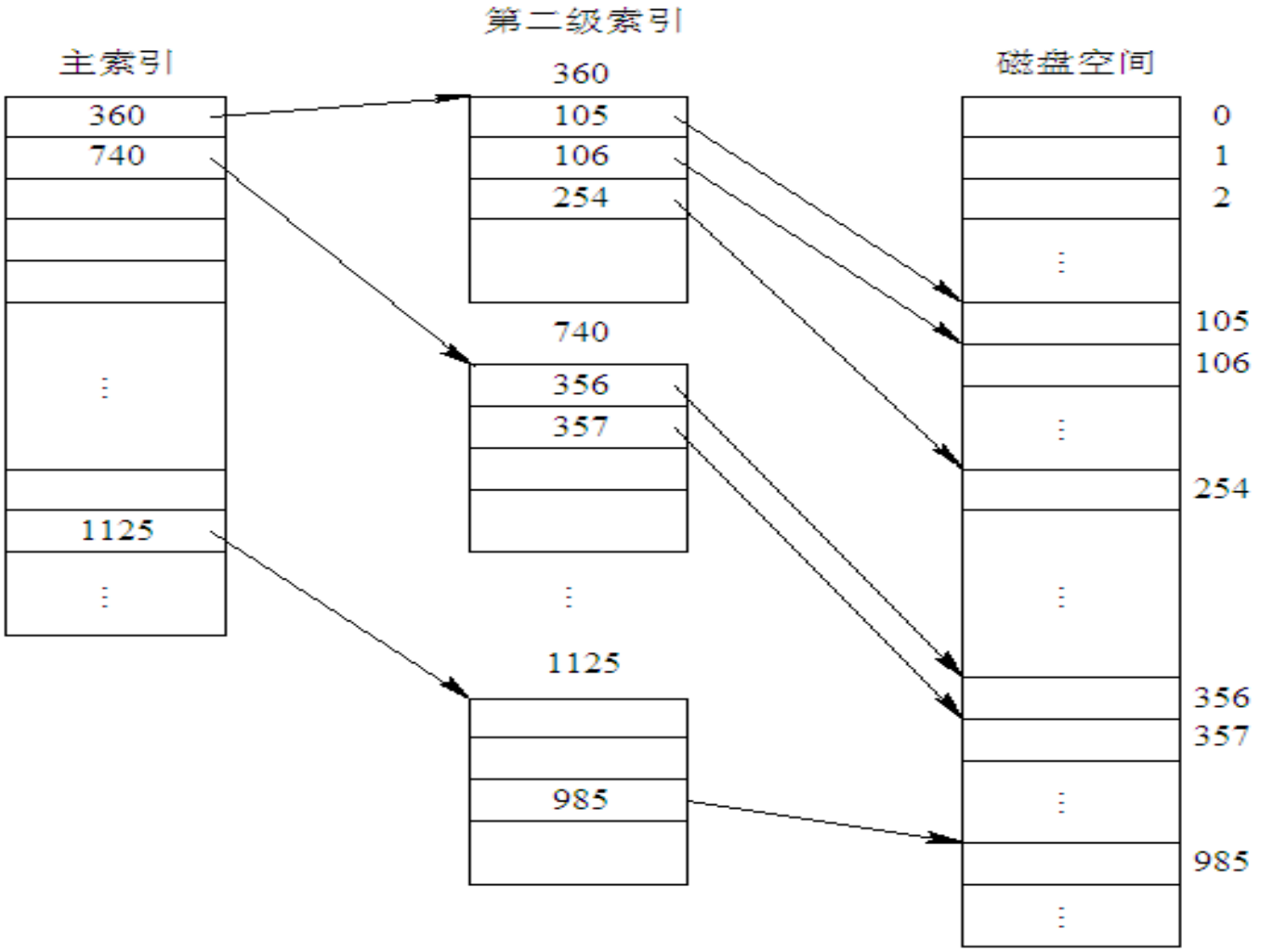


3. 索引分配

2）多级索引分配

- 如果文件比较大，一个索引块不足以存储所有盘块号，则需要多个索引块，如果为这些索引块建立索引，变形成了两级索引分配方式。类似的，更大的文件可以使用三级、四级索引方式

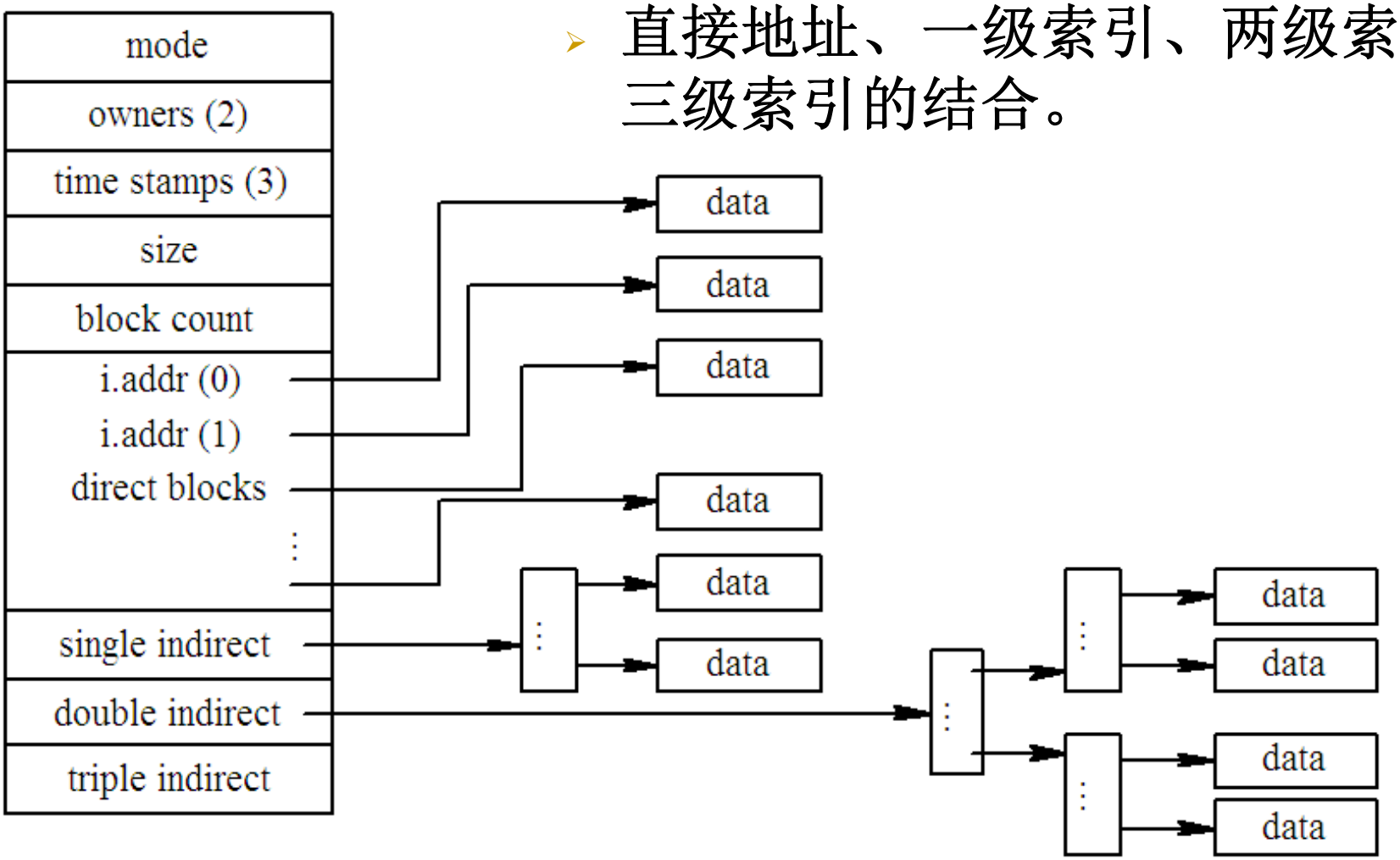
3.索引分配


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

3. 索引分配

3 ）unix的混合索引分配方式

- 直接地址、一级索引、两级索引、三级索引的结合。



2010年研究生入学试题

- 设文件索引结点中有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，这可表示的单个文件最大长度是

A.33KB　　B.519KB

C.1057KB　　D.16513KB

5.7 空闲空间管理

5.7.1 位示图

- 对于像硬盘这种把存储空间分成一个个存储块的设备而言，我们可以为每一个存储块用一个位来记录它目前是否空闲。把所有的这些标志位组织在一起就形成了所谓的位示图。

位示图法

- 建立一张位示图，以反映整个存贮空间的分配情况，每一个字的每一位都对应一个物理块。

0：未分配(空白块)

1：已分配

位示图法

字 \ 位	0	1	2	3	……	……	14	15
0	1	1	0	0	……	……	0	0
1	0	1	0	0	……	……	1	0
2	……							
…	…							

位示图法

为了找到N个自由块，就需要搜索位示图，找到N个“ 0”位，再经过一个简单的换算就可得知相应的块地址(物理地址)，存贮空间的分配和回收工作较为方便，{0,1}转换。

位示图法

- 盘块的分配
  - （1）顺序扫描找到一个或一组值为“0”的二进制位
  - （2）转换成相应盘块号
    - B=n×i+j
  - （3）修改位示图map[I,j]=1

位示图法

- 盘块的回收
  - （1）将回收盘块的盘块号转换为位示图的行号和列号
    - **i=bDIV n**
    - **J=b MOD n**
  - （2）修改位示图令**map[I,j]=0**

5.7 空闲空间管理

5.7.2 链表

- 这种方法是将所有的空闲存储空间通过链表串起来，组成空闲空间链表。链表的表头指针保存在文件系统中预先设定的某个位置，并用一个特殊的指针值来作为链表的结束标志。

链　表

（1）空闲盘块链

- 将磁盘上的所有空闲空间，以盘块为单位拉成一条链。当创造文件需要一块或几块时，就从链头依次取下一块或几块。反之，当回收空间时，把这些空白块依次链到链尾上。这种技术只要在主存中保存一个指针，令它指向第一个空白块。

（2）空闲盘区链

- 将磁盘上的所有空闲盘区拉成一条链。每个盘区上含有本盘区大小的信息。分配采用首次适应算法，回收考虑邻接空闲区的合并。

5.7 空闲空间管理

5.7.3 分组计数

- 分组计数是对链表方法的一种改进。首先，保存开始的n个存储块的地址于第一个空闲块中，其中前n-1个的确指向空闲块，最后一个指向另一个保存n个空闲块地址的空闲块，依次如此。

成组链接法

1. 空闲盘块的组织

(1)空闲盘块号栈

存放当前可用的一组空闲盘块的盘块号（最多100个）。及栈中尚有的空闲盘块号数N。N还兼作栈顶指针用。N=100，指向s.free(99)。s.free(0)是栈底，栈满时栈顶为s.free(99)。

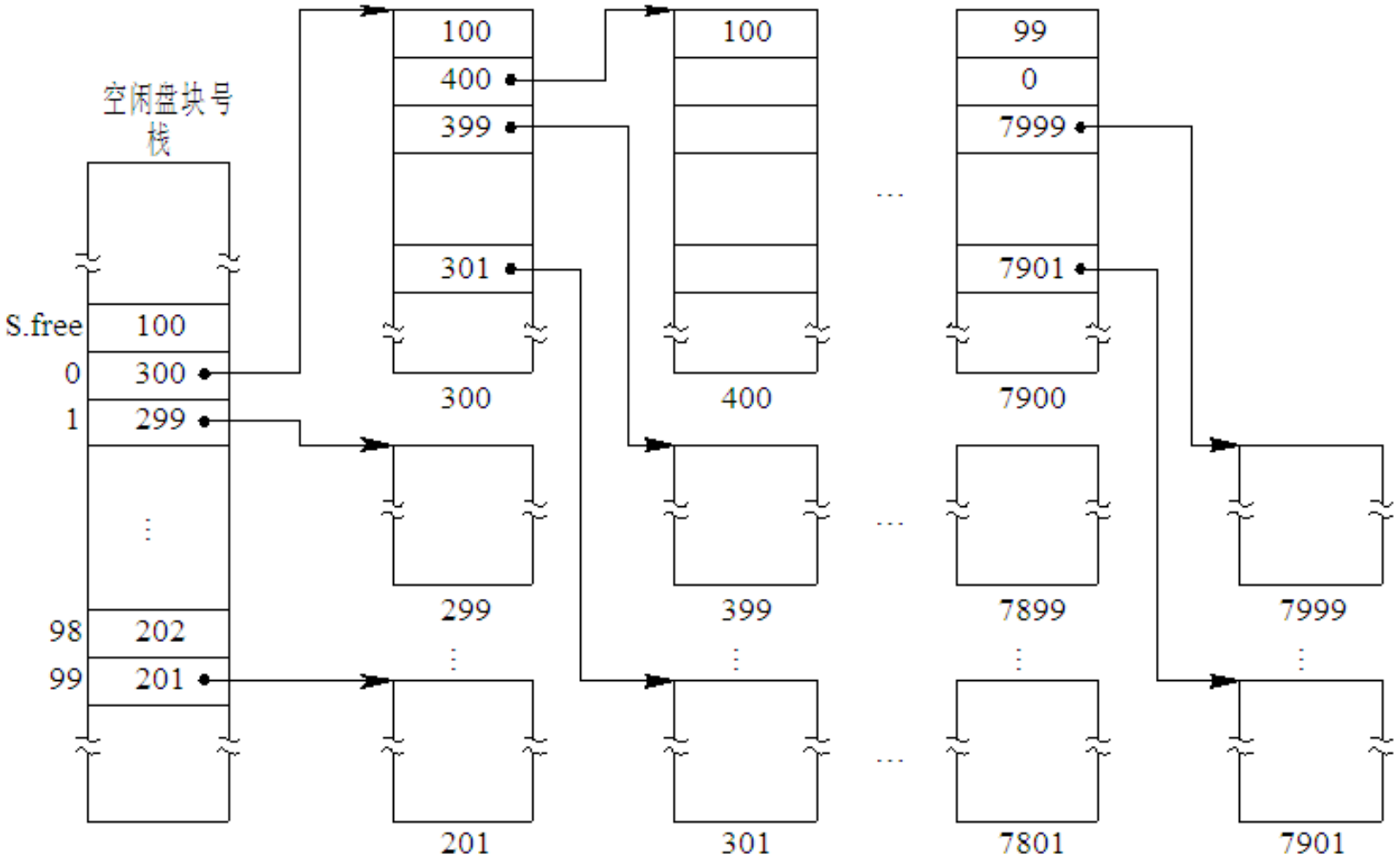
(2)文件空闲盘块被分成若干组

成组链接法

(3)将一组含有盘块总数N和该组所有盘块号，记入前一组的第一个盘块的s.free(0)~ s.free(99),这样各组的第一个盘块可以链成链。

(4)将第一组的盘块总数和所有盘块号记入空闲盘块号栈，作为当前克分配的空闲盘块号。

成组链接法


空闲盘块号
栈
S.free
100
0
300
1
299
98
202
99
201
100
400
399
301
300
299
201
100
399
301
400
99
0
7999
7901
7900
7899
7801
7999
7901

成组链接法

2．空闲盘块的分配和回收

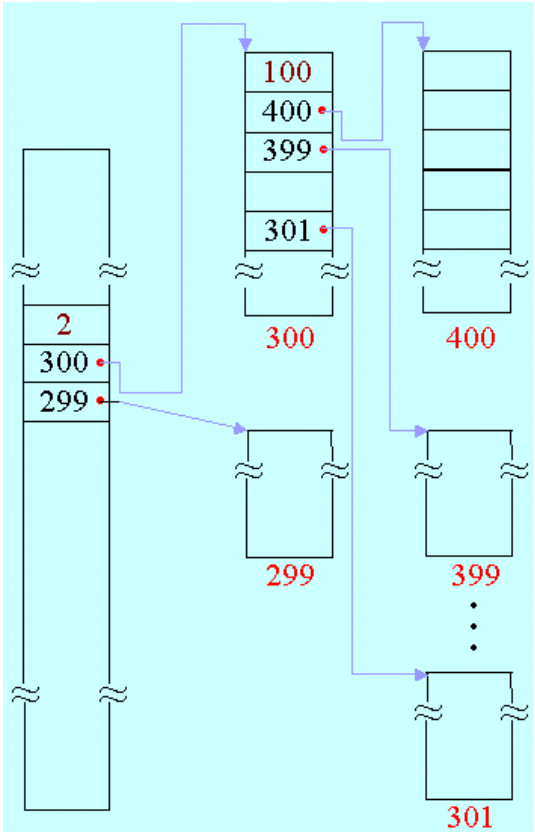
- 分配：首先检查空闲盘块号栈是否上锁，如未上锁从中取出一空闲盘块号，将对应盘块分配给用户，将栈顶下移。若已是栈底，s.free(0)，则把新的盘块号栈内容读入，在分配该盘块。最后把栈中的空闲盘块数减1并返回。

成组链接法

- 回收:将回收盘块的盘块号记入空闲盘块号栈的顶部，并执行空闲盘块号数加1的操作。当栈中空闲盘块号数已达100时，将现有栈中的100各盘块号记入新回收的盘块中，再将其盘块号作为新栈底。

成组链接法

- 文件系统采用成组链接法描述空闲盘块，对下图所示的情况，若分配4个盘块后又回收了700、750、812、505、522号盘块，请画出新的成组链接图。



5.8性能和可靠性

5.8.1 文件系统的性能

- 高速缓存
- 基于软件的系统缓存模块
- 内存磁盘（RAM disk）

5.8.2 文件系统的可靠性

- 备份
  - 周期性全部备份
  - 差异备份

5.9 文件系统实例

5.9.1 FAT文件系统

- FAT是微软公司最初为它的操纵系统DOS开发的一种文件系统，它有FAT-12，FAT-16和FAT-32三个版本。FAT-12适用于小容量设备，如软盘；FAT-16对FAT-12做了简单的扩充，用在Windows的早期版本；而FAT-32是Windows 95 OSR2（OEM Service Release 2）中主要的文件系统，开始支持大容量硬盘。为了保持向后兼容，Windows NT/XP仍然支持FAT文件系统。

5.9 文件系统实例

- FAT是File Allocation Table的简称，它采用链表作为文件分配结构。为了提高性能和可靠性，它把所有的指针收集到一张表中统一保存，称这张表为文件分配表（FAT）。文件分配表位于分区的开始，因为它是如此重要的数据结构，为了防止文件系统遭到破坏，FAT文件系统保存了两个文件分配表，它们互为备份。下图给出了FAT文件系统的总体结构：

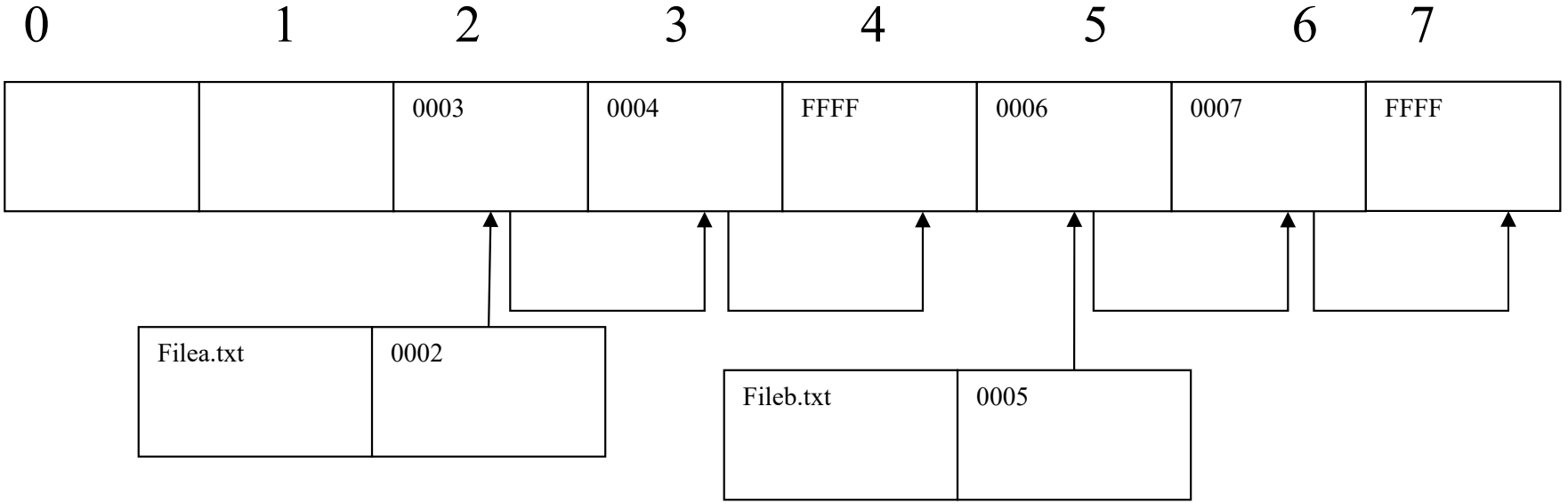
引导块	FAT（1）	FAT（2）	根目录	其他数据

FAT文件系统结构

5.9 文件系统实例

1．引导块中主要包含用于描述分区的各种信息，包括逻辑块的大小、文件分配表的位置等等。

2．在FAT文件系统的目录项中，记录了文件名和属于该文件的起始逻辑块编号。



文件分配表结构

5.9 文件系统实例

3．FAT根目录中保存了该分区下面根目录下的文件或子目录信息。它和其他子目录的唯一区别是它被保存在分区上一个确定的位置，并有固定的大小。在FAT中，文件的FCB被保存在它所在的目录中，每一个FCB占用32字节。

4．在文件分配表中，一般用FFFF来标识文件的结束。如果某个逻辑块未被分配，即空闲块，FAT用0000来表示。因此，系统必须搜索文件分配表来查找空闲块。

5.9 文件系统实例

- FAT32采用32位的文件分配表，使其对磁盘的管理能力大大增强，突破了FAT16对每一个分区的容量只有2 GB的限制。由于现在的硬盘生产成本下降，其容量越来越大，运用FAT32的分区格式后，我们可以将一个大硬盘定义成一个分区而不必分为几个分区使用，大大方便了对磁盘的管理。

5.9 文件系统实例

- FAT32具有一个最大的优点：它可以大大地节约磁盘空间。文件在磁盘上是以簇的方式存放的，簇里存放了一个文件就不能再存放另外的文件。假如一个磁盘的分区大小为512MB，基于FAT16的系统的簇的大小为8KB，而FAT32系统的簇的大小仅是4KB，那么，现在我们存放一个3KB的文件，FAT16系统就会有5KB的空间被浪费，而FAT32的浪费则会少一些。如果分区达到1GB，FAT16的簇为16KB，而FAT32还是4KB，节省的也就更多了。

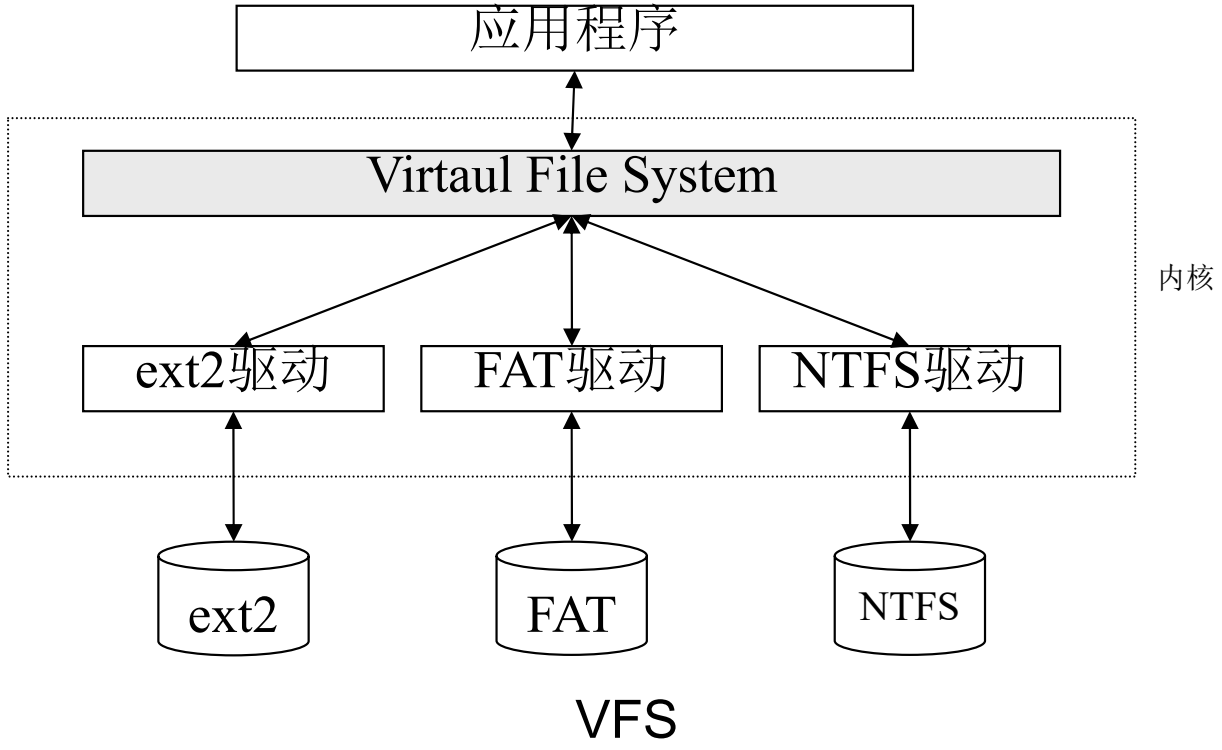
5.9 文件系统实例

- FAT32的缺点:首先是采用FAT32格式分区的磁盘，由于文件分配表的扩大，运行速度比采用FAT16格式分区的磁盘要慢。另外，由于DOS不支持这种分区格式，所以采用这种分区格式后，就无法再使用DOS系统。

5.9 文件系统实例

5.9.2 Linux文件系统概述

- 为了使Linux适应各种各样的文件系统，设计者采用了VFS的设计方法。Linux 文件系统的结构如图所示：



5.9 文件系统实例

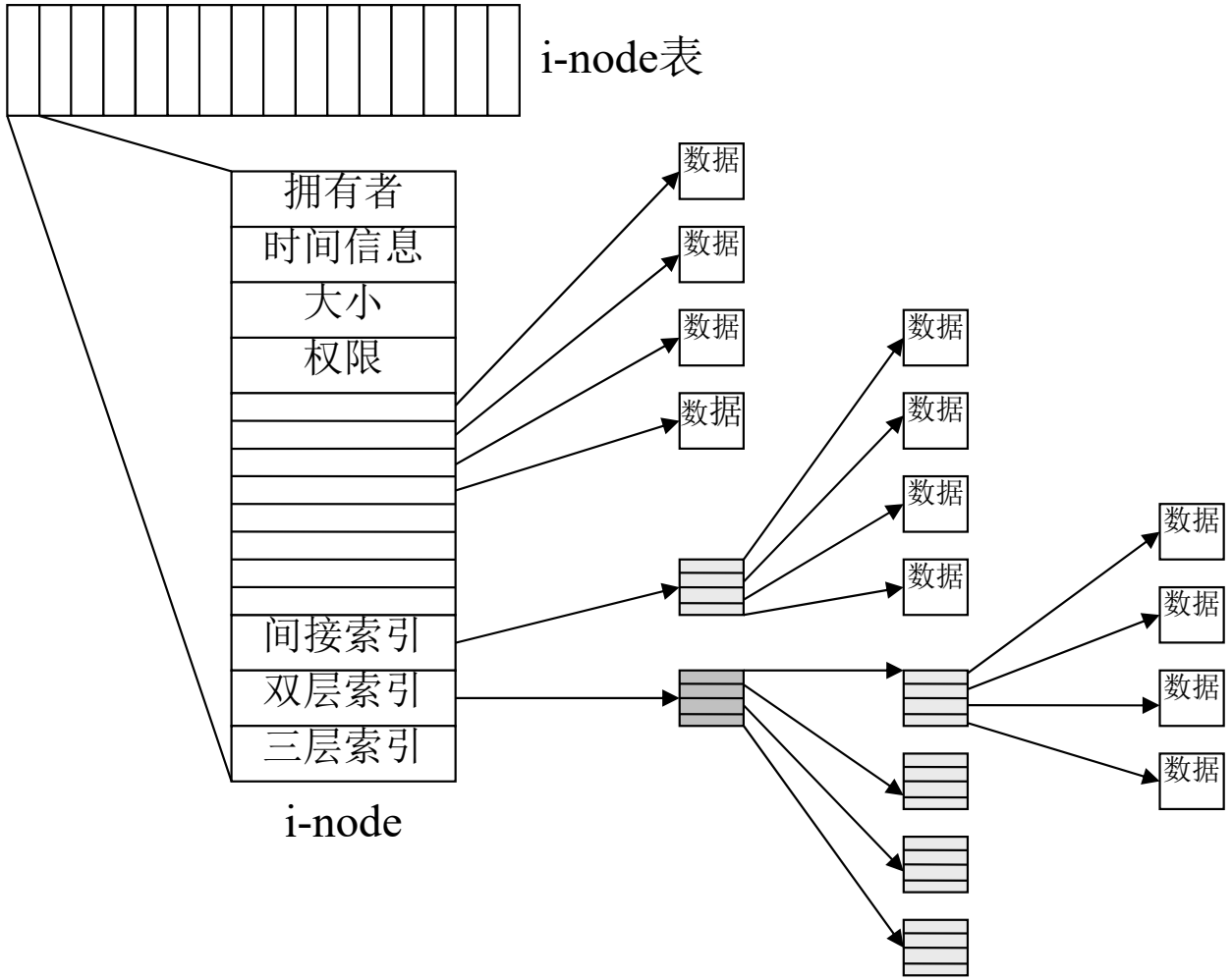
5.9.3 ext2文件系统

- Linux ext2/ext3文件系统使用索引节点来记录文件信息，作用像windows的文件分配表。索引节点是一个结构，它包含了一个文件的长度、创建及修改时间、权限、所属关系、磁盘中的位置等信息。一个文件系统维护了一个索引节点的数组，每个文件或目录都与索引节点数组中的唯一一个元素对应。系统给每个索引节点分配了一个号码，也就是该节点在数组中的索引号，称为索引节点号。

5.9 文件系统实例

- linux文件系统将文件索引节点号和文件名同时保存在目录中。所以，目录只是将文件的名称和它的索引节点号结合在一起的一张表，目录中每一对文件名称和索引节点号称为一个连接。对于一个文件来说有唯一的索引节点号与之对应，对于一个索引节点号，却可以有多个文件名与之对应。因此，在磁盘上的同一个文件可以通过不同的路径去访问它。

5.9 文件系统实例



Linux ext2索引节点结构

5.9 文件系统实例

- Linux缺省情况下使用的文件系统为ext2，ext2文件系统的确高效稳定。但是，随着Linux系统在关键业务中的应用，Linux文件系统的弱点也渐渐显露出来了:其中系统缺省使用的ext2文件系统是非日志文件系统。这在关键行业的应用是一个致命的弱点。
- ext3文件系统是直接从ext2文件系统发展而来，目前ext3文件系统已经非常稳定可靠。它完全兼容ext2文件系统。系统由ext2文件系统过渡到ext3日志文件系统升级过程平滑，可以最大限度地保证系统数据的安全性。

END