



現代操作系統

(Modern Operating System)

福州大学 计数学学院 江兰帆

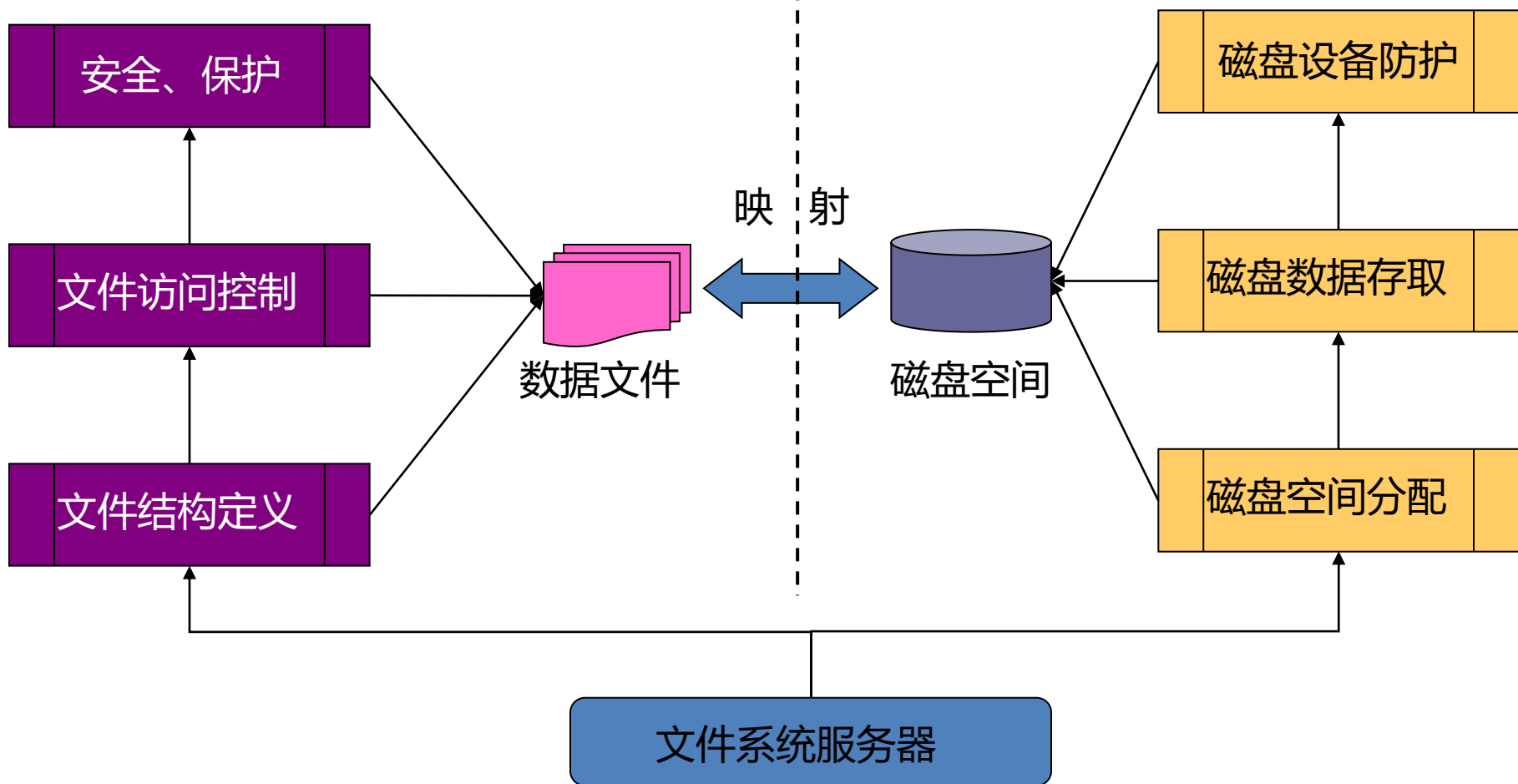
文件系统概述

- 计算机为什么需要文件？
 - 数量原因——内存无法保存大量信息
 - 时间原因——内存无法永久保存信息
 - 应用原因——内存无法方便实现共享

文件系统概述

应用层观点：逻辑抽象

物理层观点：空间管理



文件的组成和作用

- 具有标识的、在逻辑上有完整意义的信息项的序列。
- 标识——文件名、信息项——文件体。
- 文件的逻辑含义：由文件的创建者和使用者进行定义和维护。
- 文件必须便于存储、检索、共享。
- 广义的“文件”概念：所有可存储、提供信息资源的设备均可称为文件。

文件系统的组成和概念

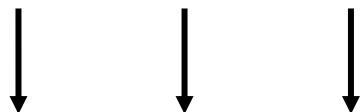
- 操作系统中统一管理信息资源的软件模块，负责管理文件的存储、检索、更新，提供安全可靠的共享和保护手段，方便用户的使用。
- 文件系统=文件管理程序+它所管理的全部文件

文件系统的功能目标

- 从系统角度来看，文件系统是对文件存储器的存储空间进行组织、分配和回收，负责文件的存储、检索、共享和保护。
- 从用户角度来看，文件系统主要是实现“按名存取”，文件系统的用户只要知道所需文件的文件名，就可存取文件中的信息，而无需知道这些文件究竟存放在什么地方。

文件系统模型

用户(程序)



文件系统接口

操纵和管理对
象的软件集合

对象及属性

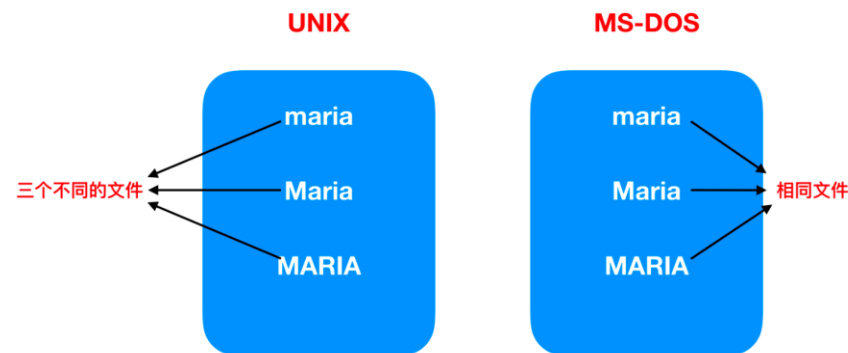
文件
目录
磁盘空间



4.1 文件



4.1.1 文件命名



- 文件命名提供了一种将数据保存在外部存储介质上以便于访问的功能。用户不必关心文件存储方法，物理位置以及访问方式等，而可以通过文件名来使用文件。
- 名称.扩展名（为什么用这种格式呢？）

4.1.1 文件命名

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

4.1.2 文件结构

- 文件的结构是指文件的组织形式，文件的结构有两种，一种是逻辑结构，另一种是物理结构。
- 逻辑结构与物理结构的差别
 - 逻辑结构：内容的组织形式
 - 物理结构：数据的存储

4.1.2 文件结构

- 逻辑结构（用户观点）：研究的是用户思维中的抽象文件，也叫逻辑文件。其目的是为用户提供一种结构清晰、使用简便的逻辑组织。用户按此去存储、检索和加工处理有关文件信息。

4.1.2 文件结构

- 物理结构（实现观点）：研究的是存储在物理设备介质上的实际文件，即物理文件。其目的是选择一些性能良好、设备利用率高的物理结构。系统按此和外部设备打交道，控制信息的传输。

文件逻辑结构

- 按文件的逻辑结构分，可将文件分为无结构的字符流式文件和有结构的记录式文件。

文件逻辑结构

1. 字符流式文件

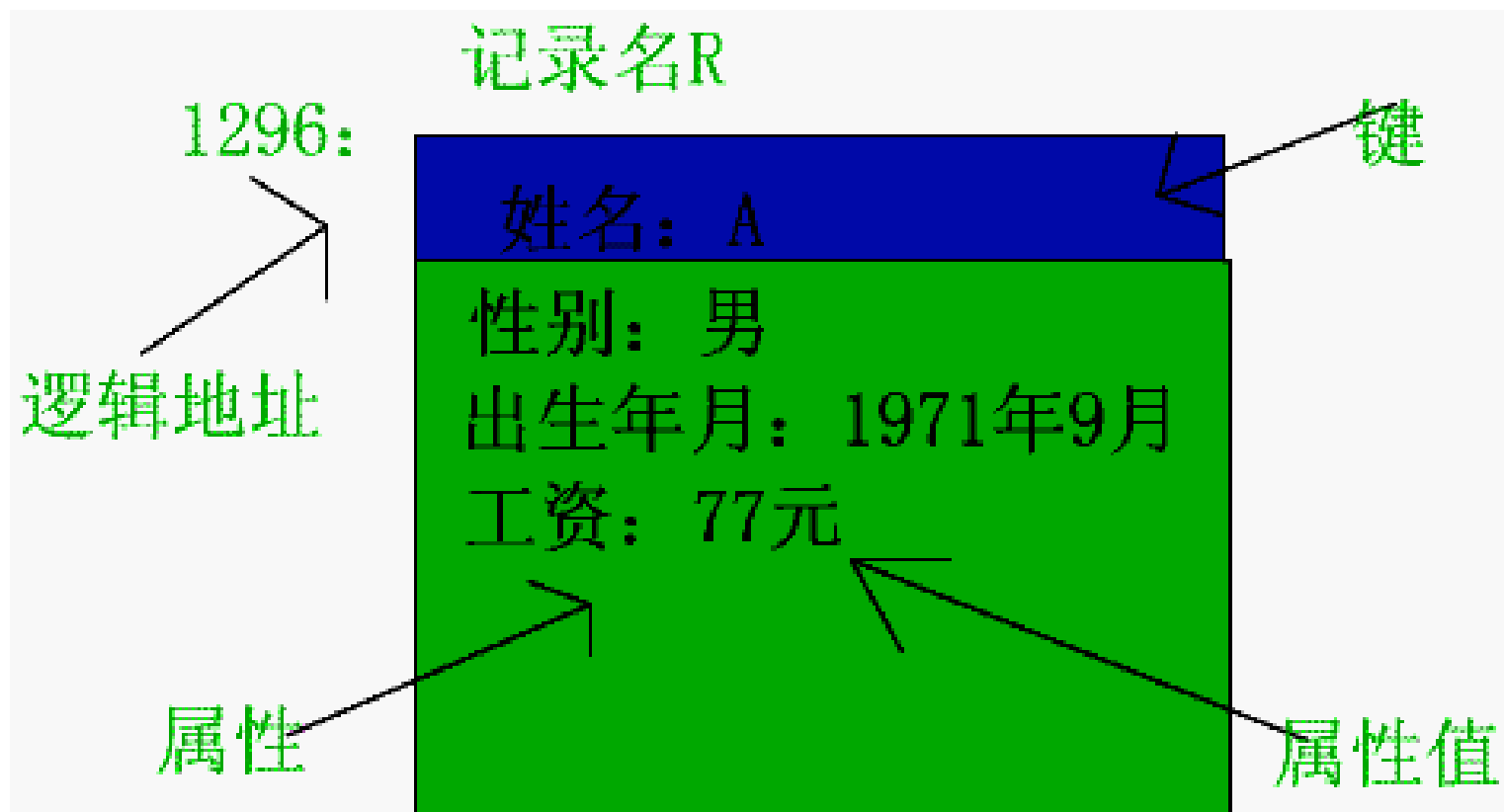
- 字符流式文件是由字符序列组成的文件，其内部信息不再划分结构，也可以理解为字符是该文件的基本信息单位。访问流式文件时，依靠读写指针来指出下一个要访问的字符。
- 这种文件的管理简单，要查找信息的基本单位困难。

文件逻辑结构

2. 记录式文件

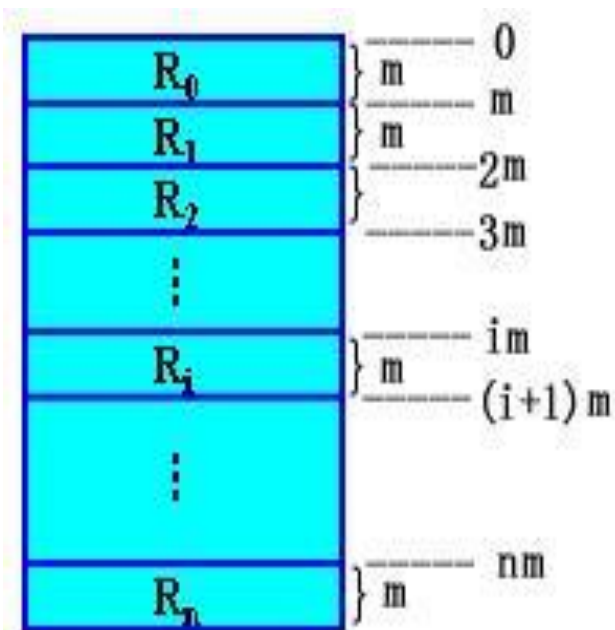
- 这是一种有结构文件。它把文件内的信息划分为多个记录，用户以记录为单位来组织信息。
- 记录是一个具有特定意义的信息单位，它由该记录在文件中的相对位置、记录名以及该记录对应的一组键、属性及属性值组成，可按键值进行查找。

记录式文件

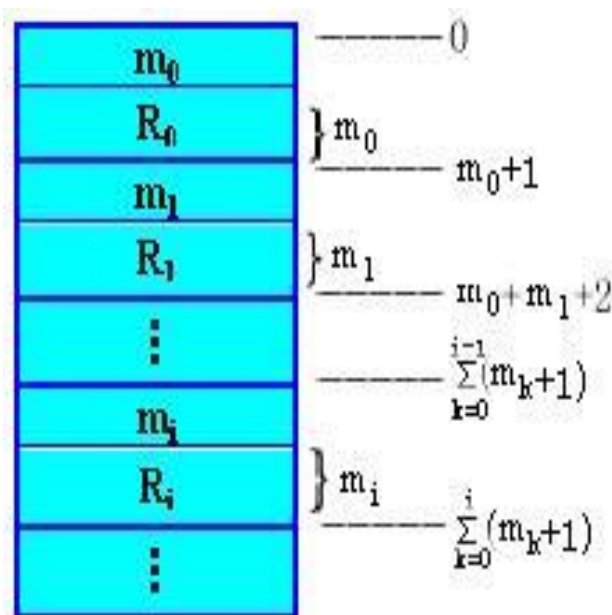


记录式文件

- 定长记录：所有记录长度相等
- 变长记录：记录长度不固定。



(a)固定长度记录

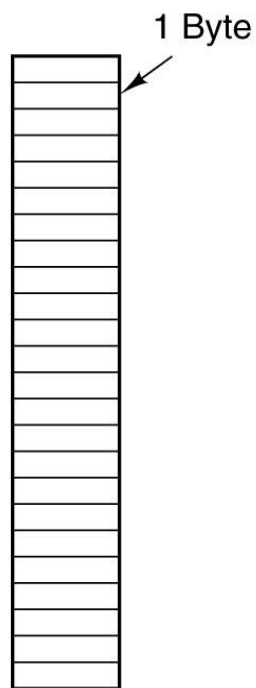


(b)可变长度记录

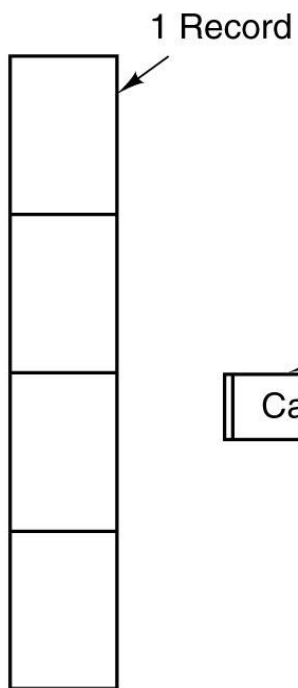
记录式文件

- 在定长记录文件中，各个记录长度相等。
在检索时，可以根据记录号 I 及记录长度 L 就可以确定该记录的逻辑地址。
- 在不定长记录文件中，各个记录的长度不等，在查找时，必须从第一个记录起一个记录一个记录查找，直到找到所需的记录。

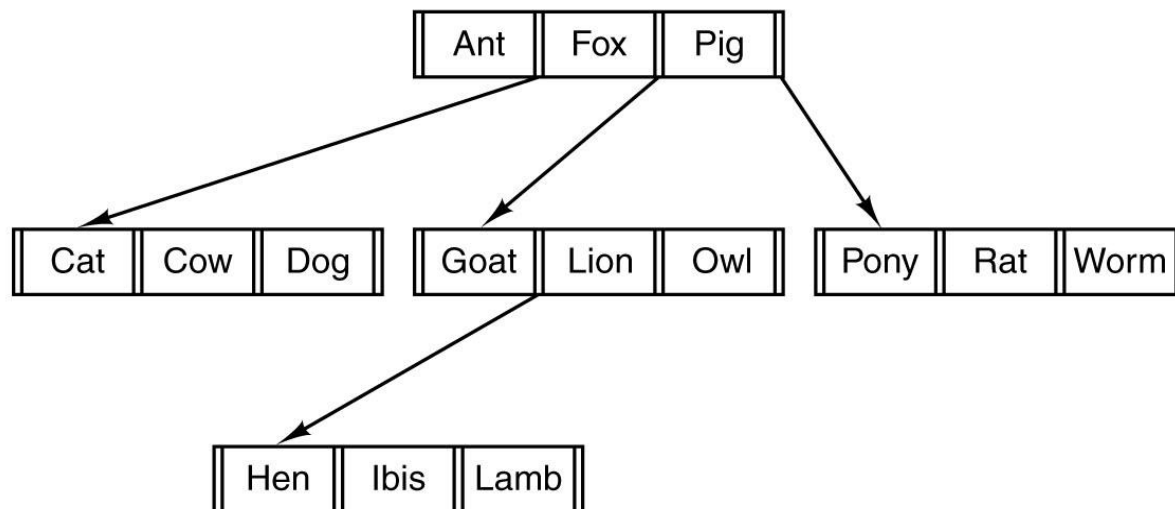
文件逻辑结构



(a)



(b)



(c)

记录序列

树

4.1.3 文件类型

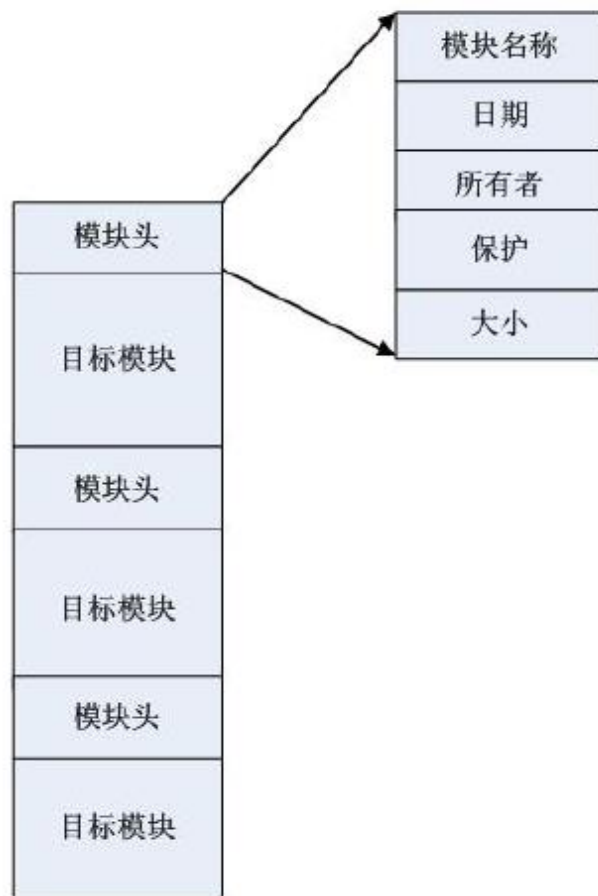
普通文件

- **ASCII文件**：可以打印和显示，还可以用任何文本编辑器进行编辑，且易于输入和输出。
- **二进制文件**：有一定的内部结构。

4.1.3 文件类型



可执行文件



存档文件

4.1.4 文件存取

- 顺序存取。顺序存取是最简单的方法。它严格按照文件信息单位排列的顺序依次存取，后一次存取总是在前一次存取的基础上进行，所以不必给出具体的存取位置。
- 随机存取。随机存取又称直接存取，在存取时必须先确定进行存取时的起始位置（如记录号、字符序号等）。

4.1.4 文件存取

- 文件索引访问，也称按键访问，这种方式对文件中的记录按某个数据项(通常称为键)的值来排列，从而可以根据键值来快速存取。如索引表很长，则可以将索引表再加以索引，以形成具有层次结构的多级索引。如果将记录块的物理位置作为键值，那么可以将随机访问作为索引访问的特例。

4.1.5 文件属性

文件可包括两个部分内容：

- 文件所包含的数据，常称为文件数据；
- 关于文件本身的说明信息或属性信息，常称为文件属性。这些信息主要被文件系统用来管理文件。

域	含义
保护	谁能访问该文件，以何种方式访问
口令	访问该文件所需口令
创建者	文件创建者的ID
拥有者	当前拥有者
只读标志	0表示读写，1表示只读
隐藏标志	0表示正常，1表示不在列表中显示
系统标志	0表示正常文件，1表示系统文件
存档标志	0表示已备份过，1表示需要备份
ASCII/二进制	0表示ASCII文件，1表示二进制文件
随机存取标志	0表示只能顺序存取，1表示随机存取
临时标志	0表示正常，1表示在进程退出时删除文件
锁标志	0表示未锁，非零表示已锁
记录长度	一条记录的字节数
关键字位置	每条记录中关键字偏移
关键字长度	关键字域的字节数
创建时间	文件创建的日期和时间
最后存取时间	文件最后存取的日期和时间
最后修改时间	文件最后修改的日期和时间
当前长度	文件字节数
最大长度	文件最大允许字节数

4.1.6 文件操作

- **CREATE**: 创建没有任何数据的文件。
- **DELETE**: 删除文件以释放磁盘空间。
- **OPEN**: 将文件属性和磁盘地址表载入主存, 便于以后系统调用的快速存取。
- **CLOSE**: 关闭文件以释放内部表空间。
- **READ**: 从文件中读取数据。一般, 读出的数据来自当前位置。调用者必须指明需要读取多少数据, 并且提供存放这些数据的缓冲区。
- **WRITE**: 向文件中写入数据, 写操作一般也是从当前位置开始。如果当前位置是文件末尾, 文件长度增加。如果当前位置在文件中间, 则现有数据被重写, 并且永远丢失了。

4.1.6 文件操作

- **APPEND**: 该调用是**WRITE**的限制形式, 它只能在文件末尾添加数据。
- **SEEK**: 把当前位置指针指向文件中特定位置。
- **GET ATTRIBUTES**: 读取文件属性。
- **SET ATTRIBUTES**: 设置文件属性。
- **RENAME**: 改变现有文件的文件名。

4.2 目錄

1. 文件系統怎樣實現文件的“按名存取”？
2. 如何查找文件存儲器中的指定文件？
3. 如何有效地管理用戶文件和系統文件？



4.2 目录

- 文件控制块FCB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有关信息。
- 文件控制块是文件存在的标志

文件控制块的内容

- 基本信息类

- 文件名

- 物理位置

- 逻辑结构

- 物理结构

- 存取控制信息类

- 文件主的权限

- 核准用户的权限

- 一般用户的权限

- 使用信息类

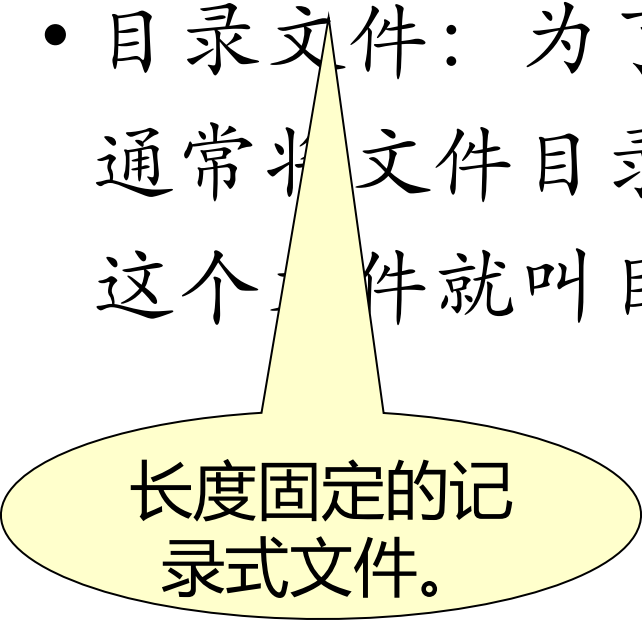
- 文件建立时间

- 上次修改时间

- 当前使用信息

4.2 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。



长度固定的记录式文件。

目录结构

- 目录结构的组织关系到文件系统的存取速度，关系到文件共享性和安全性，因此组织好文件的目录是设计文件系统的重要环节。

4.2.1 一级目录系统

- 在系统中设置一张线性目录表，表中包括了所有文件的文件控制块，每个文件控制块指向一个普通文件，这就是一级目录结构。



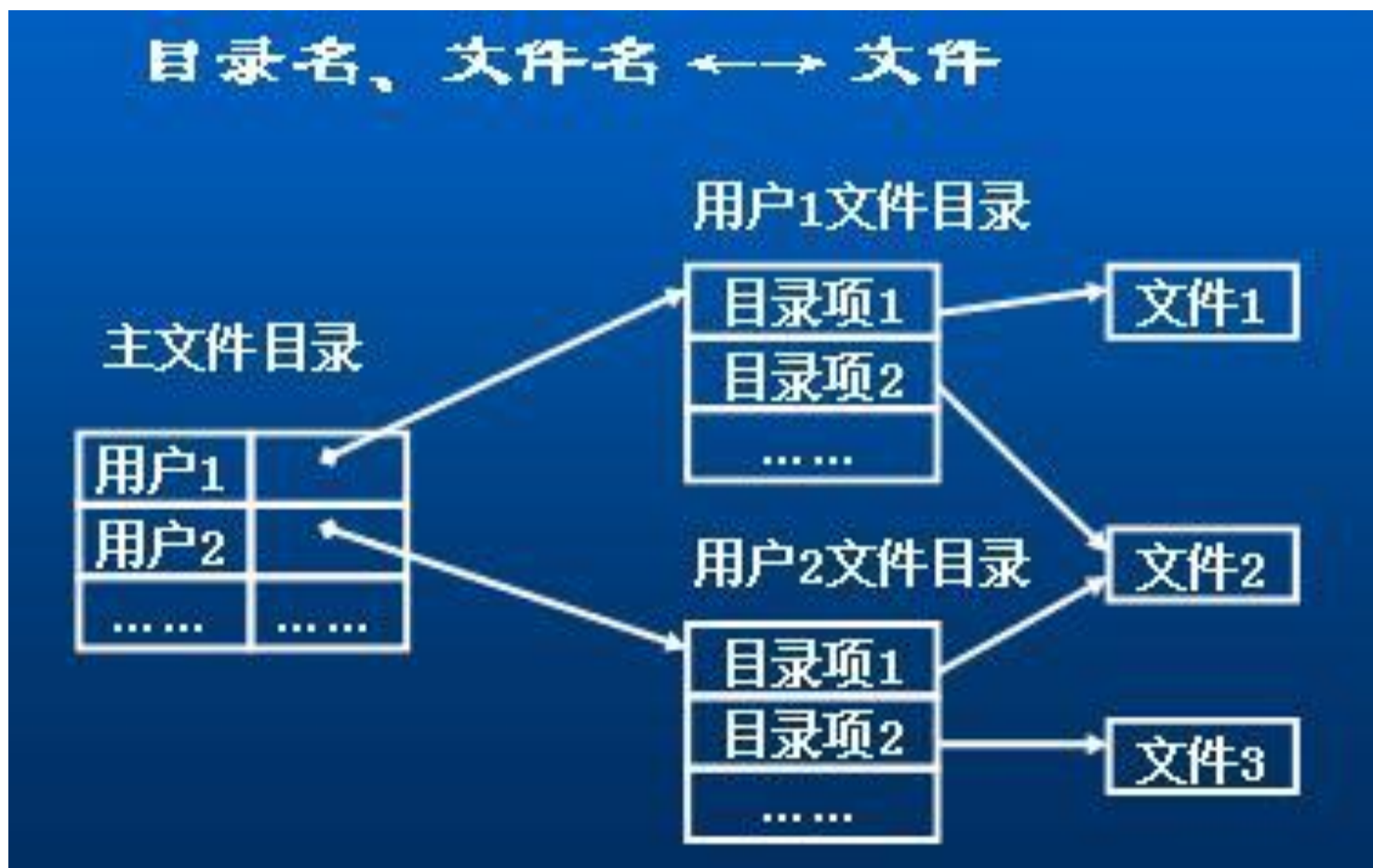
4.2.1 一级目录系统

- 一级目录结构简单，能实现目录管理的基本功能--按名存取，但存在以下缺点：
 - 查找速度慢
 - 不允许重名
 - 不便于实现文件共享
- 因此它只适用于单用户环境。

4.2.2 两级目录系统

- 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(MFD)，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(UFD)，给出该用户所有文件的FCB

4.2.2 两级目录系统



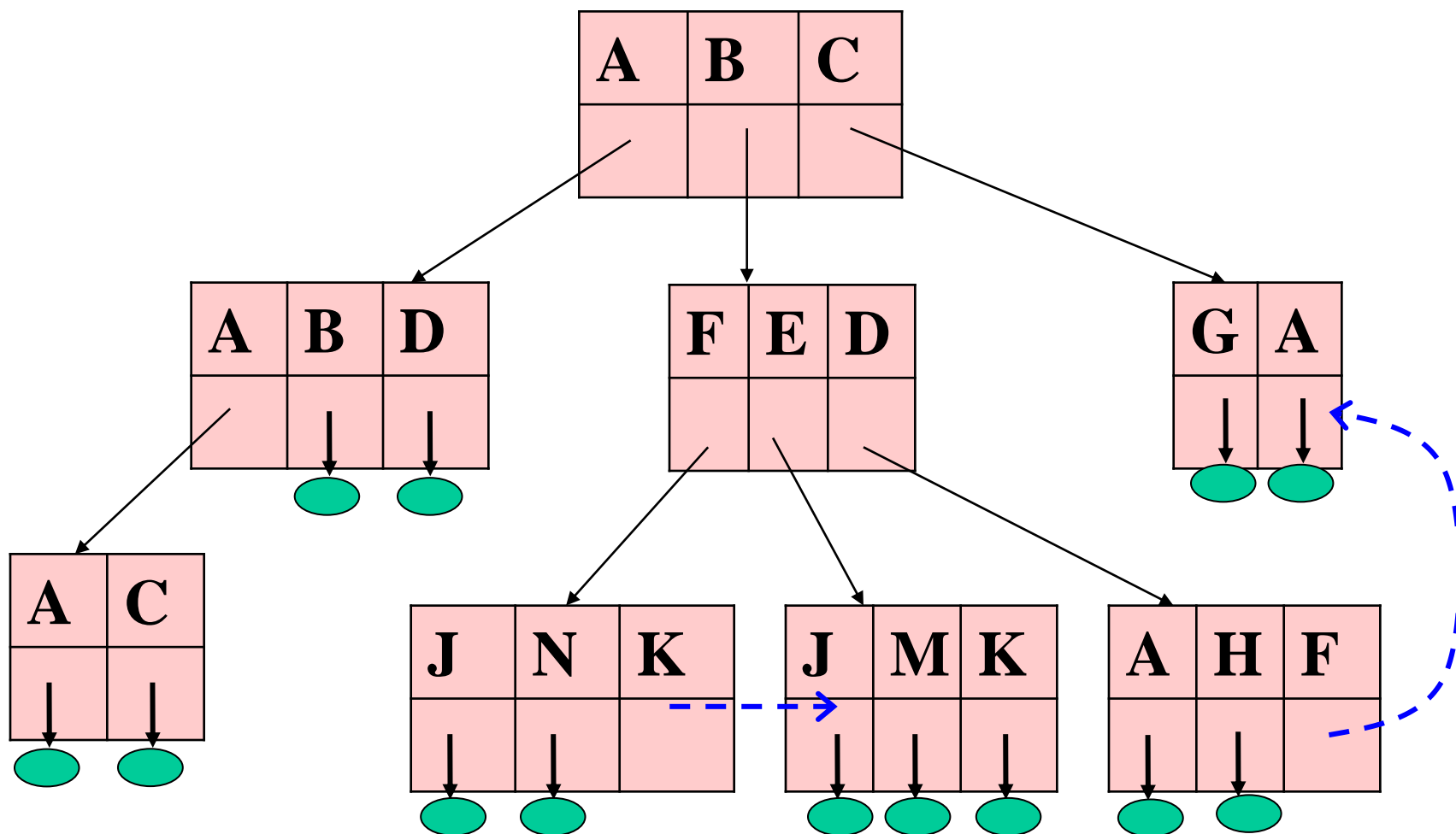
4.2.2 两级目录系统

- 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。

4.2.3 树型目录系统

- 把二级目录的层次关系加以推广，就形成了多级目录，又称为树型目录结构。
- 在树型目录结构中，除了最低一级的物理块中装有文件信息外，其他每一级目录中存放的都是下一级目录或文件的说明信息，由此形成层次关系，最高层为根目录，最低层为文件。

多级目录结构



4.2.3 树型目录系统

- 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，多次访盘影响速度

4.2.4 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，依次连接起来，即构成该数据文件的路径名。

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

目录检索

- 全路径名：从根目录开始，列出由根到用户指定文件的全部有关子目录，又称为“绝对路径名”。当目录层次较多时检索要耗费很多时间。
- 相对路径：从当前目录开始到所要访问文件的一段路径，即以当前目录作为路径的相对参照点，检索路径缩短，检索速度提高。

4.2.5 目录操作

- 对目录的操作，通常由系统调用实现，在不同系统中，管理目录的系统调用是不同的，下面给出UNIX的一些例子，以便具体了解这些系统调用及它们的工作方式。
- 请注意，在UNIX中，“.”代表当前目录，“..”代表上一级目录。

4.2.5 目录操作

- CREATE, 创建目录。
- DELETE, 删除目录。
- OPENDIR, 打开目录。
- CLOSEDIR, 关闭目录。
- READDIR, 系统调用READDIR返回打开目录的下一目录项。
- RENAME, 文件可换名, 目录也可换名。
- LINK, 链接技术允许在多个目录中出现同一文件。
- UNLINK, 删除目录项。

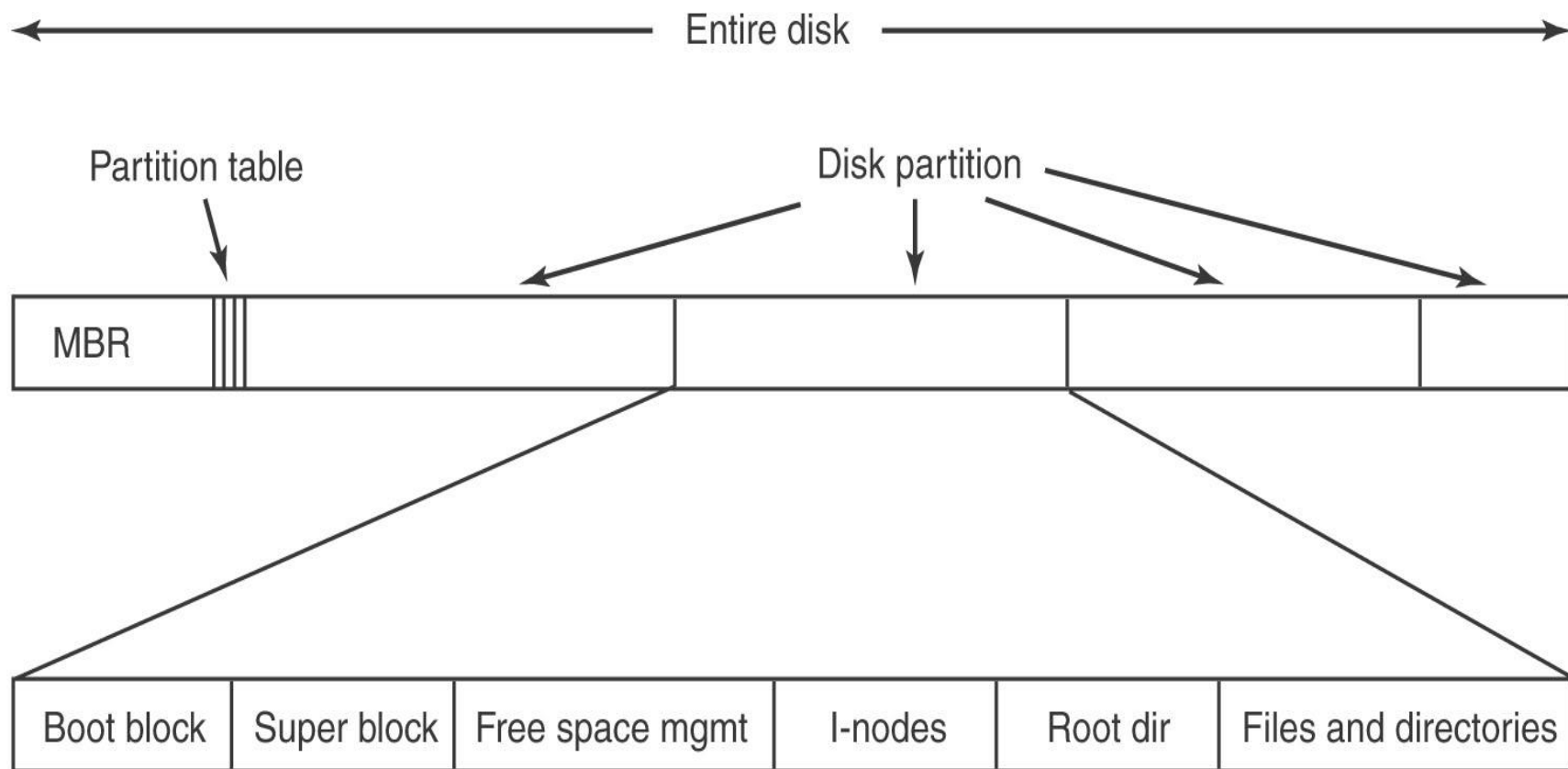
4.3 文件系统的实现

- 文件系统布局
 - 如何划分磁盘空间——磁盘分区
- 文件系统的技术实现
 - 文件的实现：磁盘空间分配方案：连续、链接、索引
 - 目录的实现：如何有效的保存文件名、文件属性和物理地址
 - 文件共享的实现

4.3 文件系统的实现

- 磁盘空间管理
 - 以块为单位使用磁盘空间——对比存储管理中的“分页式”
 - 空闲块记录与磁盘配额限制
- 文件系统的可靠性与性能保证
 - 文件备份与文件系统一致性
 - 高速缓存、块提前读、减少磁盘臂运动

4.3.1 文件系统布局



4.3.1 文件系统布局

- 在文件系统中，文件的存储设备通常划分为若干个大小相等的物理块，每块长一般为 512 字节或 1024 字节。与此相对应，为了有效地利用存储设备和便于系统管理，一般把文件信息也划分为与物理存储设备的物理块大小相等的逻辑块。从而，以块作为分配和传送信息的基本单位。

4.3.2 文件的实现

- 若1000个学生的信息放在一个文件中，每个学生的信息都占用10B.
- 要读取第60个学生的信息，偏移？



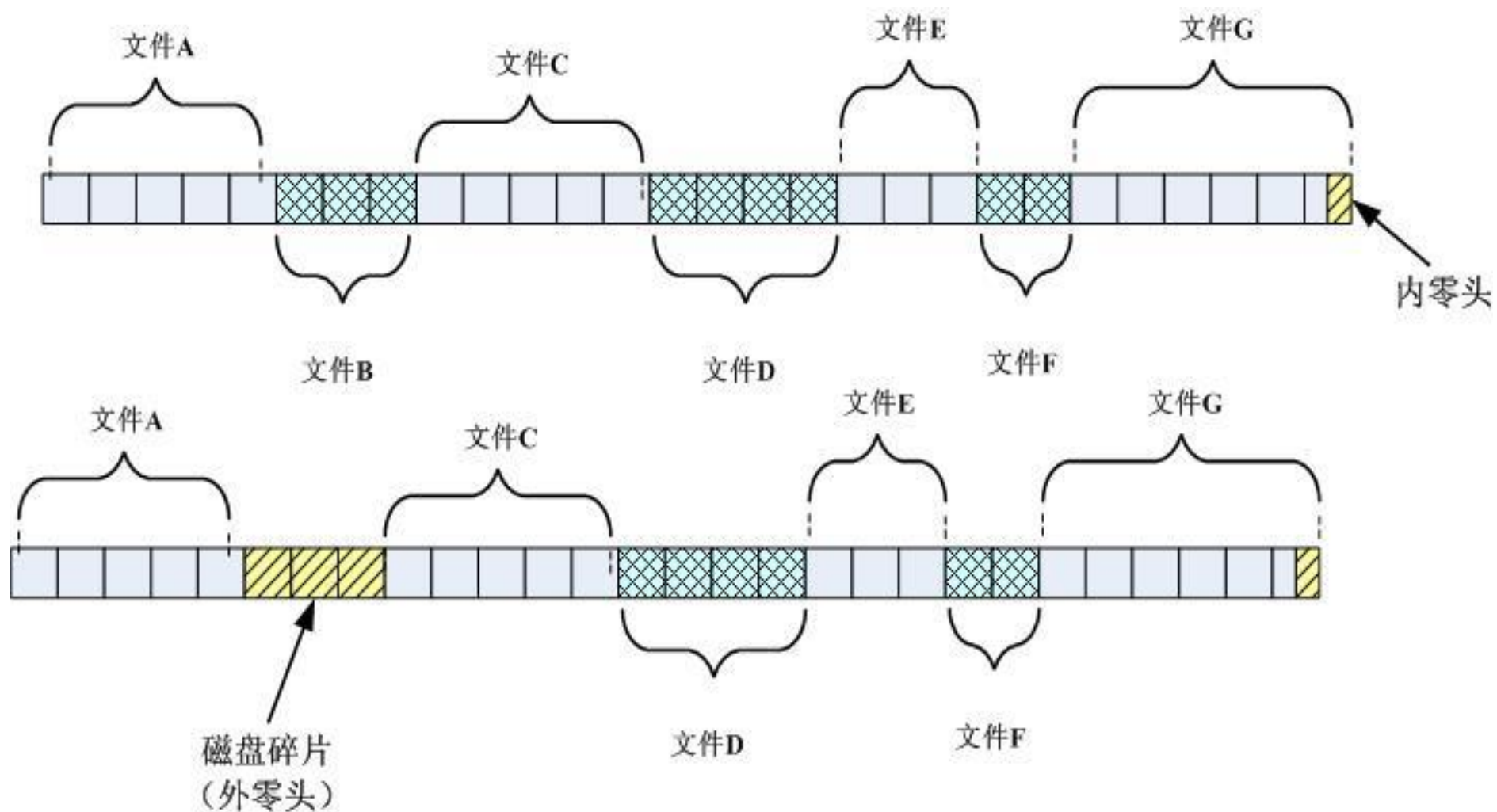
4.3.2 文件的实现

- 常用的磁盘空间分配策略主要包括连续空间分配、链接空间分配、索引空间分配等。每种策略都有各自的优缺点。一般情况下，一个文件系统只采用一种策略来分配文件空间。

连续分配

- 将一个文件中逻辑上连续的信息存放到存储介质的依次相邻的块上便形成顺序结构，这类文件叫连续文件，又称顺序文件。

连续分配



连续分配

Question:

- 目录(FCB)中用于寻址的信息有哪些?
- 对于这类文件, 目录通常只需包括文件名、文件块的起始地址和文件长度。

连续分配

连续分配的主要优点如下：

- 实现简单；
- 支持顺序存取和随机存取；
- 顺序访问速度快；
- 所需的磁盘寻道次数和寻道时间最少。

连续分配

连续分配的主要缺点如下：

- 建立文件前需要能预先确定文件长度，以便分配存储空间；
- 修改、插入和增生文件记录有困难；
- 要求有连续的存储空间：容易产生碎片问题。

不连续分配背景下的讨论

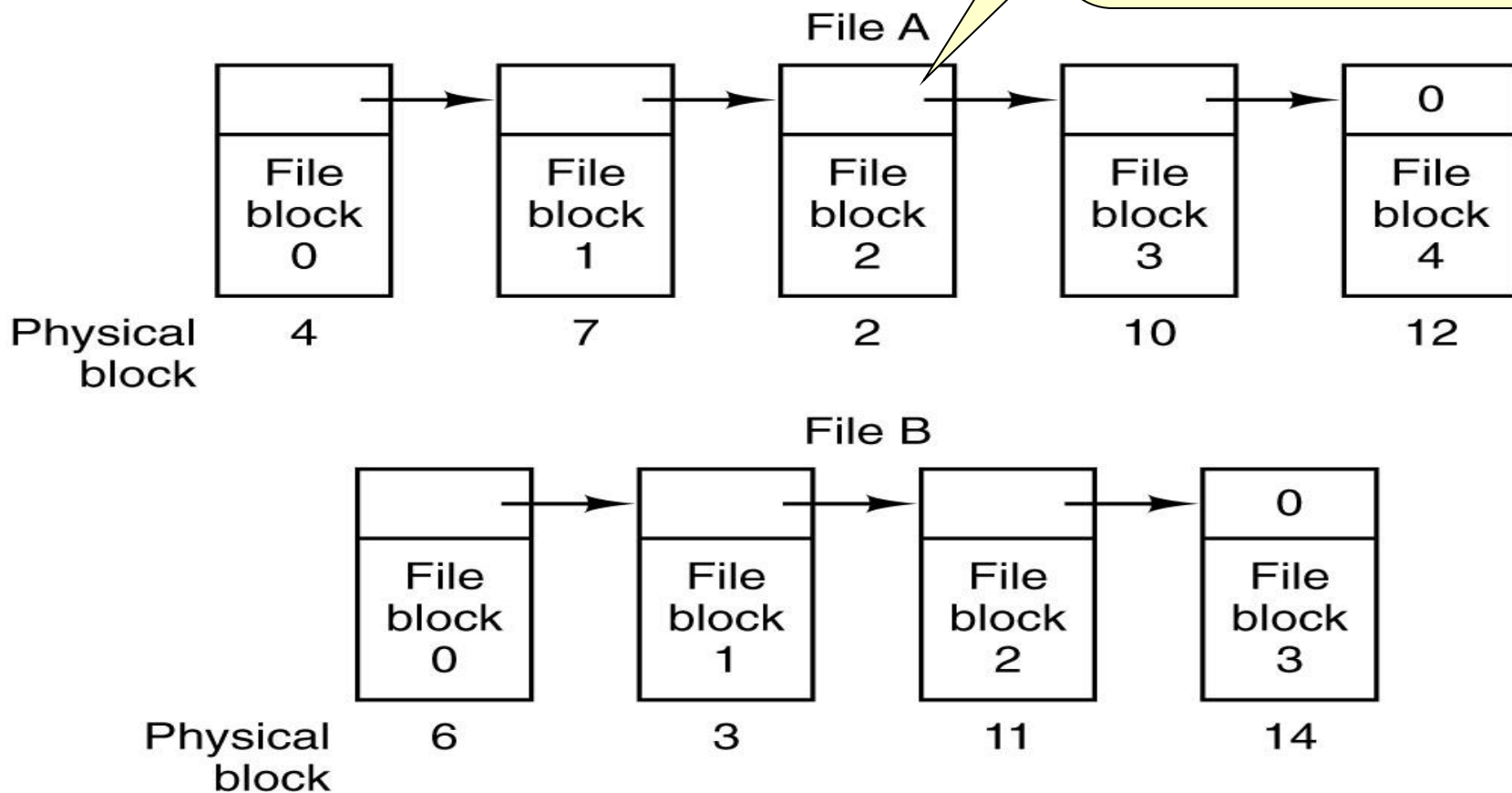
- 文件内容在外存上存储介质上不一定占有连续空间。
- 如何为每个文件分配空间？ 分配哪些扇区？

链接分配

- 链接分配采取离散分配方式，将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。

链接分配

每个块的第一个字用于指向下一块的指针，块的其他部分存放数据



链接分配

Question:

- 目录中用于寻址的信息有哪些?
- 对于采用链接空间分配的文件，目录项通常只需包括文件名、文件的开始块和结束块。

链接分配

优点：

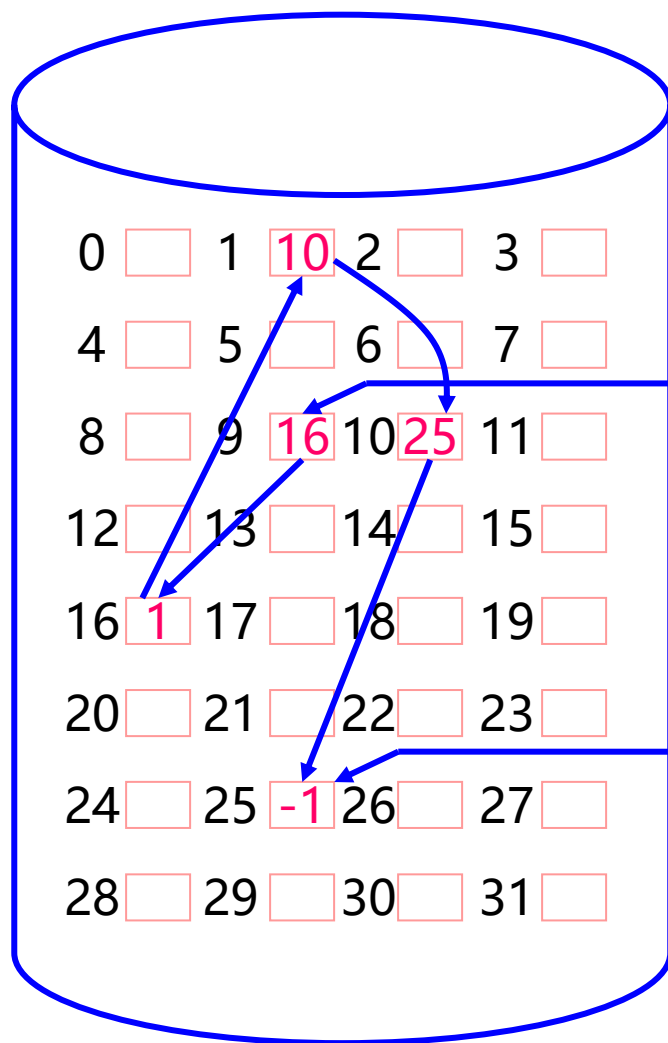
- 提高了磁盘空间利用率
- 不存在外部碎片问题
- 有利于文件插入和删除
- 有利于文件动态扩充

链接分配

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

隐式链接



文件目录

文件名	始址	末址
jeep	9	25

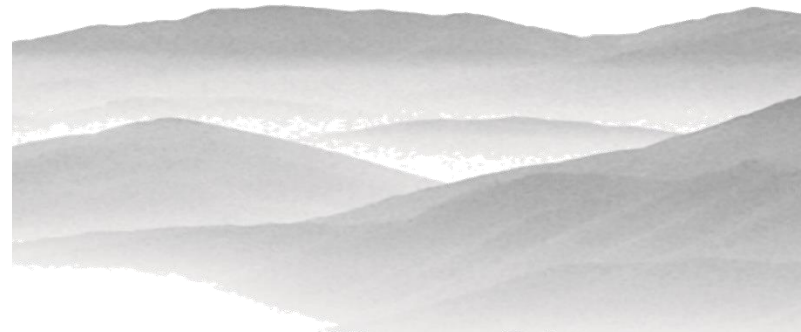
显式链接

- 设置一张文件分配表**FAT**，显式存放用于链接文件各个物理块的指针。
- 表的序号是物理盘块号，从**0**开始，直至**N-1**。其中**N**为磁盘盘块总数。
- 每个表项中存放链接指针，即下一个盘块号。
- 该表中，每个文件的第一个盘块号，作为文件地址填入相应文件**FCB**的“物理地址”字段中。



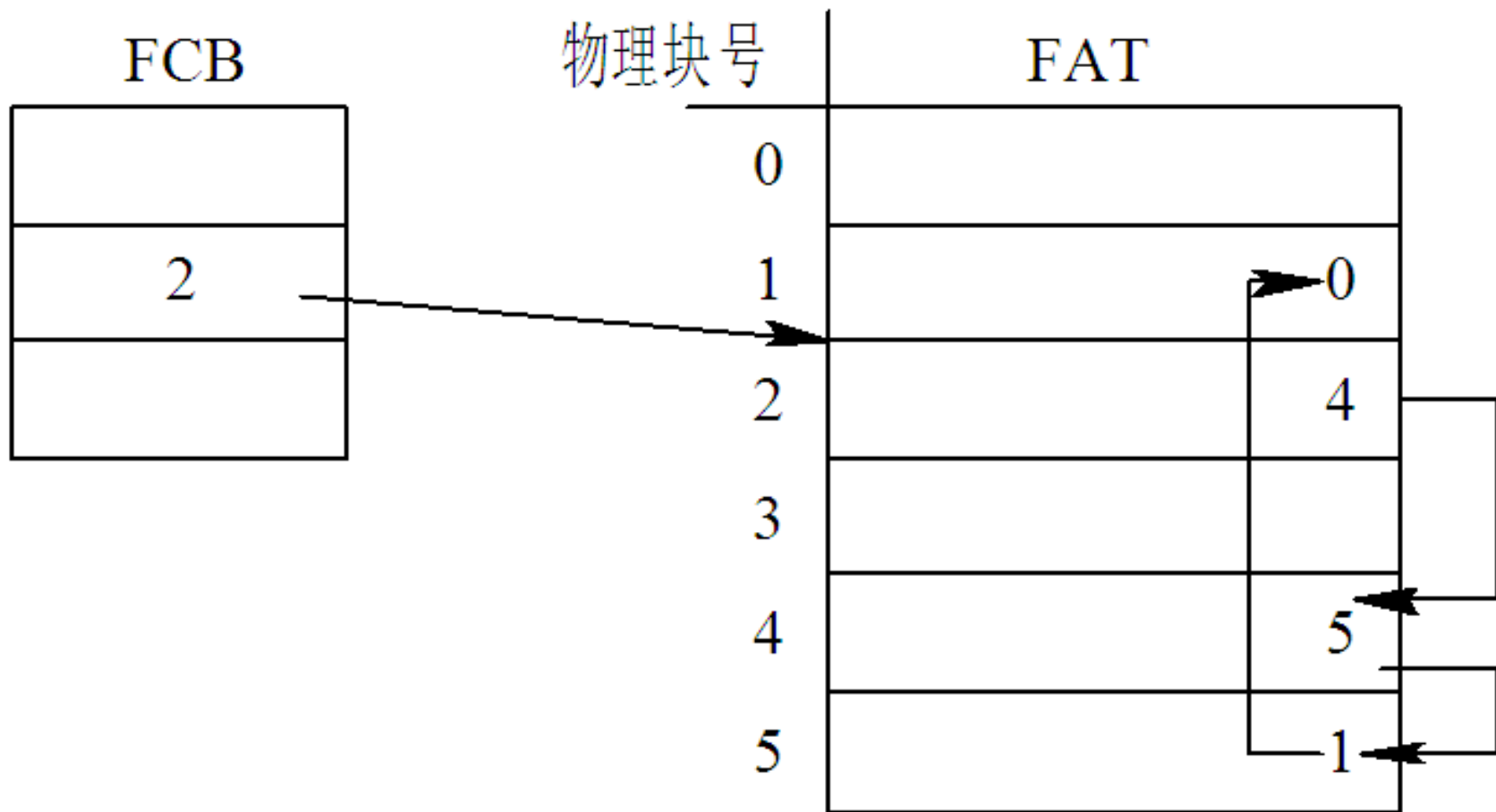
顯式鏈接

0		
1		
2	10	
3	11	
4	7	← 文件 A 从这开始
5		
6	3	← 文件 B 从这开始
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← 未用块



显式链接

整张FAT表都要加载到内存中



链接方式

链接分配方式虽然解决了连续分配方式所存在的问题，但又出现了另外两个问题，即：

- 不能支持高效的直接存取。要对一个较大的文件进行直接存取，须首先在FAT中顺序地查找许多盘块号。
- FAT需占用较大的内存空间。

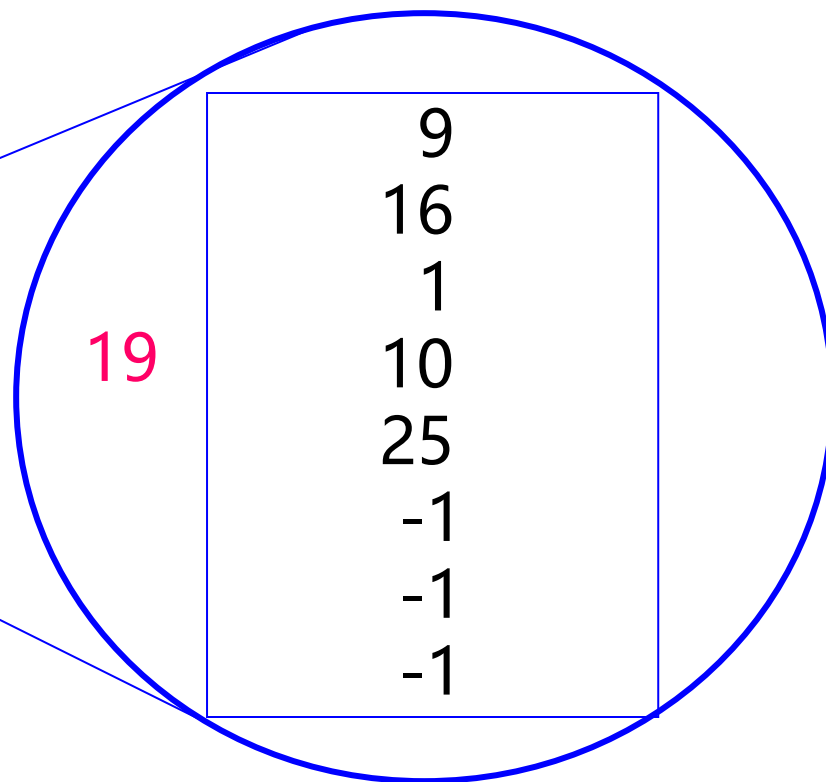
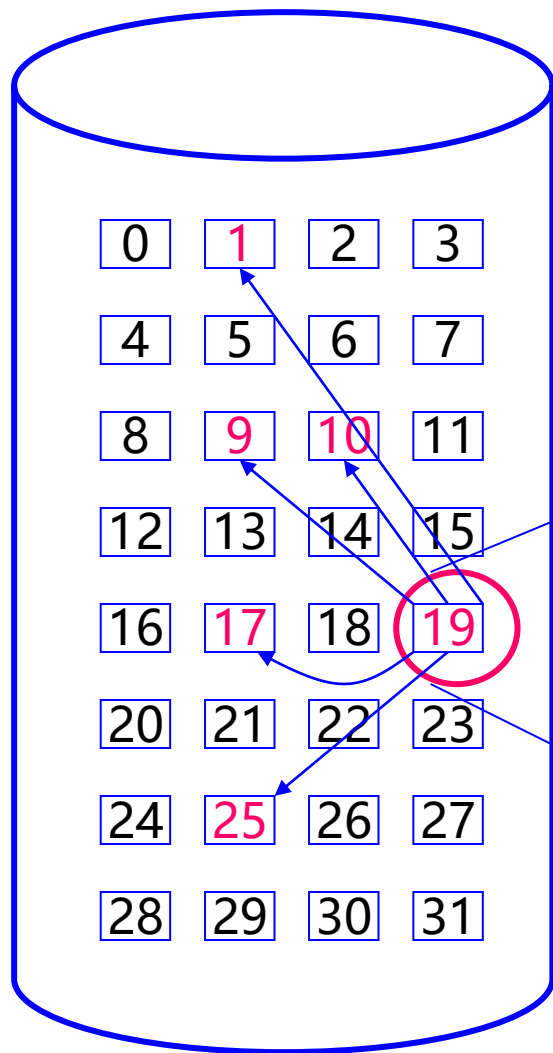
索引分配

- 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；
- 索引表：一个文件所有记录的关键字和其地址的对照表；
- 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

单级索引分配

文件目录

文件名	索引表地址
Jeep	19



索引表放在一个索引盘块中，该盘块地址（即索引表地址）放在FCB中。

索引结构优缺点

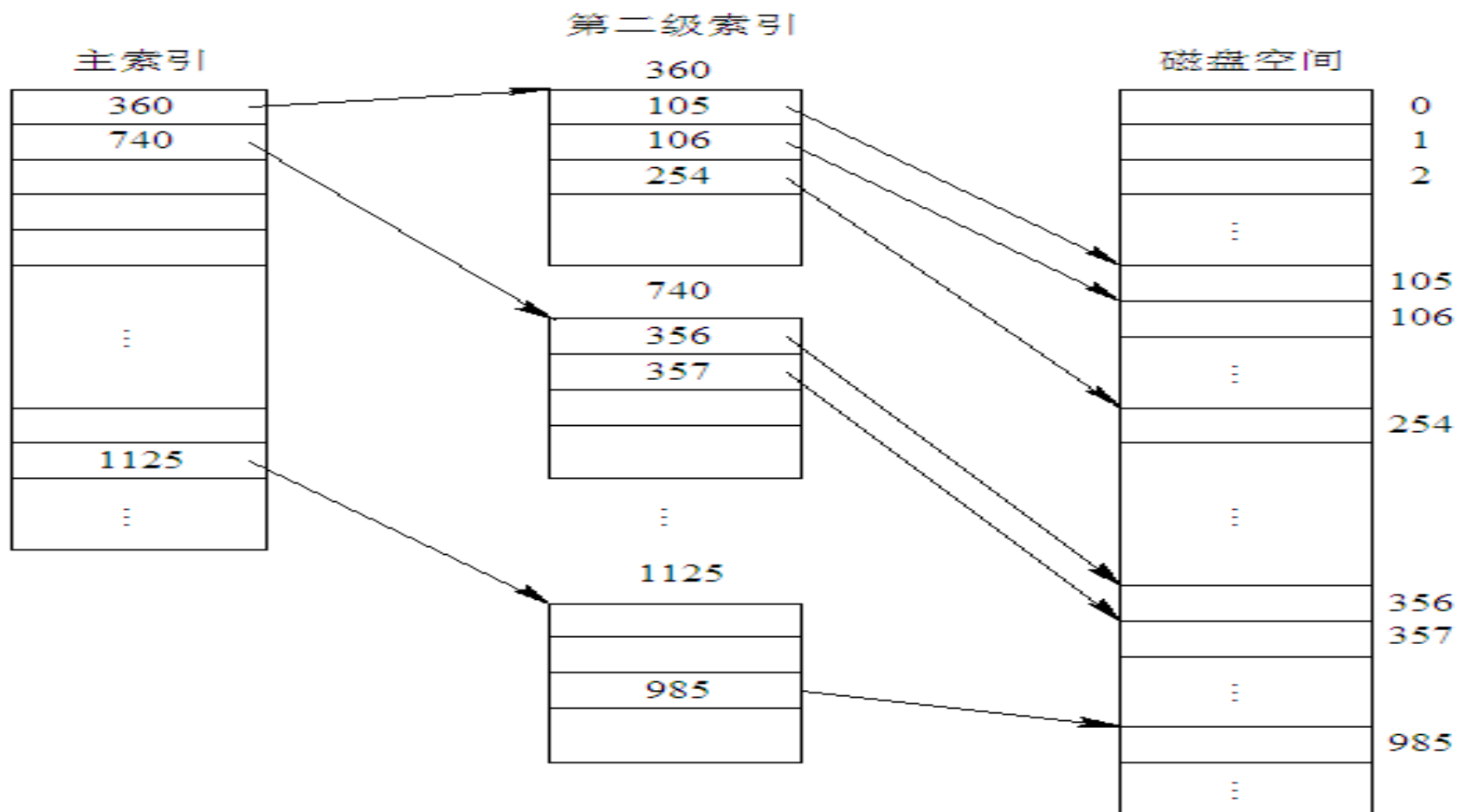
优点：

- 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，也能充分利用外存空间。

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，存取时间。

多级索引分配



问题的提出

- 单级索引方式中，文件长度受限；
- 多级索引，小文件将引起索引盘块浪费；

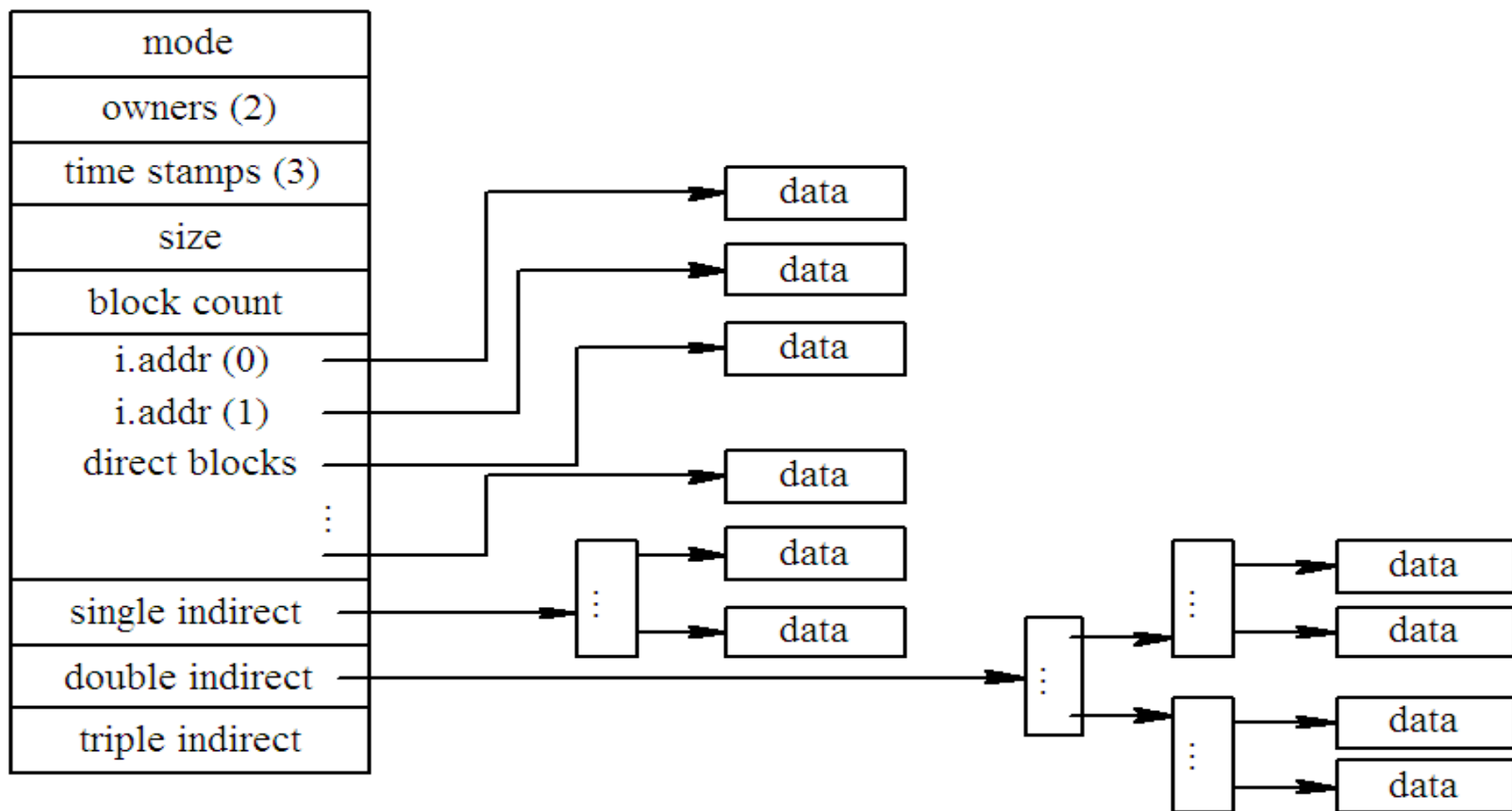
Question:

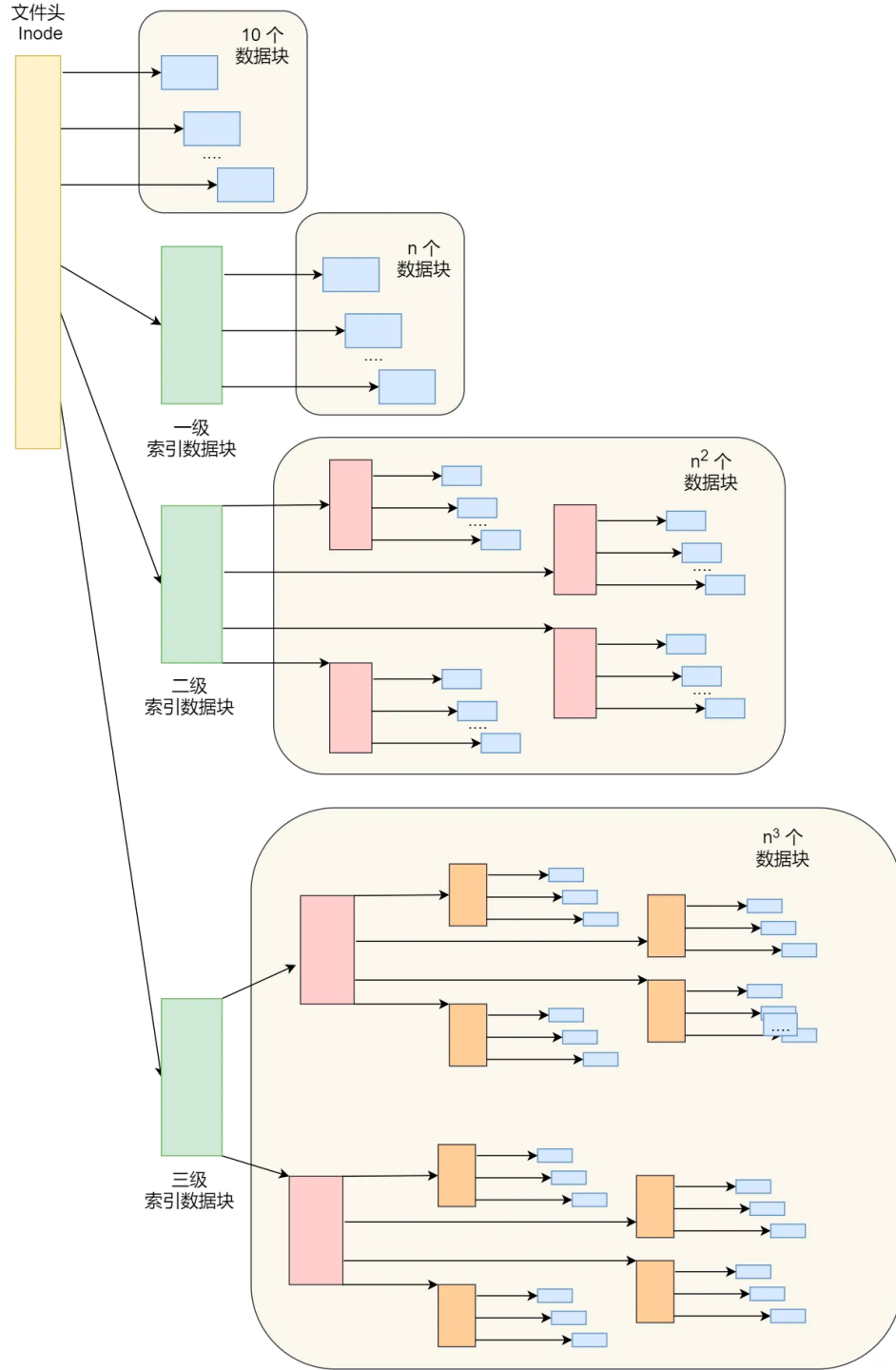
- 如何灵活解决大小不同的文件存放问题？

i-节点

- i-节点是一种多级索引文件结构。i-节点最早出现在 **UNIX** 操作系统中，是多级索引结构文件在 **UNIX** 中的具体实现。i-节点在一般多级索引结构文件的基础上，进行了结构上的变化，克服了索引结构的缺点。

混合索引方式





混合索引方式

- UNIX文件系统采用的是多级索引结构(综合模式)。每个文件的索引表为13个索引项，每项2个字节。
- 最前面10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10块，则利用第11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（一次间接寻址）；
- 对于更大的文件还可利用第12和第13项作为二次和三次间接寻址

思考

如上图，若每个物理块大小为4KB，每个索引项4个字节，则：

- 直接地址范围？ **0 — 40KB**
- 一次间址地址范围？ **40KB + 4MB**
- 二次间址地址范围？ **40KB + 4MB + 4GB**

文件物理空间分配方式的总结

		连续分配	链表方式	索引方式
存储介质	磁带	支持	不支持	不支持
	磁盘	支持	支持	支持
存取方式		顺序+随机存取	顺序	顺序+随机存取
空间利用效率		较低，会产生外零头	指针占用磁盘空间引起管理问题	利用磁盘和内存，但效率很高
应用环境分析		最简单、最原始	中间过渡阶段	广泛应用

【思考题I】

- 请分别解释在连续分配方式、隐式链接分配方式、显式链接分配方式和索引分配方式中如何将文件的字节偏移量**3500**转换为磁盘物理地址。设盘块大小为**1KB**，盘块号需占**4**个字节。

【思考题I】

- 在连续分配方式中，可从相应文件的FCB中得到分配给该文件的起始物理盘块号，例如A0。 $3500/1024=3\ldots\ldots 428$ ，即逻辑块号为3，块内偏移为428。因此，物理块号为A0+3，块内位移量为428。

【思考题I】

- 在隐式链接方式中，由于每个盘块中需留出4个字节（假设为最后的4个字节）来存放分配给文件的下一个盘块的块号，因此，字节偏移量为3500的逻辑块号为 $3500/1020$ 的商3，而块内位移量为440。
- 从相应的FCB中可获得分配给该文件的首个盘块的块号，如B0；然后通过读第B0块获得分配给文件的第1个盘块的块号，如B1；再从B1块中得到第2块的块号，如B2；从B2块中得到第3块的块号，如B3。

【思考题1】

- 在显式链接方式中，可从文件的FCB中得到分配给该文件的首个盘块的块号，如C0；再在FAT表的第C0项中得到分配给文件的第1个盘块的块号，如C1，依次类推，可在FAT表中得到文件的第3个盘块的块号，如C3。块内偏移为428。

【思考题1】

- 在索引分配方式中，可从文件的FCB中得到索引表的地址。从索引表的第3项（距离索引表首字节12字节的位置）可获得字节偏移量3500对应的物理块号，而块内偏移量为428。

【思考题2】

- 假定现在有一个空的完全没有存放数据的磁盘，大小为100KB。为了存储管理上的便利，我们人为的将这100KB的空间均分成100份，每份1KB。我们来依次存储这样几个文件：A.TXT(大小10KB)，B.TXT(大小53.6KB)，C.TXT(大小20.5KB)。请模拟FAT文件系统中数据的存储原则。

磁盘整体布局

整个100KB空间



文件分配表

文件分配表

簇号 对应数据	1	2	3	...	11	12	13	...	65	66	67	...	86	87	...	99
	目录	3	4	...	FF	13	14	...	FF	67	68	...	FF	00	...	00

文件目录

每行100B 共10行

A. TXT	2	10	2004.3.22 10:41	2004.3.22 10:41	只读	
B. TXT	12	53.6	1949:10:1 12:0	2003.8.22 20:40	隐藏	
C. TXT	66	20.5	2000:3:8 21:11	2005:3:8 9:11	系统	
⋮						

共10行记录

有效记录

内容留空

文件名(占50个字节)

开始簇(占4个字节)

文件大小(占10个字节)

创建日期、时间(占10字节)

修改日期、时间(占10字节)

读写属性(占4字节)

保留(12字节)

【思考题2】

- 若将b.txt删除以后，存入一个大小为60.3KB的d.txt，以上结构将如何变化？



整个100KB空间划分

文件分配表	目录	A.TXT	D.TXT	C.TXT	D.TXT	空
第0簇	第1簇	第2~11簇	第12~65簇	第66~86簇	第87~93簇	第94~99簇

文件分配表

簇号	1	2	3	...	11	12	13	...	65	66	67	...	86	87	88	...	93	94	...	99
对应数据	目录	3	4	...	FF	13	14	...	87	67	68	...	FF	88	89	...	FF	00	...	00

目录 每行100B 共10行

A.TXT	2	10	2004.3.22 10:41	2004.3.22 10:41	只读	
C.TXT	66	20.5	2000:3:8 21:11	2005:3:8 9:11	系统	
D.TXT	12	60.3	1999:5:1 8:00	2003:3:20 14:0	存档	
⋮						

共10行记录

内容留空

文件名(占50个字节)

开始簇(占4个字节)

文件大小(占10个字节)

创建日期、时间(占10字节)

修改日期、时间(占10字节)

读写属性(占4字节)

保留(12字节)

4.3.3 目录的实现

- 目录系统的主要功能
 - 从路径名到文件物理位置的转换
 - 文件各类属性的保存
- 目录表与目录项
 - 目录表：保存该目录下所有文件的属性信息
 - 目录项：保存一个特定文件的相关属性信息

4.3.3 目录的实现

- 检索目录文件的过程中，是否需要用到FCB中的所有信息？

只用到了文件名!!



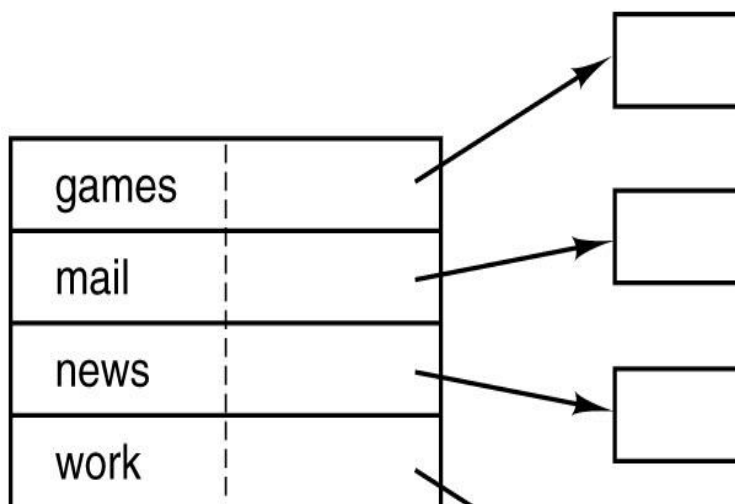
4.3.3 目录的实现

- 因此，可采用将文件名与文件描述信息分开存放的办法。即：
- 使文件描述信息单独形成一个称为索引结点的数据结构，简称为 **i** 结点。
- 文件目录中的每个目录项，仅由文件名和指向该文件所对应的 **i** 结点的指针所构成。

目录项的实例说明

games	attributes
mail	attributes
news	attributes
work	attributes

(a)



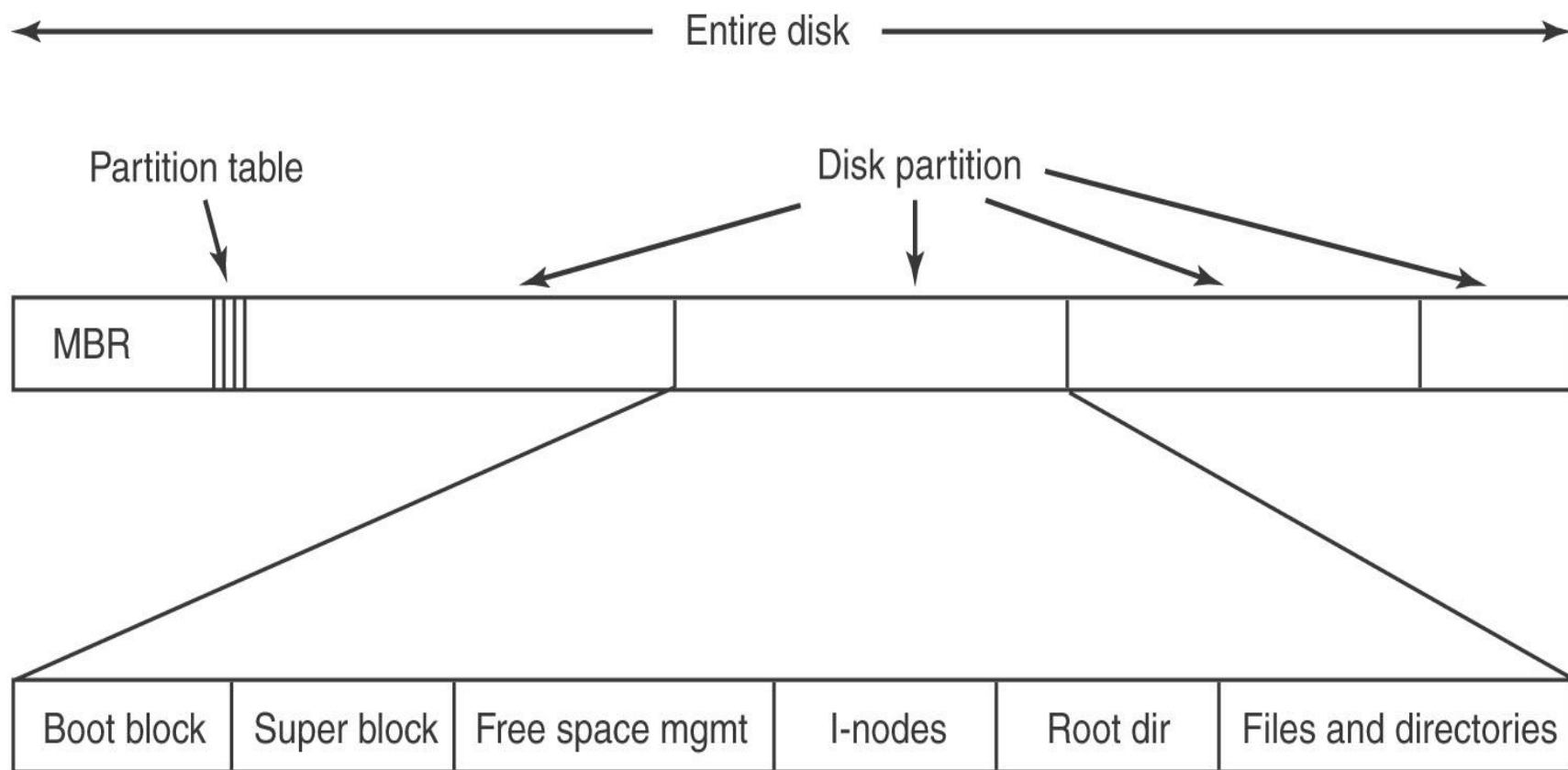
(b)

Data structure
containing the
attributes

4.3.3 目录的实现

- 在存储文件的外存空间中划分出一个固定的区域，用于保存所有文件的 i 结点。所有文件的 i 结点长度固定而且相同。

文件系统布局



4.3.3 目录的实现

- 将FCB分为两部分后，文件的目录项中仅包含一个文件名字和一个标识文件i结点的文件号，这样，根据文件名查找文件目录可以找到文件的目录项，根据目录项得到文件号可以找到文件的i结点，进而找到文件。

4.3.3 目录的实现

将文件的FCB划分为两部分具有如下两个主要的优点:

- 提高查找速度: 将FCB分为两部分之后, 一个外存块中可容纳较多的FCB, 从而大大地提高了文件的检索速度。
- 实现文件连接: 所谓连接就是给文件起多个名字, 这些名字都是路径名, 可为不同的用户所使用。

【例】

- 在某个文件系统中，每个盘块为512字节，文件控制块占64个字节，其中文件名占8个字节。如果索引结点编号占2个字节，对一个存放在磁盘上的256个目录项的目录，试比较引入索引结点前后，为找到其中一个文件的FCB，平均启动磁盘的次数。

【例】

- 在引入索引结点前，每个目录项中存放的是对应文件的FCB，故256个目录项的目录总共要占用 $256 * 64 / 512 = 32$ 个盘块。因此，在该目录中检索到一个文件，平均启动磁盘的次数为 $(1 + 32) / 2 = 16.5$ 次。

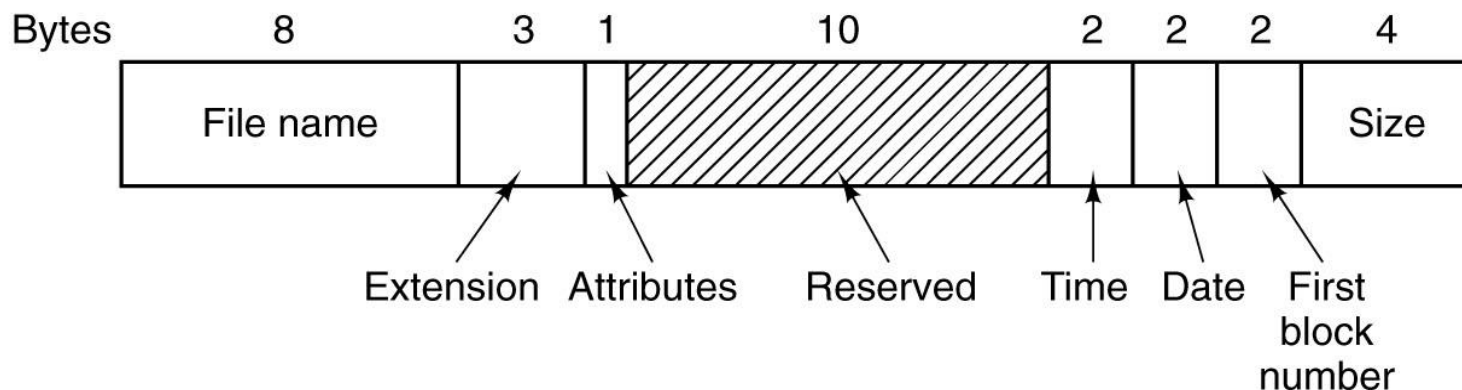
【例】

- 在引入索引结点之后，每个目录项中只需存放文件名和索引结点的编号，因此256个目录项的目录总共需要占用 $256 \times (8+2)/512 = 5$ 个盘块。因此找到匹配的目录项平均需要启动 $(1+5)/2 = 3$ 次磁盘；而得到索引结点编号后，还需启动磁盘将对应文件的索引结点读入内存，故平均需要启动磁盘4次。

小结

- 可见，引入索引结点，可大大减少启动磁盘的次数，从而有效地提高检索文件的速度。

目录实例说明——DOS中的目录



- 一个MS-DOS的目录项共32个字节长，其中包含了文件名、文件属性和第一个磁盘块的块号。根据第一个磁盘块的块号，顺着FAT表中的链，我们可以找到文件的所有块。

目录实例说明——UNIX中的目录

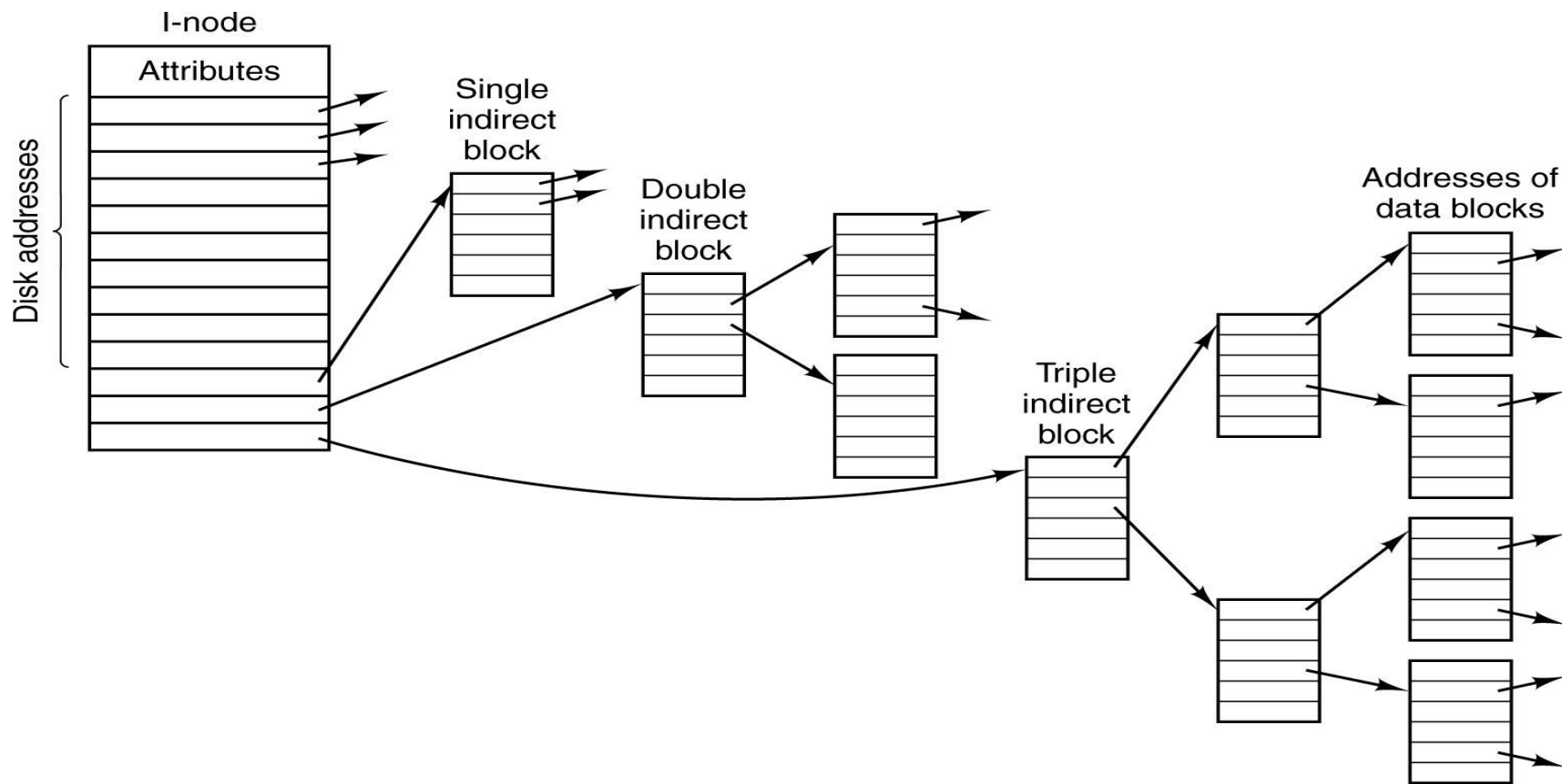
- UNIX 中使用的目录结构非常简单，如图所示，每个目录项只包含一个文件名及其i-节点号。有关文件类型、长度、时间、拥有者和磁盘块等所有信息都放在i-节点中。

2字节

14字节

i-节点号	文件名
-------	-----

一个UNIX的 i 节点



目录实例说明——UNIX中的目录

- 在打开文件时，文件系统必须根据给出的文件名找到它所在的磁盘块。
 - ① 首先文件系统找到根目录。在UNIX中，根目录的i-节点位于磁盘上的固定位置。
 - ② 然后在根目录中查找路径的第一部分从而也就获得了文件的i-节点号。因为i-节点都位于磁盘的固定位置，所以根据i-节点号找到i-节点是很直接的。

目录实例说明——UNIX中的目录

- ③ 利用这个i-节点，文件系统找到目录，并接着查找下一部分。当找到目录项后，得到目录的i-节点。从而找到目录并在该目录中查找文件。
- ④ 接着，文件的i-节点被读入内存，并保存在内存中，直至关闭该文件。

目录项的实例说明

	2	14
Unix中的目录项结构	i-结点号	文件名

查找/usr/ast/mbox
的过程说明

根目录	
1	.
1	..
4	bin
7	dev
14	etc
9	lib
6	usr
8	tmp

查找根目录，得知usr的i-结点号为6

6号i-结点内容	
模式	
大小	
时间	
132	

分析i-结点，得知usr在磁盘块132中

/usr目录内容	
6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast对应的i-结点号为26

26号i-结点内容	
模式	
大小	
时间	
406	

/usr/ast在磁盘块406中

/usr/ast目录内容	
26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox对应的i-结点号为60

4.3.4 共享文件

- 文件共享是指不同的用户或进程共同使用一个文件，实际上文件的实体只有一个。
- 目的:节省时间和存储空间，减少了用户工作量；进程间通过文件交换信息。



4.3.4 共享文件

- 共享形式：
 - 被多个用户使用，由存取权限控制
 - 被多个程序使用，但各用自己的读写指针
 - 被多个程序使用，但共享读写指针
 - 多个用户用相同或不同的名字来访问同一文件

共享的实现

- 物理复制——多处存储、多处出现。
- 一处存储、一处出现。
- 链接——一处存储，多处出现。



共享的实现

- 物理复制：各用户把该文件复制到自己的目录下使用。这样意味着外存物理空间上的重复存储浪费，而且各用户修改（各自目录下的）该文件时，会造成数据不一致性，一个用户看不到其他用户对该文件所做的修改，实际上最终变成多个内容不同的文件。

共享的实现

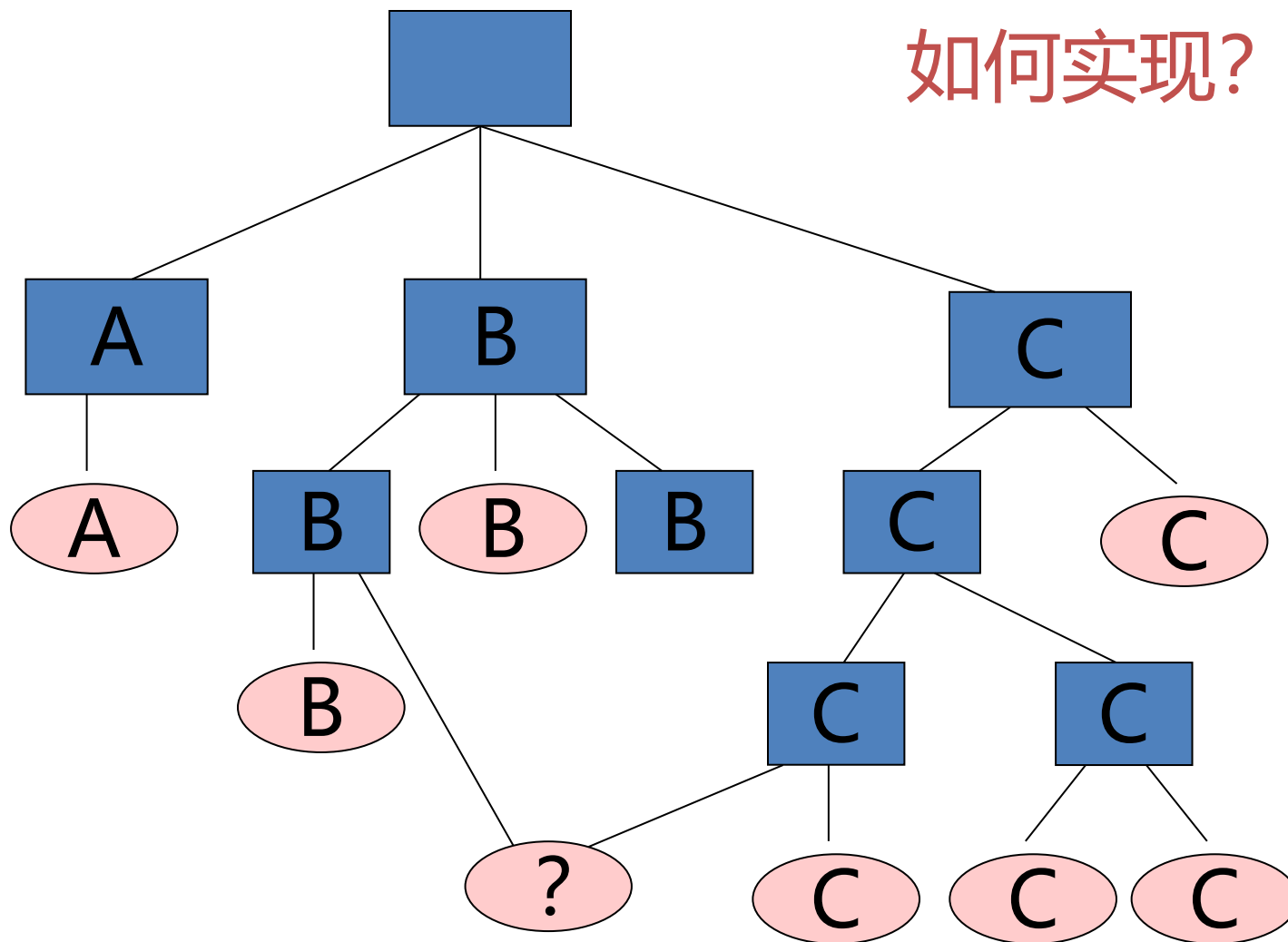
- 一处存储、一处出现。被共享的文件只在一处存储，且只在一个目录下使用，不同的用户使用该文件时都通过同一路径名来使用，该文件对不同的用户可以赋予不同的访问权限。这种处理避免了重复存储冗余浪费和数据不一致性，但每次使用都要写出路径名，而且用户不能按照自己的需要、习惯、命名体系来给文件命名。

共享的实现

- 链接：链接就是指一个文件或目录在目录树中多处出现，但实际上在外存介质上只有一份物理存储。简单地说，链接就是一个文件属于多个目录。链接避免了前两种办法的所有缺点：避免了重复存储冗余浪费和数据不一致性；不用每次使用共享文件都要写出路径名；用户可以按照自己的需要、习惯、命名体系来给共享文件命名。

一处存储，多处出现

如何实现？



4.3.4 共享文件

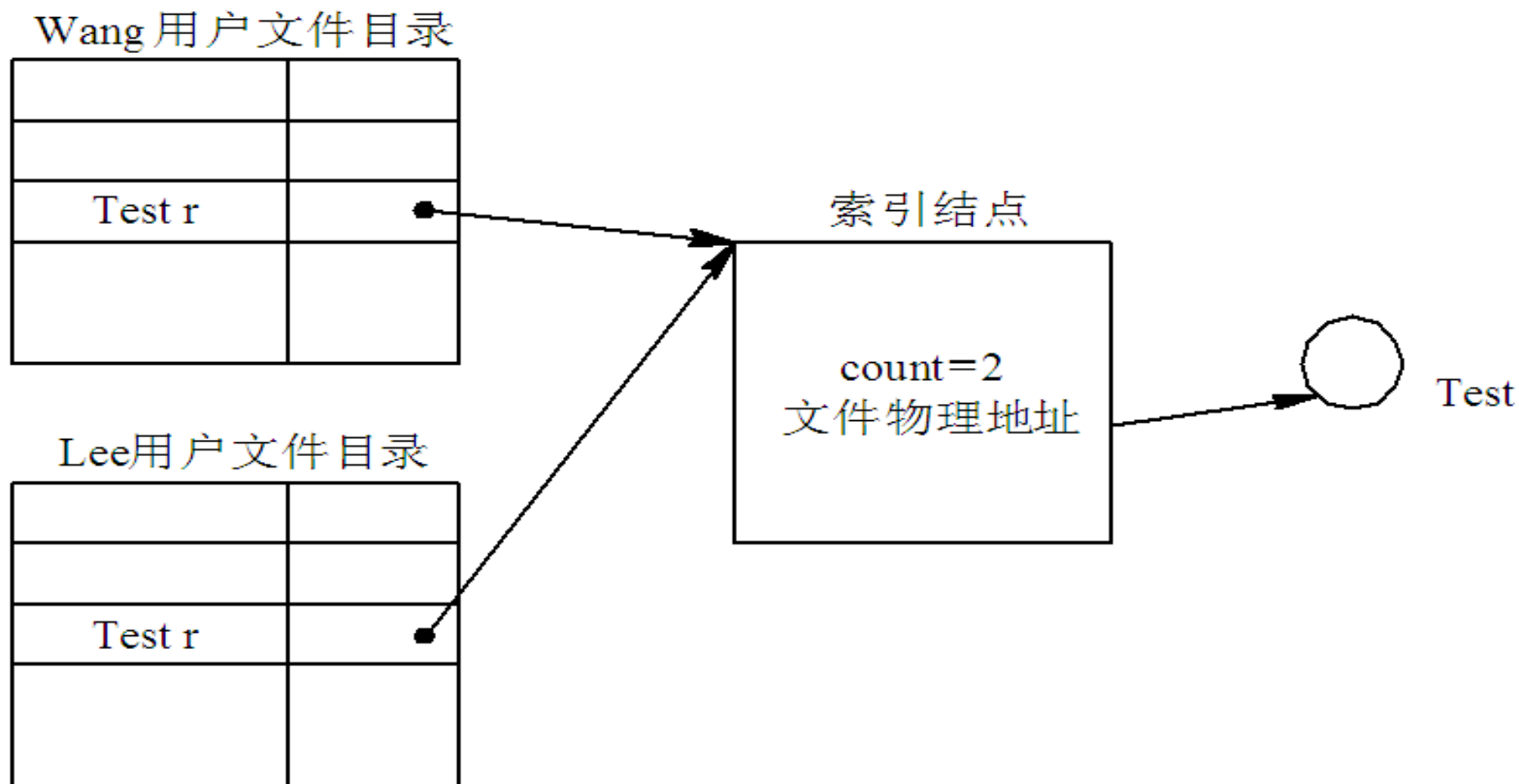
- 共享的实现：一个文件属于多个目录。
- 若不使用索引结点，即在文件目录中包含了文件的物理地址，存在什么问题？

容易出现文件的不一致性！

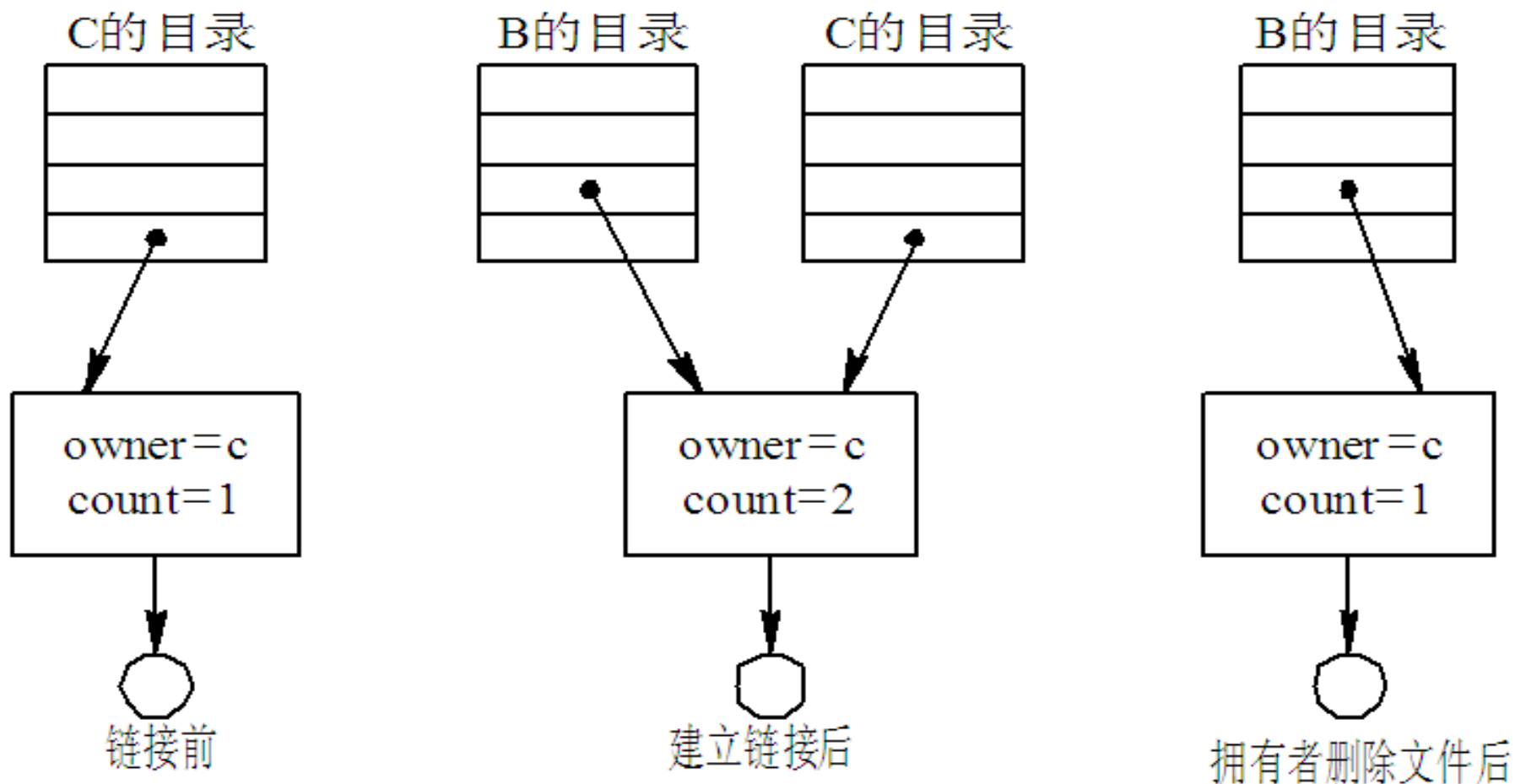
基于 i 结点的共享方式

- 将文件的物理地址和文件属性等信息放在索引结点中，在文件目录中，设文件名及指向索引结点的指针。另外在索引结点中增加链接计数 **count**，表示共享的用户数。删除时必须 **count=0**，方可。

基于i结点的共享方式



基于 i 结点的共享方式



利用符号链实现文件共享

- 共享某文件时，创建一新文件，并加到用户目录中，该文件仅包含被链接文件的路径名，称该链接方法为符号链接。该方式中，只有文件主才拥有指向其i结点的指针，其它共享的用户只有该文件的路径名。

利用符号链实现文件共享

- 并不记录链至文件的符号链接个数，符号链接文件也没有指针指向被链文件的i结点，符号链接文件有自己的i结点和文件主（即建立该符号链接的用户），其文件类型为符号链接文件。

目录/usr/joe的
目录文件内容

...
...
aa
...
...

文件/usr/joe/aa
的i节点内容

...
文件类型：普通文件
引用计数为1
盘块地址
...

文件/usr/joe/aa
的文件内容

学生信息
...

目录/usr/sue的
目录文件内容

...
...
bb
...
...

文件/usr/sue/bb
的i节点内容

...
文件类型：链接文件
引用计数为1
盘块地址
...

文件/usr/sue/bb
的文件内容

/usr/joe/aa

利用符号链实现文件共享

- 优点：方便地链接任一文件（用路径名）
- 缺点：访问共享文件时开销大（多次读盘，消费盘空间），每一共享文件都要增加一文件名（因路径名各不相同）

实例：Linux 中的文件链接

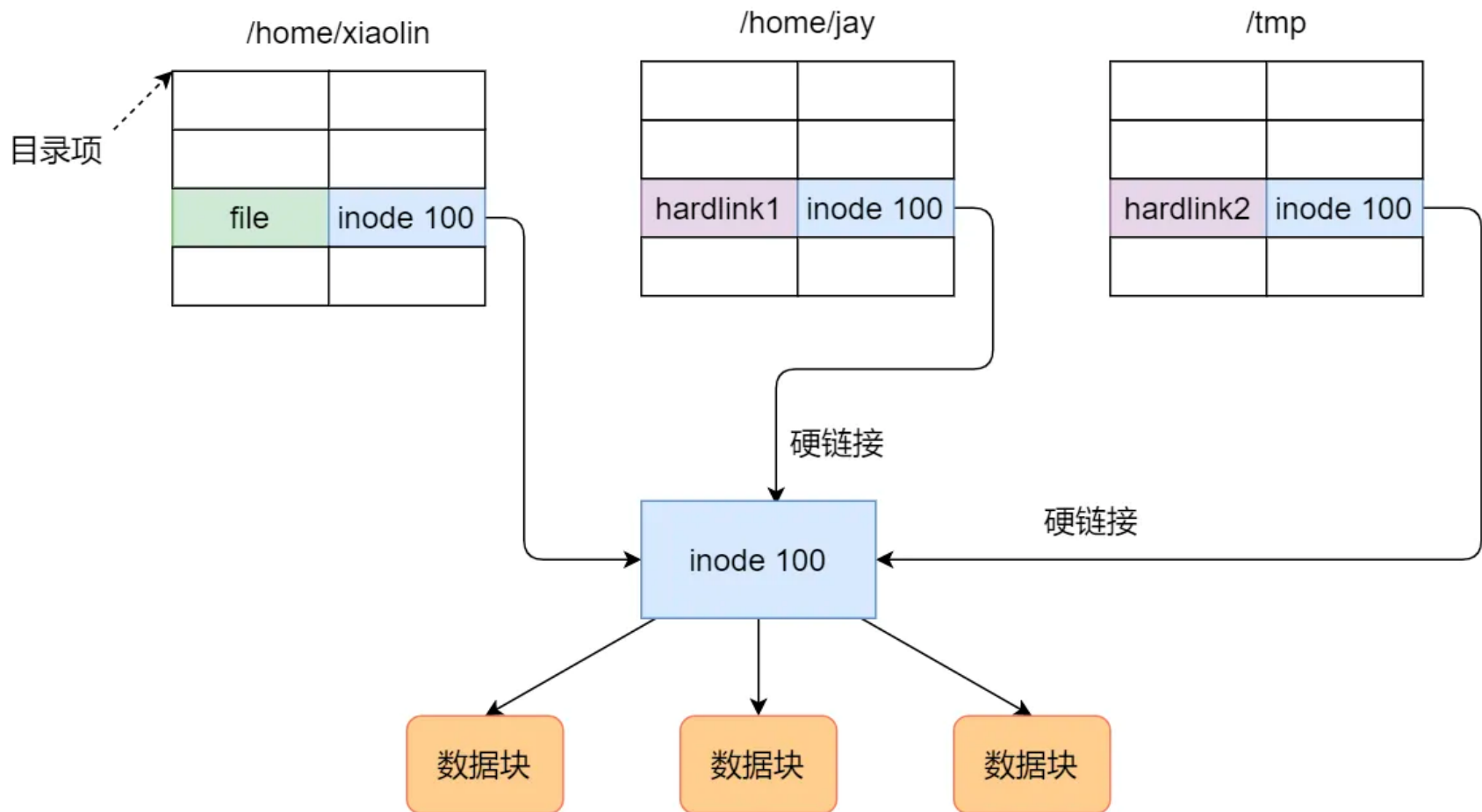
- 硬链接：记录的是目标的 inode

`ln -d existfile newfile`

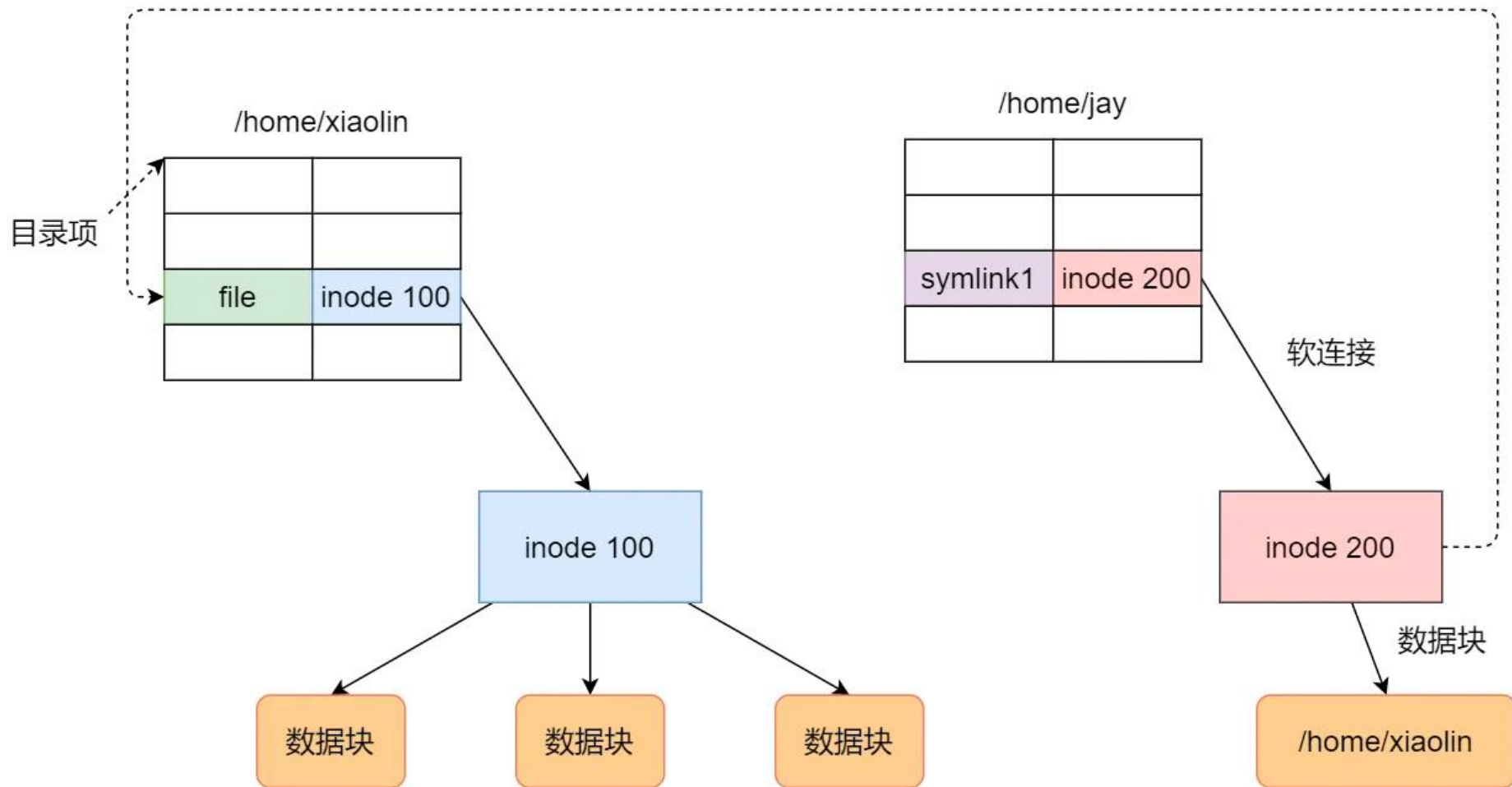
- 符号链接：记录的是目标的 path，即该文件包含了另一个文件的路径名。

`ln [-s] source_path target_path`

硬链接



软链接



4.3.5 磁盘空间管理

- 存储空间管理是文件系统的重要任务之一。只有有效地进行存储空间管理，才能保证文件共享的实现和文件按名存取。由于文件存储设备是分成若干个大小相等的物理块，并以块为单位来交换信息的，因此，文件存储空间的管理实质上是一个空闲块的组织和管理问题。

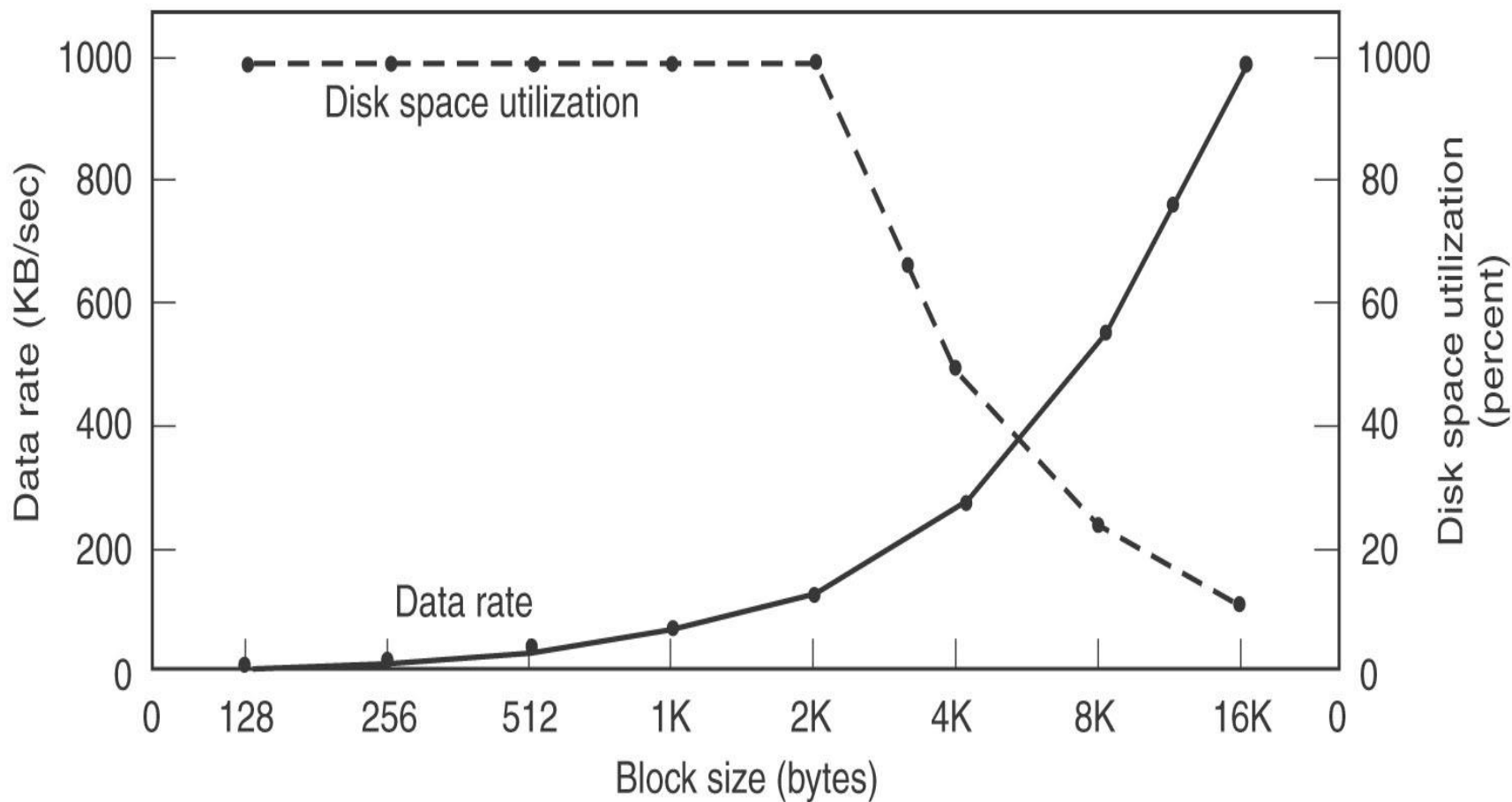
磁盘空间管理策略

- 空间单位划分
 - 字节序列：对磁盘空间不进行划分，管理效率低下
 - 块定义：类似于内存管理中的分页模式

磁盘空间管理策略

- 关键设计问题
 - 块大小的设计：空间利用效率与访问速度的权衡
 - 重点——影响磁盘访问速度的因素
 - 磁道容量、旋转时间、平均寻道时间

块大小定义与磁盘访问速度



磁盘空闲空间的管理

1. 位示图法

- 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, “1”表示对应的物理块已分配, “0”表示其对应的块未分配。
- 申请物理块时, 可以在位示图中查找为0的位, 返回对应物理块号
- 归还时, 将对应位转置0

圖示位

[illegible]

盘块的分配

(1) 顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位(“0”表示空闲时)。

(2) 将所找到的一个或一组二进制位，转换成与之相应的盘块号。假定找到的其值为“0”的二进制位，位于位示的第*i*行、第*j*列(*i*、*j*从1开始编号)，则其相应的盘块号应按下式计算（以上图为例）：

$$b = n(i-1) + j \quad // n \text{ 代表每行的位数。}$$

(3) 修改位示图，令 $\text{map}[i,j] = 1$ 。

盘块的回收

① 将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：

- $i = (b-1) \text{ DIV } n + 1$

- $j = (b-1) \text{ MOD } n + 1$

② 修改位示图。令 $\text{map}[i,j] = 1$ 。

位示图

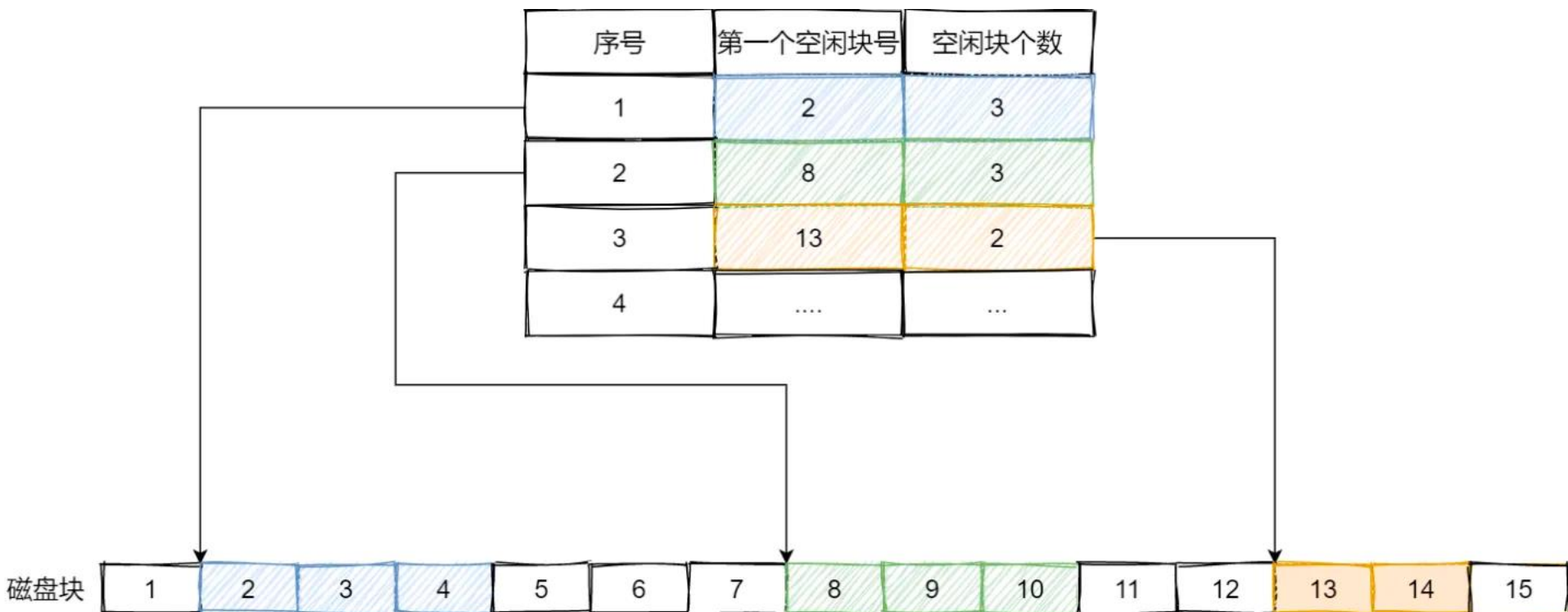
- 位示图对空间分配情况的描述能力强。一个二进制位就描述一个物理块的状态。
- 位示图占用空间较小，因此可以复制到内存，使查找既方便又快速。
- 位示图适用于各种文件物理结构的文件系统。

磁盘空闲空间的管理

2. 空闲块表（空白文件目录）

- 一个连续的未分配区域称为“空白文件”，系统为所有这些“空白文件”单独建立一个目录。每个空白文件，在目录中建立一个表目。表目的内容包括：第一空白物理块的地址（块号）、空白块的数目。

空闲块表



磁盘空闲空间的管理

- 当请求分配存储空间时，系统依次扫描空白文件目录的表目，直到找到一个合适的空白文件为止。
- 当用户撤消一个文件时，系统回收该文件所占用的空间。扫描目录，寻找一个空表目，并将释放空间的第一物理号及它所占的物理块数填到这个表目中。同时注意合并相邻空闲区。

空闲块表法

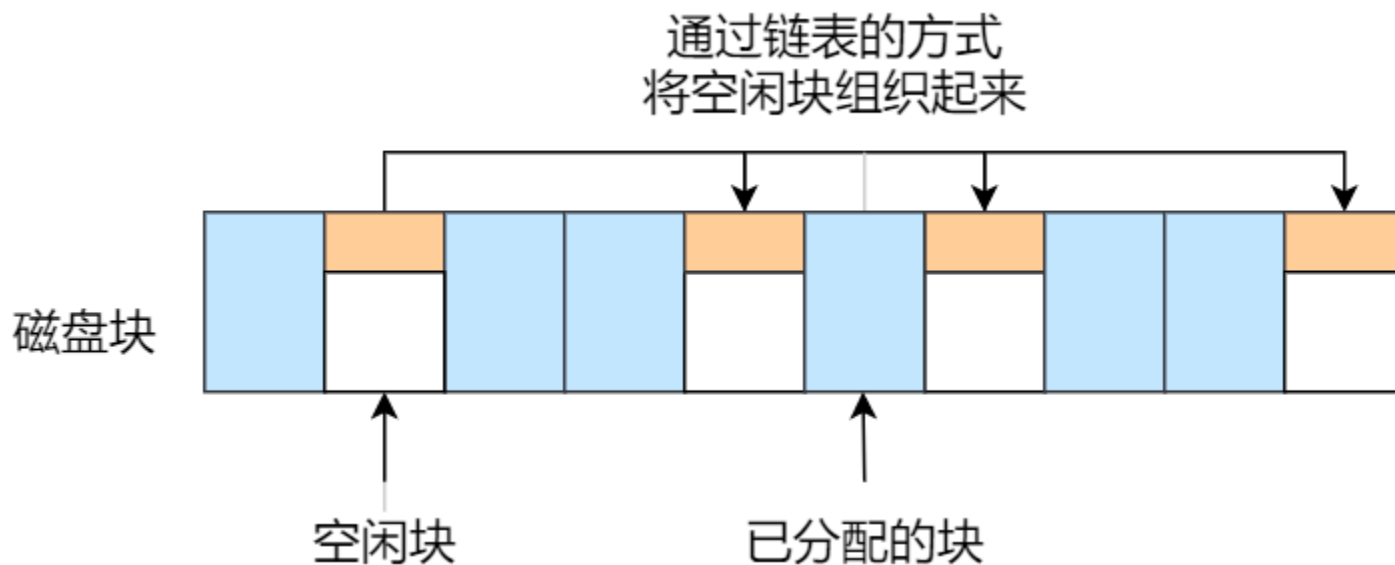
- 仅当有少量的空白区时才有较好的效果
- 如果存取空间中有着大量的小的空白区，
则其目录变得很大，因而效率大为降低。
- 这种分配技术适用于建立连续文件。

磁盘空闲空间的管理

3. 空闲链表法

- 即把所有的“空闲块”链在一起。
- 创建文件需要一个或几个物理块时，就从链头依次取下一块或几块。
- 回收文件时将回收物理块插入空闲链末尾。

空闲链表法



空闲盘块链



空闲盘区链



磁盘空闲空间的管理

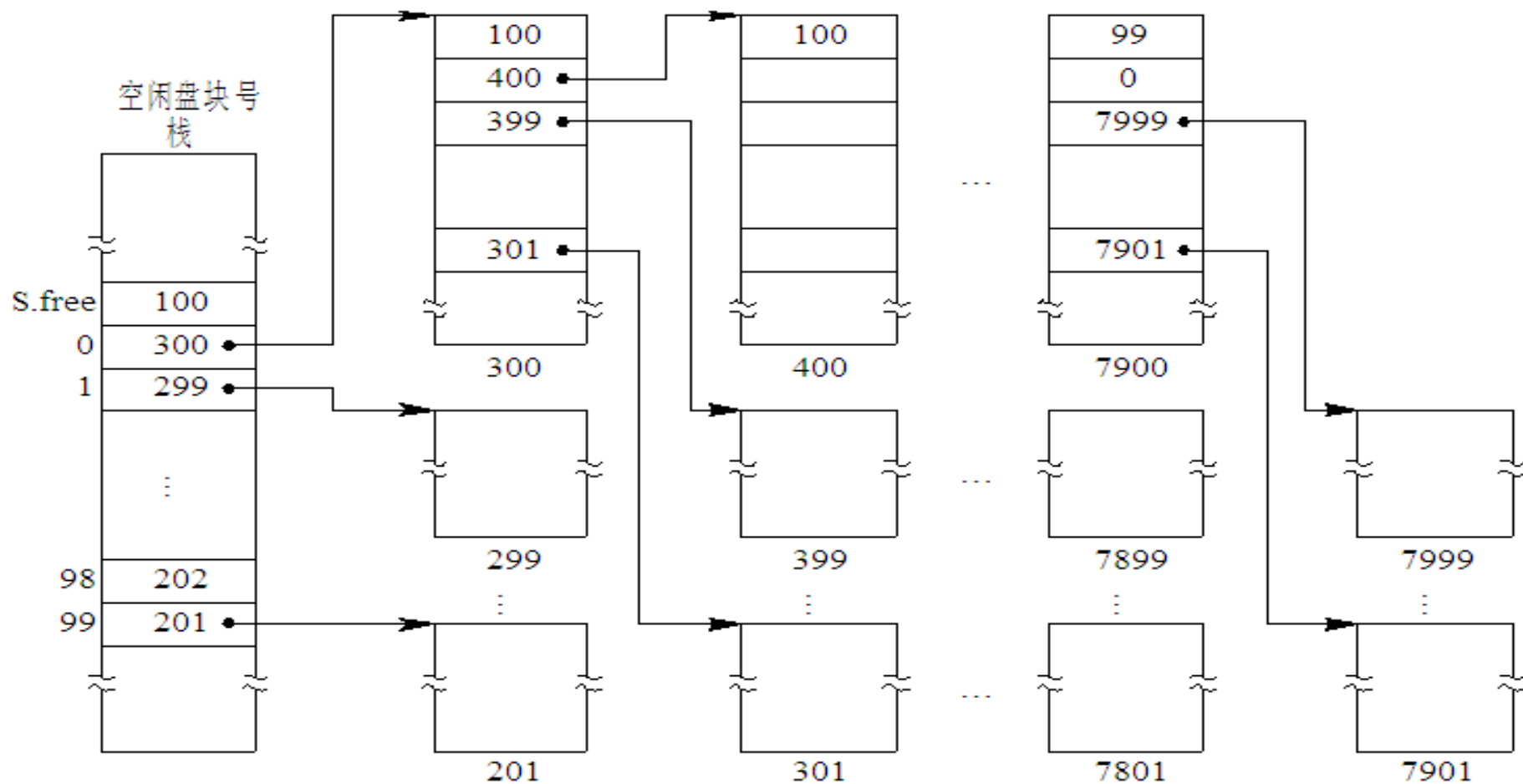
4. 成组链接法

- 空闲盘块号栈：用来存放当前可用的一组空闲盘块的盘块号以及栈中尚有的空闲盘块号数 N 。
- 文件区中的所有空闲盘块按固定大小(如每组100块)分成若干组，并将每一组的盘块数和该组所有的盘块号记入前一组的最后一个盘块中。

成组链接法

- 第一组的盘块数(可小于100)和该组所有的盘块号记入空闲盘块号栈中。
- 最后一组只有99个盘块，其盘块号分别记入其前一组最后一块的S.free(1)~S.free(99)中，而在S.free(0)中则放“0”，作为空闲盘块链的结束标记。

成组链接法



空闲盘块的分配

- 若空闲盘块栈中不止一块，则将空闲盘块栈中的空盘块数减1，并将空闲盘块栈栈顶的盘块分配出去；
- 若空闲盘块栈中只剩一块且栈顶的盘块号不是结束标记0，则先将该块的内容(记录有下一组的盘块数和盘块号)读入到空闲盘块栈中，然后将该块分配出去；
- 若栈顶的盘块号为结束标记0，则表示该磁盘上已无空闲盘块可供分配。

空闲盘块的回收

- 若栈中空闲盘块号数目不满100块，则只需将回收盘块的盘块号记入空闲盘块号栈的顶部，并执行空闲盘块数加1操作。
- 当栈中空闲盘块号数目已达100时，表示栈已满，便将现有栈中的100个盘块号，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

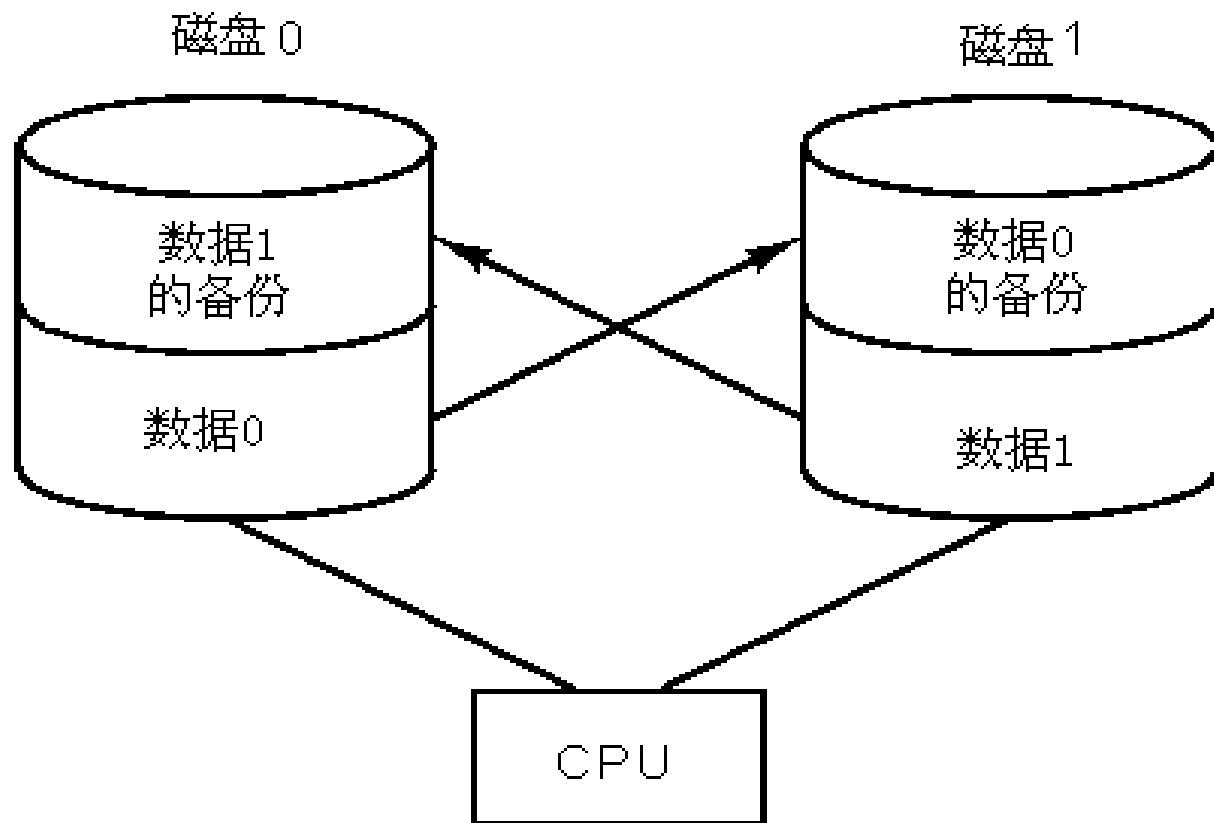
- 注意，空闲盘块栈是临界资源，对该栈的操作必须互斥进行，因此，系统为空闲盘块栈设置了一把锁，并通过上锁和解锁来实现对空闲盘块栈的互斥操作。

6.3.6 文件系统的可靠性

- 文件系统必须有防止硬、软件的各种可能破坏文件的能力。为此，文件系统经常采用建立副本和定时转储的方法来保护文件。

建立副本

- 把同一个文件保存到多个存储介质上，这些存储介质可以是同类的，也可以是不同类型的。
- 当对某个存储介质保管不善而造成文件信息丢失时，或当某类存储设备故障暂不能读出文件时，就可用其他存储介质上的备用副本来替换。
- 实现简单，但设备费用和系统开销增大。而且当文件需修改或更新时，必须要改动所有的副本。因此，这种方法一般用于短小且极为重要的文件。



将一个驱动器中的数据备份到另一个驱动器上,浪费了一半存储空间。

定时转储

- 每隔一定的时间就把文件转储到其他的存储介质上。当文件发生故障时，就用转储的文件来复原，把有故障的文件恢复到转储时刻文件的状态。这样，文件仅丢失了自上次转储以来新修改或增加的信息，可以从文件转储恢复后的状态开始重新执行。

4.3.6 文件系统的可靠性

- 影响文件系统可靠性的另一个问题是文件系统的一致性。许多文件系统读取磁盘块，进行修改后，再写回磁盘。如果在修改过的磁盘块全部写回之前，系统崩溃，那么文件系统可能出现不一致。如果一些未被写回的块是i-节点块、目录块或者包含空闲表的磁盘块时，这个问题尤为严重。

文件系统一致性

- 一致性检查分为两种：块的一致性检查和文件的一致性检查。
- 在检查块的一致性时，该程序建立两张表。每张表中，每块对应有一个计数器，初始值设为0。第一张表的计数器记录了每块在文件中出现的次数，第二张表的计数器记录了每块在空闲块链表(或空闲块位图)中出现的次数。

块的一致性检查

- 检验程序读取所有的i-节点，从i-节点开始，可以建立相应文件中使用的所有块的块号表。每当读到一个块号时，该块在第一张表中的计数器加1。接着这个程序检查空闲块链表或位图，查找所有未使用的块。每当在空闲表中找到一个块时，它在第二张表中的计数器加1。

块的一致性检查

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	0 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	1 0	0 0	0 1	1 0	0 0	1 1

(a) 正常情况

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	0 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	0 0	1 0	0 0	0 1	1 0	0 0	1 1

(b) 丢失了盘块

块的一致性检查

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	2 1	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	0 0	0 0	0 1	1 0	0 0	1 0

(c) 空闲盘块号重复出现

计数器组 \ 盘块号	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
空闲盘块号计数器组	1 1	0 1	1 0	1 1	1 0	0 1	1 1	0 0
数据盘块号计数器组	0 0	1 0	0 2	0 0	0 1	1 0	0 0	1 1

(d) 数据盘块号重复出现

目录系统的检查

- 要用到一张计数器表，每个计数器对应于一个文件。检验程序从根目录开始，沿着目录树递归下降，检查文件系统中的每个目录。对每个目录中的文件，其 i -节点对应的计数器加1。

目录系统的检查

- 当全部检查完成后，得到一张表，对应于每个i-节点号，表中给出了指向这个i-节点的目录数目，然后，检验程序把这些数字与存储在文件i-节点中的链接数目相比较。在一致的文件系统中，这两个数目相吻合。但是，有可能出现两种错误，i-节点中的链接数太大或太小。

4.3.7 文件系统性能

- 影响文件系统性能的因素?
 - 磁盘性能的好坏
 - 磁盘调度算法的好坏
 - 磁盘高速缓冲区的大小

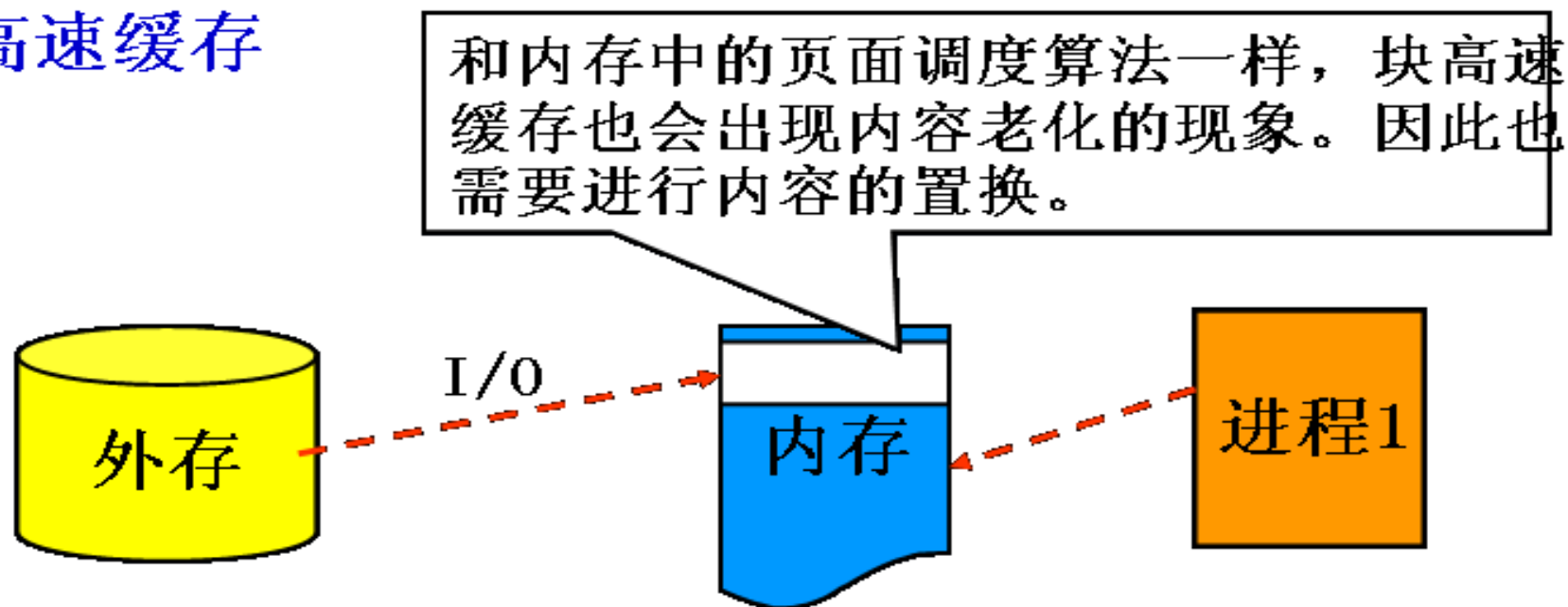
磁盘高速缓存

- 由于磁盘的I/O速度比内存的访问速度要低4~6个数量级，因此磁盘的I/O已成为提升计算机系统性能的瓶颈之一。于是，人们想方设法提高磁盘的I/O速度，其中最主要的技术之一就是采用块高速缓存（也称磁盘高速缓存）。

块高速缓存

- 逻辑上属于磁盘，实际上基于性能考虑保存在内存。

块高速缓存



磁盘调度

- 磁盘是一种共享设备，为了保证信息的安全，系统每一时刻只允许一个进程启动磁盘进行I/O操作，其余的进程只能等待。因此合理的设置调度算法就十分重要。
- 设置调度算法的目标：使各进程对磁盘的平均访问时间最小。
- 影响磁盘访问时间的因素
 - 移臂时间
 - 旋转时间



4.3.8 实现文件系统的表目

- 当用户申请打开一个文件时，系统要在内存中为该用户保存一些必要的信息，这些信息以表格栏目中内容的形式出现，被称为表目。在实现文件系统时所需要的表目有若干种，其中在内存中所需的重要表目有如下一些：
 - 系统打开文件表
 - 用户打开文件表

系统打开文件表

- 系统打开文件表，专门用于保存已打开文件的**FCB**。该系统打开文件表放在内存。除了保存已打开文件的**FCB**之外，在该表格中还保存有已打开的文件号、共享计数、修改标志等等。

系统打开文件表

i - 节点	文件号	共享计数	修改标志
...

用户打开文件表

- 在每个进程中，都有一个“用户打开文件表”。该表的内容有文件描述符、打开方式、读写指针、系统打开文件表入口等。另外在进程的进程控制块PCB中，还记录了“用户打开文件表”的位置。

用户打开文件表

文件描述符

打开方式	读写指针	系统打开文件表入口
...

用户打开文件表与系统打开文件表

- 用户打开文件表与系统打开文件表之间的联系：用户打开文件表指向系统打开文件表。如果多个进程共享同一个文件，则多个用户打开文件表目对应系统打开文件表中的同一入口处。

其它	打开方式	读写指针	系统打开文件表入口
...
...

进程 P1 的用户打开文件表

其它	打开方式	读写指针	系统打开文件表入口
...
...

进程 P2 的用户打开文件表

共享计数	其它
...	...
2	...
...	...

系统打开文件表



总结

文件系统层次模型

