



文件系统与文件操作

张路

2017.12





主要内容

- 存储器介绍
- 文件系统与文件操作
- I/O算法简介



存储介质的分类

■ 传统存储介质

■ 半导体存储器

- ◆ 动态随机存取存储器DRAM、静态随机存取存储器SRAM、只读存储器ROM、电子可擦除只读存储器EEPROM

■ 磁表面存储器

- ◆ 磁盘、磁带、磁芯、磁鼓

■ 光电存储存储器

- ◆ CD、DVD、蓝光

■ 新型存储介质

■ 相变存储器



不同存储器的特点

■ 感应

- 半导体、光、磁的差别
- 读、写的差别

■ 寻址

- 电子寻址
 - ◆ 容量的制约
- 机械寻址
- 寻址方式决定存储器是随机存储还是顺序存储
 - ◆ 电子寻址的存储器采用随机存储
 - ◆ 机械寻址的存储器采用顺序存储

■ 易失性

存储器的层次结构及访问方式

■ 三层结构

- 高速缓存、内存、外存

■ 显式访问

- 内存

- ◆ 变量

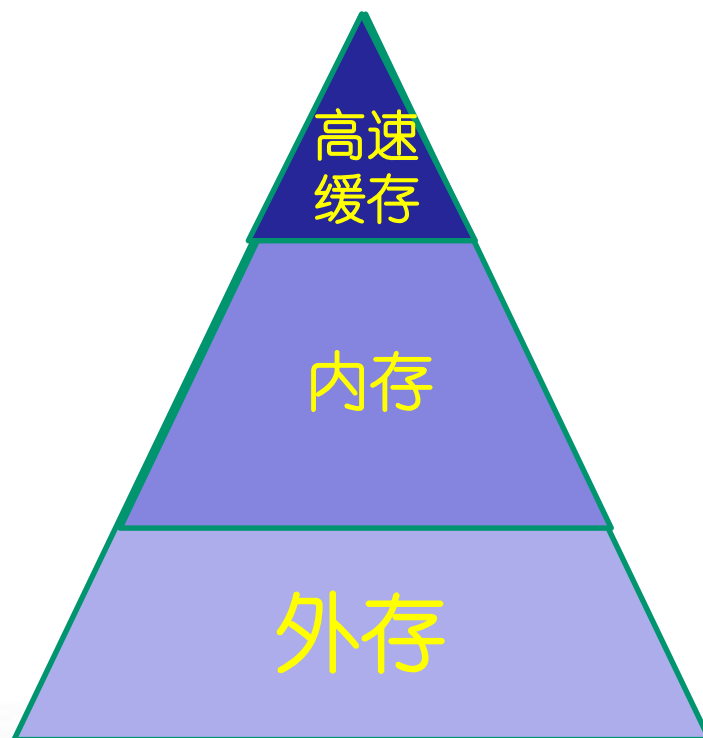
- 外存

- ◆ 文件

■ 透明访问


- 高速缓存

- 虚拟内存






主要内容

- 存储器介绍
 - 文件系统与文件操作
 - I/O算法简介
- 



文件系统

- 文件是由操作系统来管理的，包括文件的结构、文件的命名、文件的使用、文件的保护和文件的实现等等，都是在操作系统的设计当中需要解决的问题。
 - 总之，在一个操作系统当中，负责处理文件相关事宜的部分，就称为是文件系统。
- 



文件系统的两个视角

可以从两种不同的观点来看待文件系统：

- ◆ **用户观点：**关心的是文件系统所提供的对外的用户接口，包括文件如何命名、如何保护、如何访问（创建、打开、关闭、读、写等）；
- ◆ **操作系统观点：**关心的是如何实现与文件有关的各个功能模块，包括如何来管理存储空间、文件系统的布局、文件的存储位置等。



文件系统的目标

- **方便**的文件访问和控制：以符号名称作为文件标识，便于用户使用；
- **并发**文件访问和控制：在多道程系统中支持对文件的并发访问和控制；
- **统一**的用户接口：在不同设备上提供同样的接口，方便用户操作和编程；
- 多种文件访问**权限**：在多用户系统中的不同用户对同一文件会有不同的访问权限；
- **优化性能**：存储效率、检索性能、读写性能；
- **差错恢复**：能够验证文件的正确性，并具有一定的差错恢复能力；



文件系统的基本概念（1）

- 文件是具有符号名的数据项的集合。文件名是文件的标识符号。文件包括两部分：
 - 文件体：文件本身的信息；
 - 文件说明：文件存储和管理信息；如：文件名、文件内部标识、文件存储地址、访问权限、访问时间等；
- ◆ 文件是一种抽象机制，它提供了一种把信息保存在磁盘等存储设备上，并且便于以后访问的方法。抽象性体现在用户不必关心具体的实现细节。
- ◆ 可以视为一个单独的连续的逻辑地址空间，其大小即为文件的大小，与进程的地址空间无关。



文件系统的基本概念 (2)

■ 文件系统

- 文件系统是操作系统中管理文件的机构，提供文件存储和访问功能
- 是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

■ 目录

- 目录是由文件说明索引组成的用于文件检索的特殊文件



文件系统的基本概念 (3)

- 文件命名的一般格式为
文件名 · [文件扩展名]

通常对文件名的长度，字符、首字符、是否区分大小写有规定

扩展名用于表示文件的类型

例子：.bak .c .gif .jpg

.hlp .html .mpg .o

.ps .tex .txt .zip



文件系统的基本功能

- 统一管理文件的存储空间，实施存储空间的分配与回收
- 实现文件的按名存取
名字空间 ^{映射} → 存储空间
- 实现文件信息的共享，并提供文件的保护和保密措施
- 向用户提供一个方便使用的接口（提供对文件系统操作命令，以及提供对文件的操作命令：信息存取、加工等）



文件分类（1）

■ 按文件性质和用途分类

■ 系统文件：

- ◆ 有关OS及各种系统应用程序和数据所组成文件
- ◆ 只允许用户通过系统调用来执行它们，不允许进行读写和修改

■ 用户文件：

- ◆ 用户委托文件系统保存的文件
- ◆ 由文件的所有者或所有者授权的用户才能使用
- ◆ 源程序、目标程序、用户数据文件、用户数据库等

■ 库文件：

- ◆ 标准子程序及常用应用程序组成文件
- ◆ 允许用户使用但不能修改，如C语言子程序库等



文件分类 (2)

■ 按文件的组织形式分类

■ 普通文件

- ◆ 文件的组织格式为文件系统中所规定的最一般的格式，如字符流组成的文件
- ◆ 包括：系统文件、用户文件、库函数文件等

■ 目录文件

- ◆ 由文件目录构成的特殊文件
- ◆ 含有文件目录信息的一种特定文件，主要用来检索文件的目录信息

■ 特殊文件

- ◆ 形式上与普通文件相同，也可进行查找目录等操作
- ◆ UNIX类系统，输入输出设备看作是特殊文件



文件属性

- 每个文件都有一个名字和它所保存的信息，此外，OS 还给每个文件附加了一些其他的信息，这些信息称为文件的属性。
 - 文件属性反映文件的类型、存取控制等
- 常见的一些文件属性：
 - 保护信息：谁可以对该文件进行何种操作；
 - 创建者：该文件是谁创建的；
 - 只读标志位：0表示可读/写，1表示只读；
 - 隐藏标志位：0表示普通文件，1表示隐藏文件；
 - 系统标志位：0表示普通文件，1表示系统文件；
 - 创建时间、最后修改时间；
 - 文件长度。

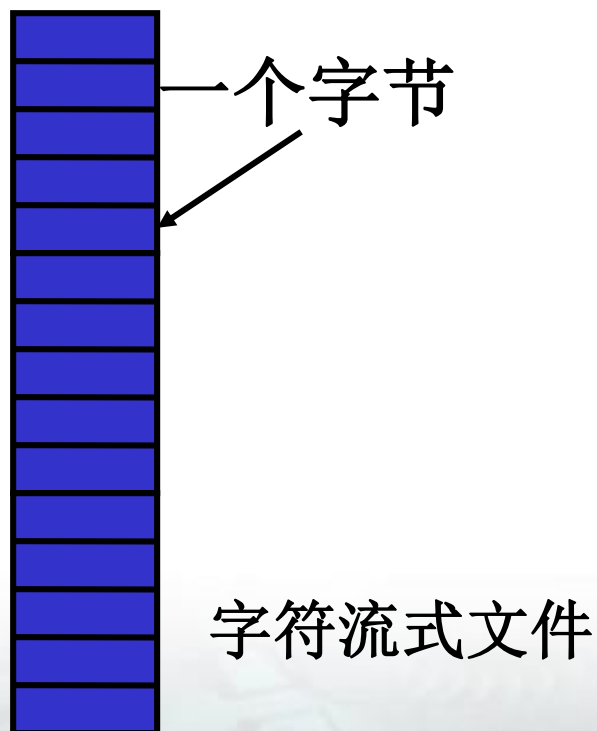


文件逻辑结构

- ◆ 文件的逻辑结构，即文件系统提供给用户的文件结构形式，它独立于在外存上的物理存储结构。
 - ◆ **无结构文件**：整个文件由一序列无结构的字节流组成，故又称为流式文件；
 - ◆ 简单的记录结构：
 - ☞ 每一行（如80字符）作为一个记录；
 - ☞ 记录的长度固定；
 - ☞ 记录的长度可变；
 - ◆ **复杂结构**：文件的结构形式比较复杂，例如，把文件中的记录组织成一棵树的形式。
- 有结构文件，是指由一个以上的记录构成的文件，故又把它称为记录式文件

文件结构—字符流文件

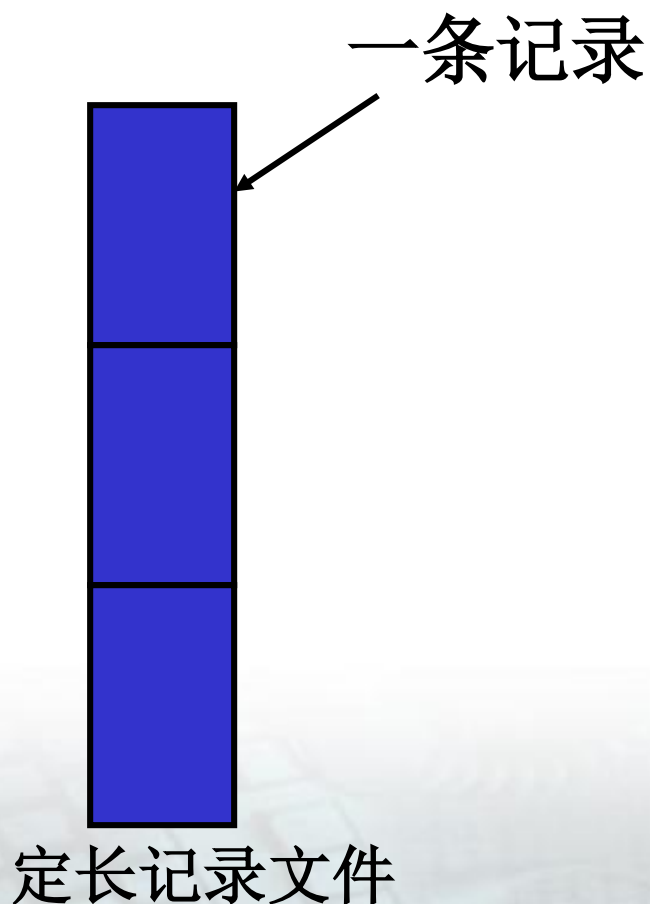
■ 无结构的字符流式文件



- 构成的基本单位是：字符
 - 文件是有逻辑意义的、无结构的一串字符的集合
 - 长度为该文件所包含的字符个数
 - 其内在含义由使用该文件的程序自行理解
 - 文件体为字节流，不划分记录，顺序访问，每次读写访问可以指定任意数据长度。
 - 当前操作系统中常用的文件组织，源程序、目标代码等文件
- 好处：提供很大的灵活性

文件结构—定长记录文件

■ 定长记录文件



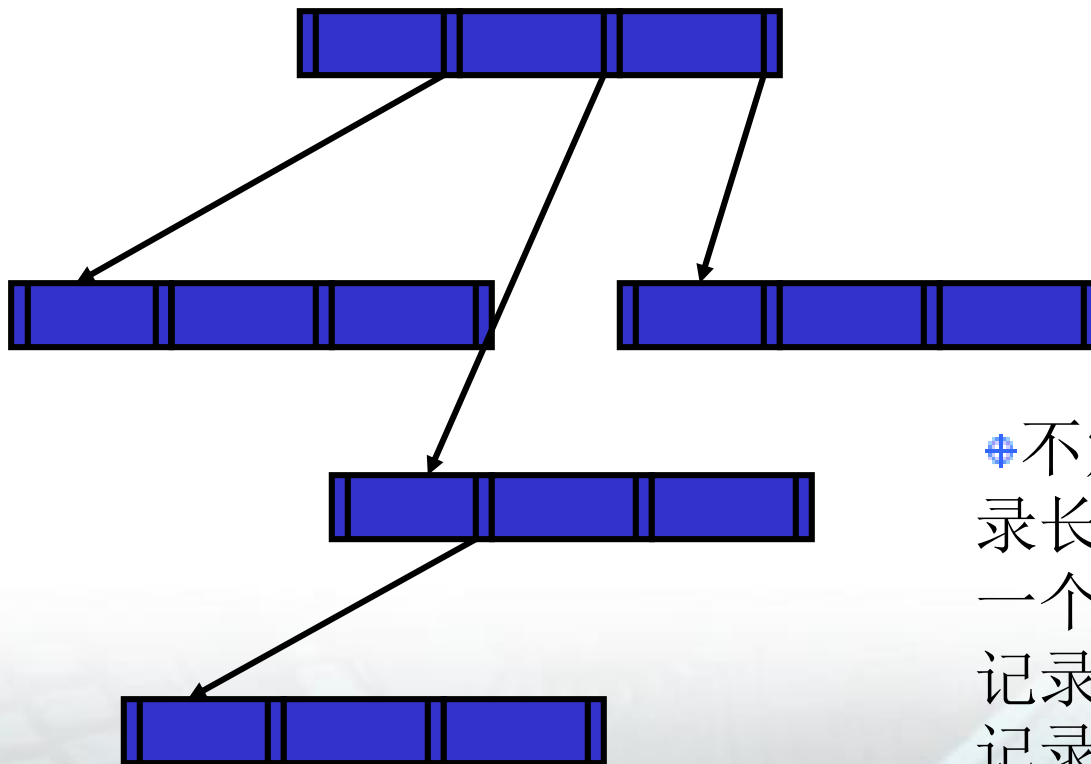
⊕记录式文件：一组有序记录的集合，是有结构的文件

⊕记录：具有特定意义的信息单位，由该记录在文件中的逻辑地址与记录名所对应的一组键、属性及属性值所组成

⊕定长记录文件：各个记录长度相同，可以根据记录号 i 及记录长度 L 确定给记录的逻辑地址

文件结构—不定长记录文件

■ 不定长记录文件构成的记录树



✦不定长记录文件：各个记录长度不相同，必须从第一个记录起一个记录一个记录查找，直到找到所需记录



文件结构比较

■ 两种文件的比较

- 流式文件就象给一张白纸给用户，用户可将他的信息任意地写到纸上，没有任何格式上的限制。
- 记录式文件就象给一张表格给用户，用户要按表规定的格式填信息。
- 显然，结构式文件对用户的限制很大，使用起来就不方便，所以记录式文件被淘汰是理所当然的



文件物理结构（1）

- 从系统的角度来看文件
 - 文件的物理结构：文件在物理介质上的存放方式
- 具体而言，以块为单位，研究如何把文件的一个个连续的逻辑块存放在不同的物理块当中
 - 即研究逻辑块与物理块之间的映射关系
 - ◆ 类似于页式存储管理方案当中逻辑页面与物理页面的映射关系
- 主要有三种物理结构：连续结构、链表结构和索引结构。



文件物理结构（2）

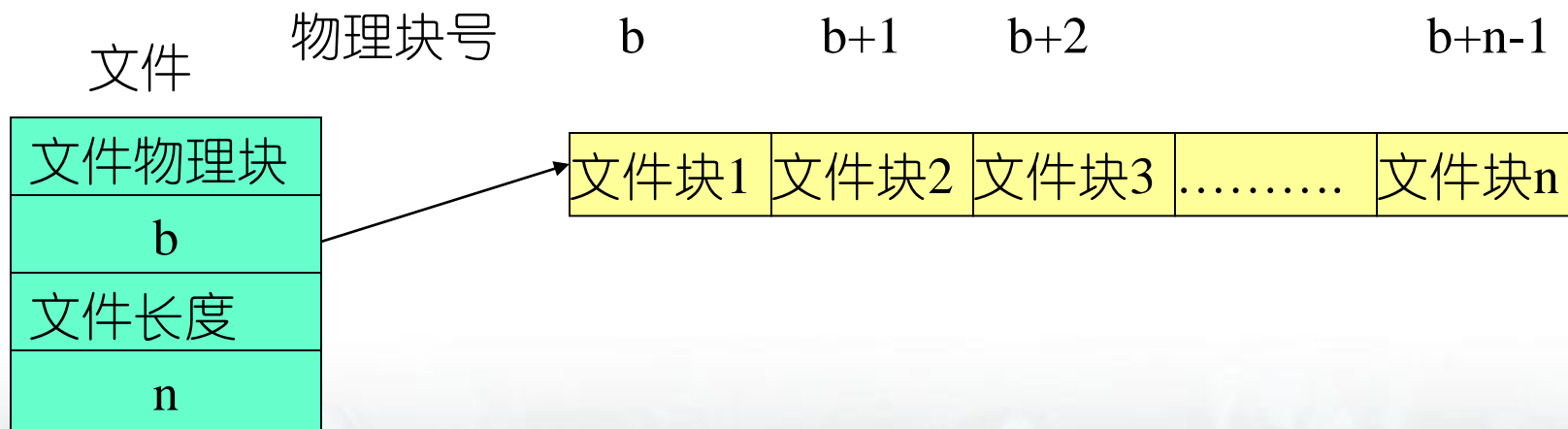
■ 物理块

- 在文件系统中，文件的存储设备常常被划分为若干大小相等的物理块。同时也将文件信息划分成相同大小的块。同一盘符上所有块统一编号
- 操作系统以块为单位进行磁盘空间的分配
- Windows 中，物理块被称为“簇”。簇的大小是扇区的整数倍，一般来说硬盘越大簇就越大
- 由于“块”是磁盘空间分配的单位，所以每个文件占用的磁盘空间体积一定是“块”的整数倍

顺序结构文件 (1)

■ 连续结构 (顺序结构)

- 把文件的各个逻辑块按照顺序存放在若干个连续的物理块当中。
- 一个文件的信息存放在若干连续的物理块中





顺序结构文件 (2)

■ 优点:

- 简单、易于实现，只需记住第一个物理块的编号和物理块的个数，即实现从逻辑块到物理块的映射关系：假设文件总共有 N 个逻辑块，第一个逻辑块存放在第 X 个物理块当中，则第 i 个逻辑块保存在第 $X + i - 1$ 个物理块中；
- 由于物理块是顺序存放的，所以在访问文件时，只需将磁头定位到第一个物理块，即可顺序地读取，因而速度很快；
- 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。



顺序结构文件（3）

■ 缺点：

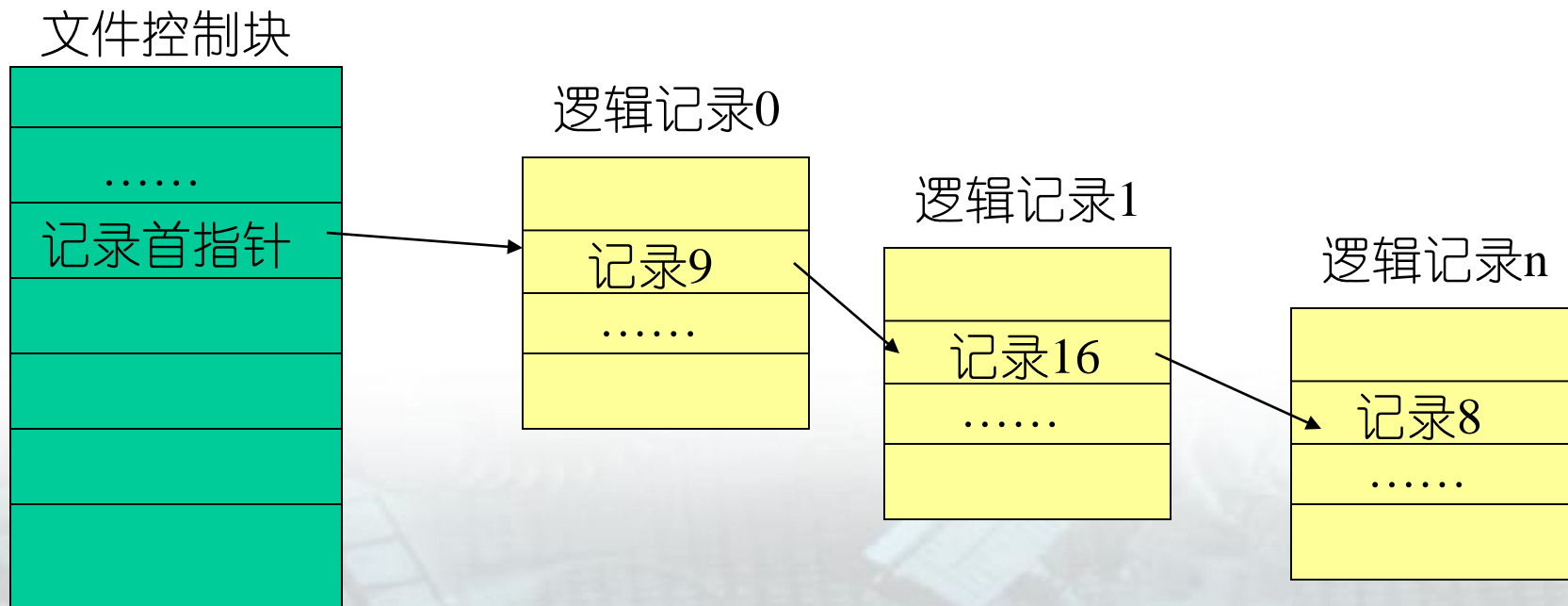
- 随着磁盘文件的增加和删除，将形成已占用物理块与空闲物理块交错的情形，那些较小的、无法再利用的物理块，即成为外碎片。可以使用存储紧缩技术，但代价高昂；
- 文件的大小不能动态的增长，对于在创建之时未知大小的文件，预留空间少了则不够用，预留多了则浪费；
- 不利于文件插入和删除

■ 用于早期的文件系统，但在当今的CD-ROM、DVD和其他一些一次性写入的光学存储介质中，有着广泛的应用。

链接结构文件 (1)

■ 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





链接结构文件（2）

■ 特点：

- 在每个物理块当中，必须利用若干个字节，来作为指针，指向下一个物理块；
- 只需记住链表结构的首结点指针（也可再加上尾结点），即可定位到文件的每一个物理块；

■ 优点：提高了磁盘空间利用率，不存在碎片问题

- 每一个物理块都能够用上，克服了连续结构的所有问题，不存在外碎片问题（最后一个物理块有内碎片），而且文件的大小也可以动态地变化。
 - ◆ 有利于文件插入，删除和动态扩充



链接结构文件 (3)

■ 缺点：

- 只能进行顺序访问，不能进行随机访问。为了访问一个文件的第 n 个逻辑块，必须从文件的第一个物理块开始，按照指针所指向的地址，顺序地访问前 n 个块，即为了得到第 n 个逻辑块的物理块地址，必须先访问 $n-1$ 次的磁盘，速度非常慢；
- 每个物理块上的数据存储空间不再是2的整数次幂（指针占用了若干个字节），从而计算文件中某个字节的位置比较复杂，使文件的访问不太方便。
- 可靠性问题，如指针出错
- 更多的寻道次数和寻道时间



索引结构文件（1）

■ 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

■ 特点：

- 每一个文件都有一个索引结点；
- 直接实现逻辑块与物理块的映射；
- 只需把那些正被使用的文件的索引结点装入内存即可，不必象FAT表那样须全部装入。



索引结构文件 (2)

- 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
如：内外存空间，存取时间



索引结构文件 (3)

■ 索引表应该多大？如何组织？

索引表组织：

■ 链接模式

- 一个物理块一个索引表项, 多个索引表项链接起来
- FAT采用链接模式

■ 多级索引

- 将一个大文件的所有索引表（二级索引）的地址放在另一个索引表（一级索引）中
- Unix文件系统采用的是多级索引结构——I结点



索引结构文件（4）

- 带有文件分配表的链表结构，即单级索引结构(FAT 表)——索引结构变种1
- 基本思路：在链表结构的基础上，把**每一个物理块当中的链表指针抽取出来，单独组成一个表格**，即文件分配表（File Allocation Table, FAT），并把它放在**内存**当中。
- 特点：
 - 实现了链表指针与文件数据的分离，整个物理块都可以用来存放数据；
 - FAT表放在内存当中，因此，若要随机地访问文件的第n个逻辑块，可以先从FAT表中查到相应的物理块编号（地址），然后再去访问外存，这样只需访问一次外存，速度快。



索引结构文件 (5)

■ 如何来实现FAT？

- 想想在页式存储管理当中，如何实现逻辑页面到物理页面的映射？
- FAT表的实现方法：有点类似于反置页表。
 - ◆ 在文件系统当中，设置一个一维的线性表格，表格项的个数等于磁盘上物理块的个数，并按物理块编号的顺序来建立索引。
 - ◆ 对于每一个文件来说，在它的FCB中记录了该文件的第一个物理块编号X1，而在FAT表的第X1项中，记录了该文件的第二个物理块编号X2，在FAT表的第X2项中，记录了该文件的第三个物理块编号X3，...，从而形成一个链表
 - ◆ 在最后一个FAT表项存放了一个特殊的文件结束的标识。



索引结构文件 (6)

■ 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
- 充分利用外存空间



索引结构文件 (7)

- UNIX文件系统采用的是多级索引结构：i节点（i-node, index node）

——索引结构变种2


- 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



- 文件太多了怎么办？不同的应用程序有不同类型的文件，不同的用户有不同的文件，如何对它们进行组织、分类？
- 如何对文件进行管理？当用户需要访问某个文件时，如何根据这个文件名迅速地定位到相应的文件，从而对文件的属性和内容进行各种操作？
- 解决的办法就是：目录。



目录管理

- 目录是由文件说明索引组成的用于文件检索的特殊文件。
 - 文件目录的内容主要是文件访问的控制信息（不包括文件内容）。
 - 它是一张记录所有文件名及其存放地址、文件的说明和控制信息的表格。
 - 对于目录包含的每个文件，在此目录文件中有一个表项（称为：该文件的目录项）
- 



目录管理的目标

- **效率**：要能迅速定位一个文件；
- **命名**：要方便用户；
 - **重名**：对不同的用户文件，可使用相同名字；
 - **别名**：对同一个文件，可有多个不同的名字；
- **分组功能**：能够把不同的文件按照某种属性进行分组（如某门课程的所有文件可分为讲义、作业、参考资料等不同的组）。



文件目录的基本概念（1）

- **文件目录**：把所有的文件控制块（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



文件目录的基本概念 (2)

■ 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有关信息

- 文件控制块的内容：

- ◆ 文件名，文件号，用户名，文件地址，文件长度，文件类型，文件属性，共享计数，文件的建立日期，保存期限，最后修改日期，最后访问日期，口令，文件逻辑结构，文件物理结构等

■ 文件控制块是文件存在的标志

■ 文件与文件控制块一一对应



文件控制块

1. 基本信息类。包括文件名，文件物理位置，文件逻辑结构，文件的物理结构。
2. 存取控制信息类。包括文件主的存取权限，核准用户的存取权限和一般用户的存取权限。
3. 使用信息类。包括文件的建立日期和时间、文件上次修改的日期和时间及当前使用信息。



文件目录——目录内容

文件目录项中存储了文件属性信息(properties), 其中的一部分是用户可获取的。

1. 文件名

2. 文件的大小, 单位: 字节

3. 文件在物理存储介质中的位置。
取决于文件的物理结构。

对于连续文件: 文件起始块号 (即文件的第一个物理块块号);

对于串联文件: 指向第一个物理块的指针;

对于索引文件: 索引表。

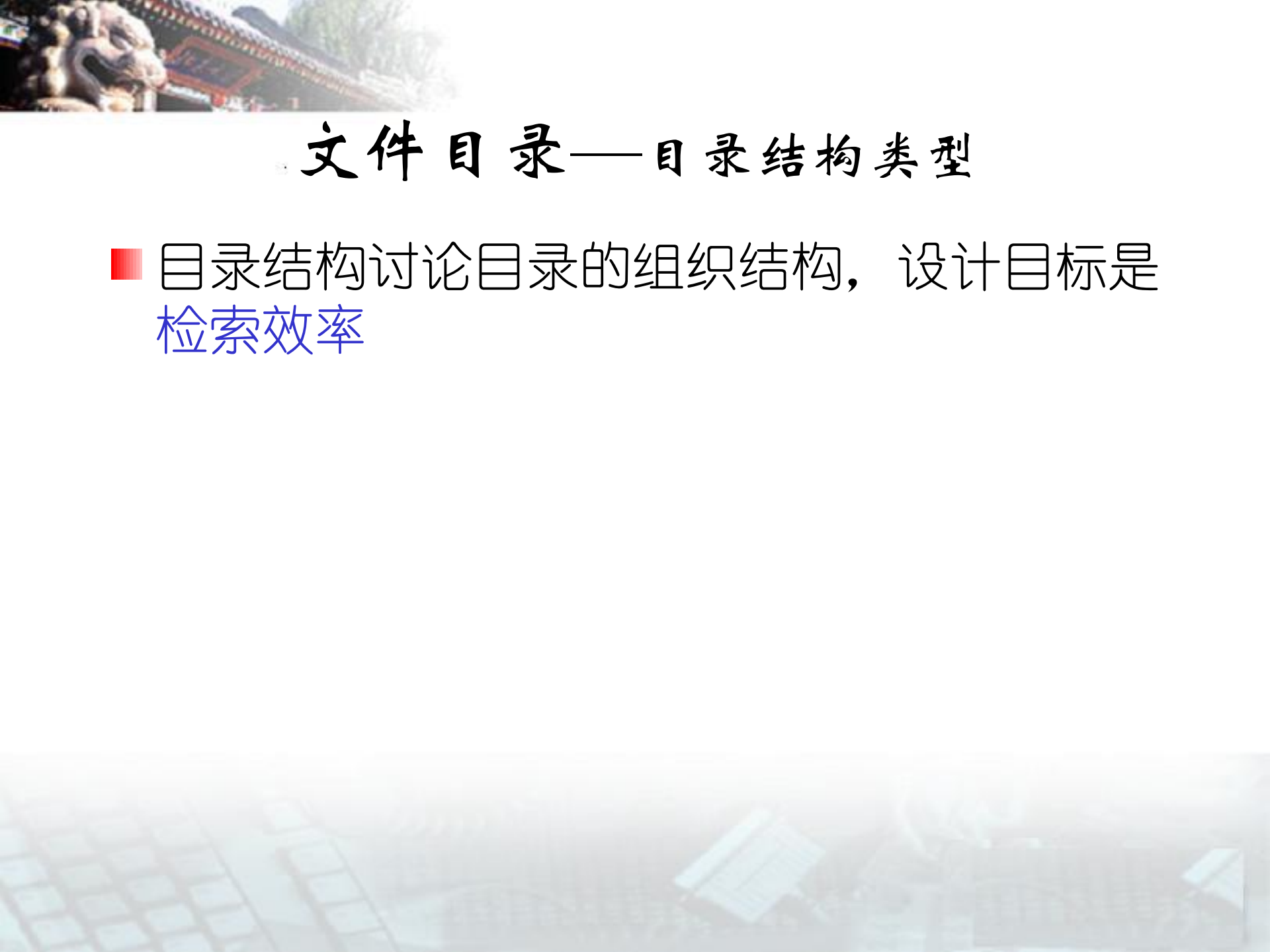
4. 存取控制信息

文件主和其它用户对该文件的访问权限。

5. 管理信息

包含文件创建的日期和时间, 最近修改该文件的日期和时间等。

6. 文件的类型



文件目录——目录结构类型

- 目录结构讨论目录的组织结构，设计目标是检索效率



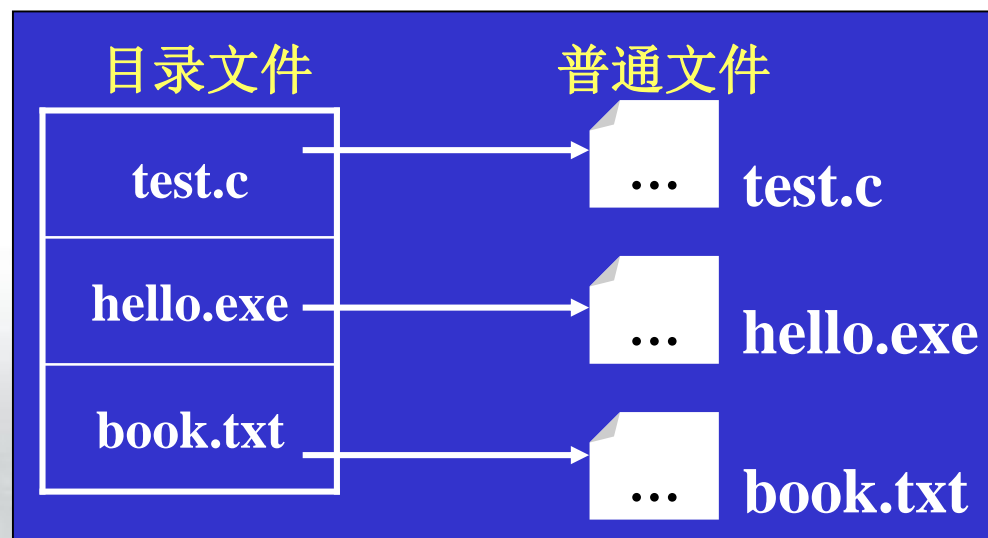
文件目录——目录结构类型

- **一级目录**：整个目录组织是一个线性结构，系统中的所有文件都建立在一张目录表中。
- 它主要用于单用户操作系统。它具有如下的特点：
 - 结构简单；
 - 文件多时，目录检索时间长；
 - 有命名冲突：如重名（多个文件有相同的文件名）或别名（一个文件有多个不同的文件名）
 - 不具备分组功能

文件名	文件说明	物理地址
文件名1	FCB1	
文件名2	FCB2	
.....	

Diagram illustrating the mapping of files to physical addresses:

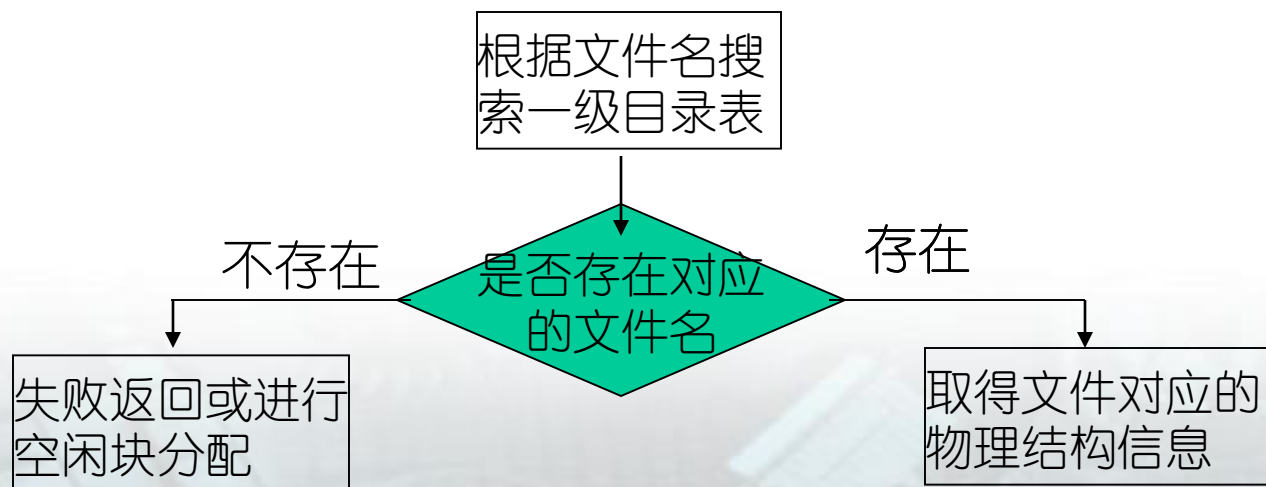
- 文件名1 (File Name 1) maps to 文件1 (File 1)
- 文件名2 (File Name 2) maps to 文件1 (File 1)
- (Ellipsis) maps to

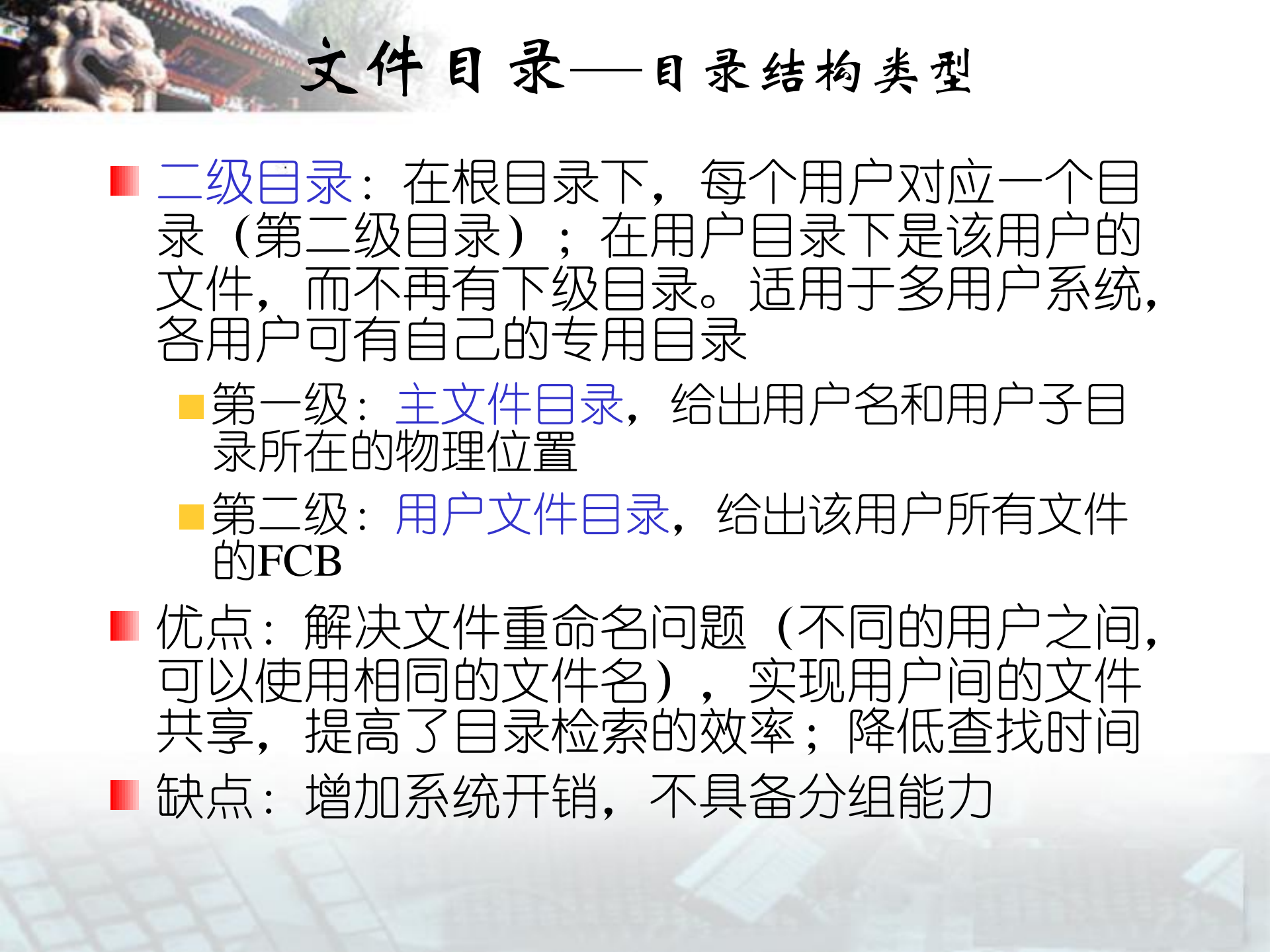


文件目录——目录结构类型

■ 一级目录

- 存放在存储设备的某个固定区，系统启动或需要时，系统将该目录表调入内存或部分调入内存
- 文件系统通过该目录表提供的信息，对文件进行创建、搜索、读写和删除等操作
- 文件系统可实现对文件空间的管理和按名存取





文件目录——目录结构类型

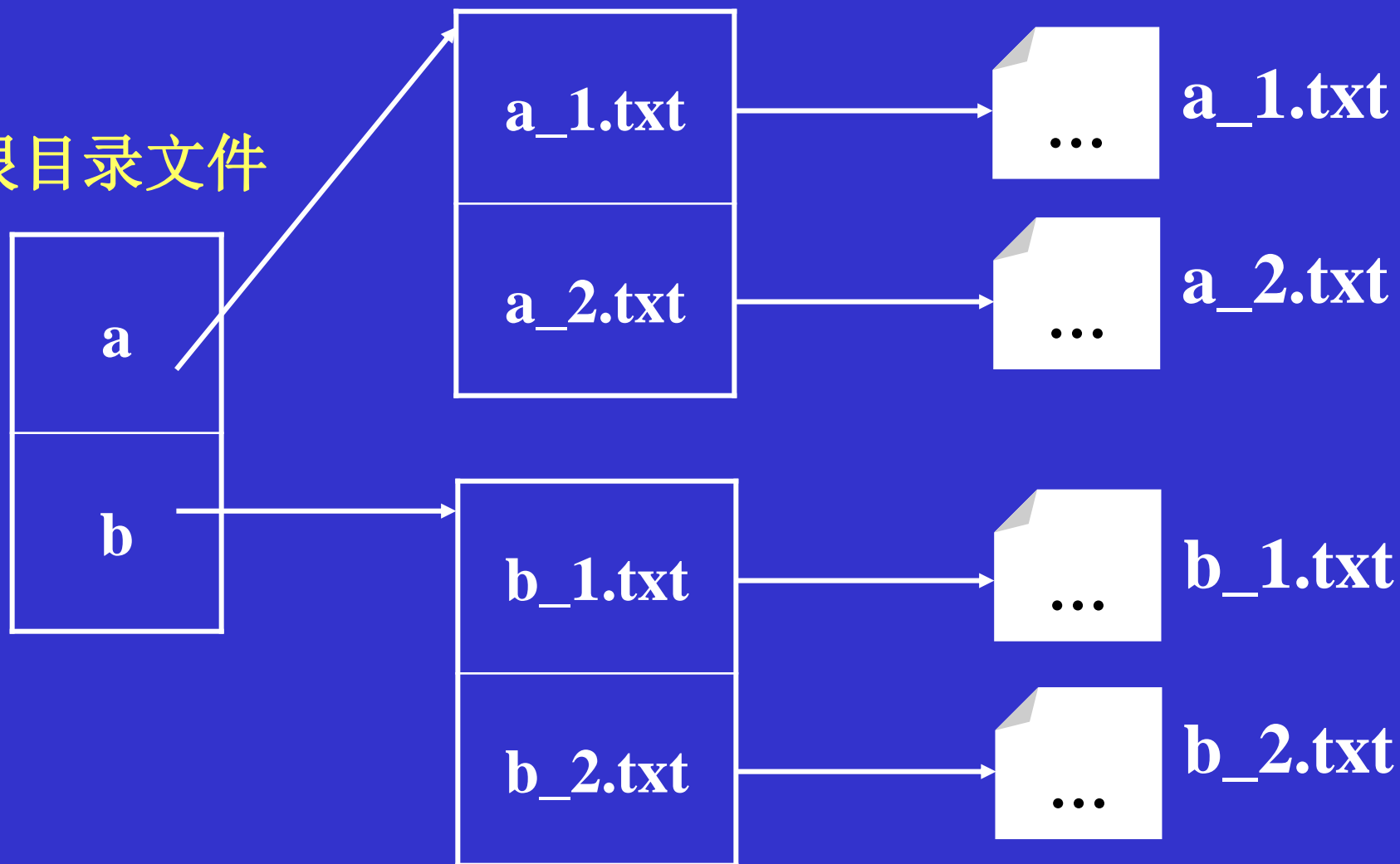
- **二级目录**：在根目录下，每个用户对应一个目录（第二级目录）；在用户目录下是该用户的文件，而不再有下级目录。适用于多用户系统，各用户可有自己的专用目录
 - 第一级：**主文件目录**，给出用户名和用户子目录所在的物理位置
 - 第二级：**用户文件目录**，给出该用户所有文件的FCB
- **优点**：解决文件重命名问题（不同的用户之间，可以使用相同的文件名），实现用户间的文件共享，提高了目录检索的效率；降低查找时间
- **缺点**：增加系统开销，不具备分组能力

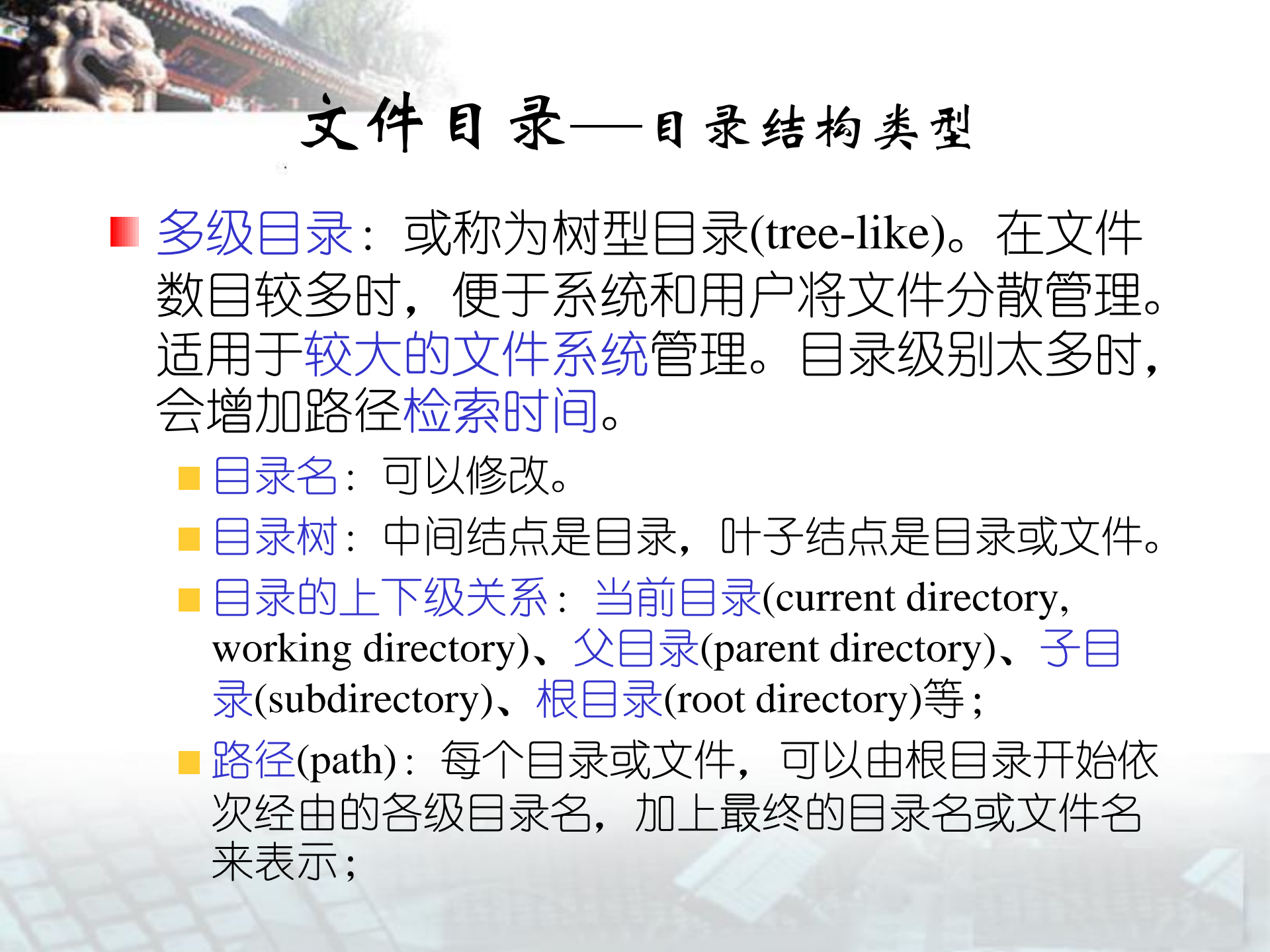
二级目录结构

用户目录文件

普通文件

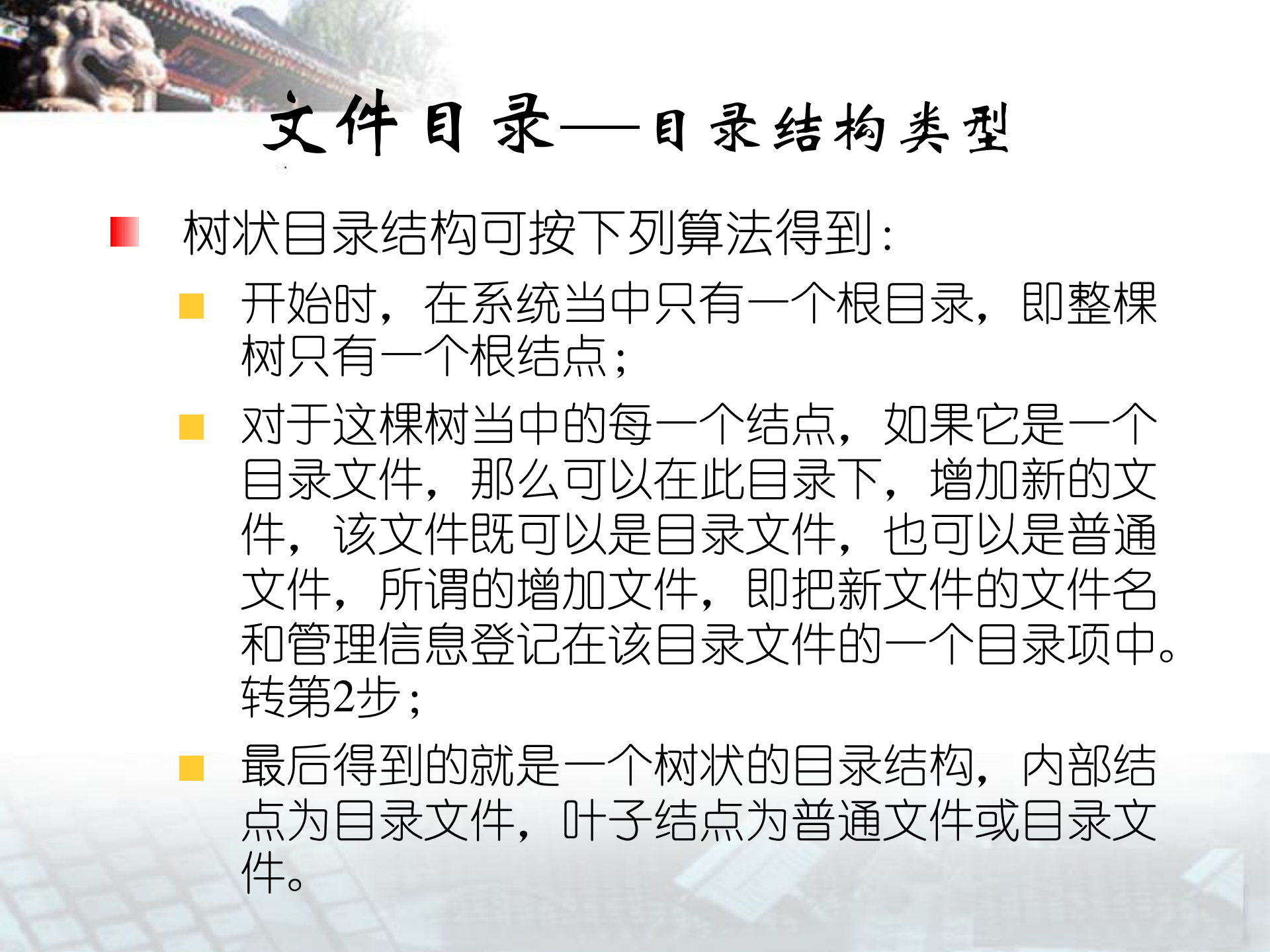
根目录文件





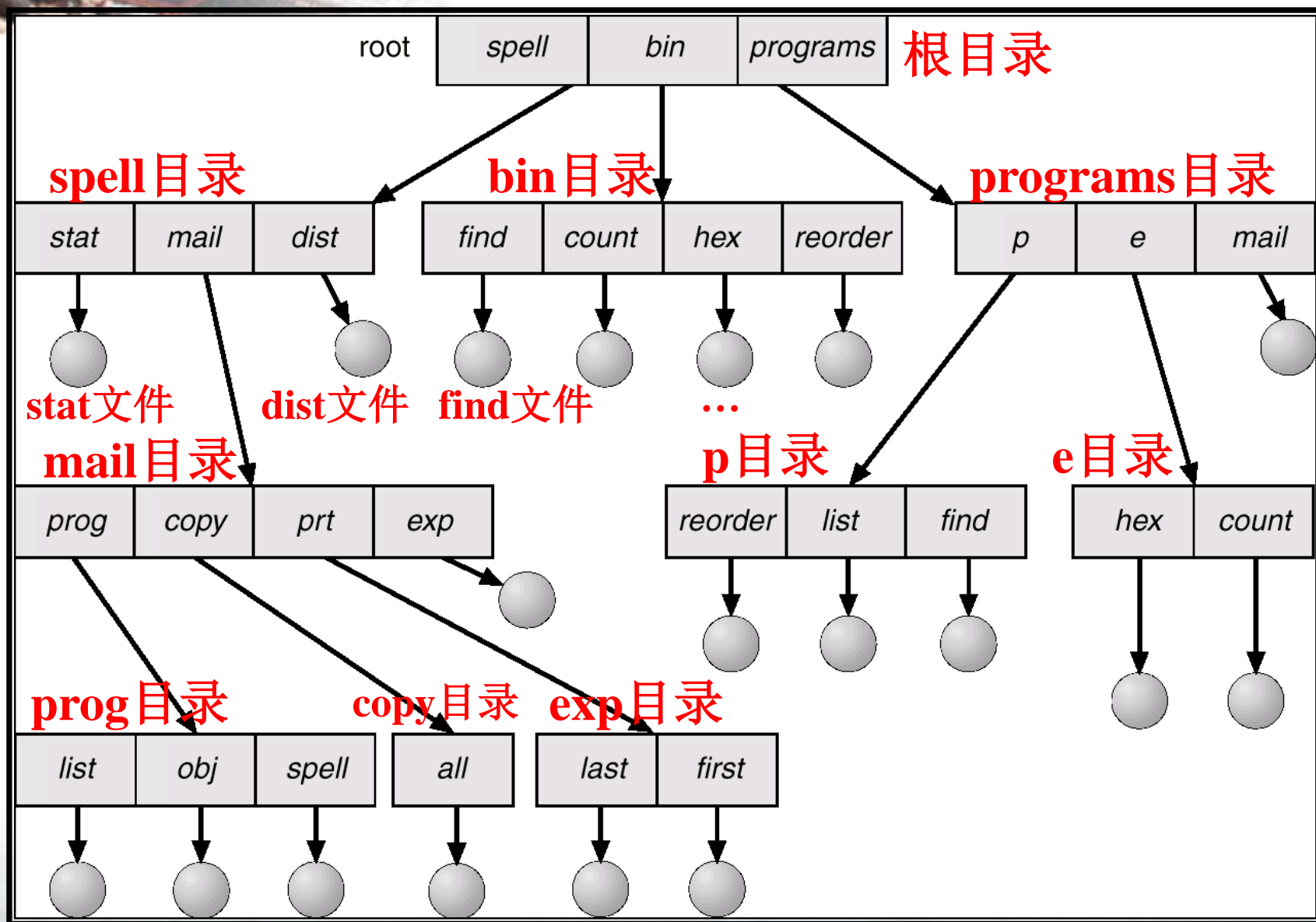
文件目录——目录结构类型

- **多级目录**：或称为树型目录(tree-like)。在文件数目较多时，便于系统和用户将文件分散管理。适用于**较大的文件系统**管理。目录级别太多时，会增加路径**检索时间**。
 - **目录名**：可以修改。
 - **目录树**：中间结点是目录，叶子结点是目录或文件。
 - **目录的上下级关系**：**当前目录**(current directory, working directory)、**父目录**(parent directory)、**子目录**(subdirectory)、**根目录**(root directory)等；
 - **路径**(path)：每个目录或文件，可以由根目录开始依次经由的各级目录名，加上最终的目录名或文件名来表示；



文件目录——目录结构类型

- 树状目录结构可按下列算法得到：
 - 开始时，在系统当中只有一个根目录，即整棵树只有一个根结点；
 - 对于这棵树当中的每一个结点，如果它是一个目录文件，那么可以在此目录下，增加新的文件，该文件既可以是目录文件，也可以是普通文件，所谓的增加文件，即把新文件的文件名和管理信息登记在该目录文件的一个目录项中。转第2步；
 - 最后得到的就是一个树状的目录结构，内部结点为目录文件，叶子结点为普通文件或目录文件。



多级目录组织



文件目录——目录结构类型

■ 在多级目录结构中，如何来指定需要访问的文件？

■ **绝对路径名**：对于每一个文件或目录，可以用从根目录开始依次经由的各级目录名，再加上最终的文件名或目录名来表示（之间用分隔符隔开）一个文件或目录的绝对路径名是唯一的。

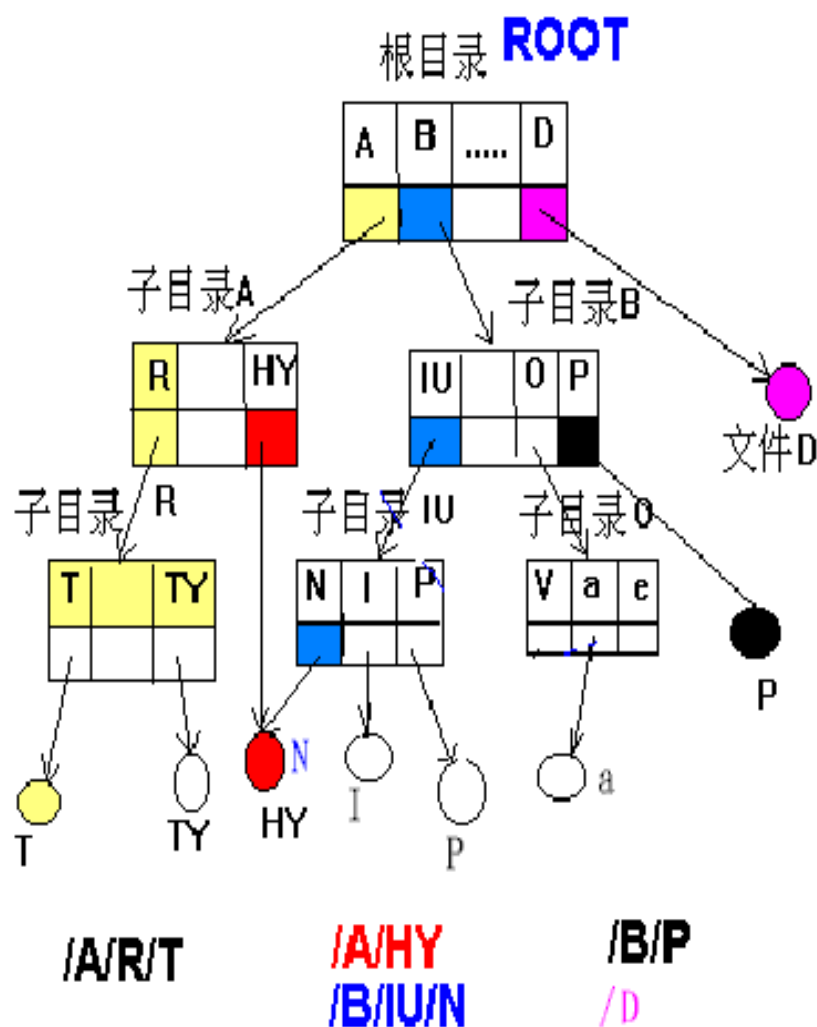
■ **当前目录**：也叫工作目录，

- ◆ 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
- ◆ 用户可以指定一个目录作为当前的工作目录；
- ◆ 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

■ **相对路径名**：在指定一个文件或目录时，可以采用相对于当前工作目录的部分路径名。例如，假设当前目录为：
`\spell\mail\copy`，则绝对路径名`\spell\mail\copy\all`与相对路径名`all`是等效的。



文件目录——目录结构类型



路径名:

一个文件的路径名是由根目录到该文件的通路上所有目录文件名和该文件的符号名组成的。

DOS WINDOWS系统中文件路径名

\A\R\T \B\IU\I

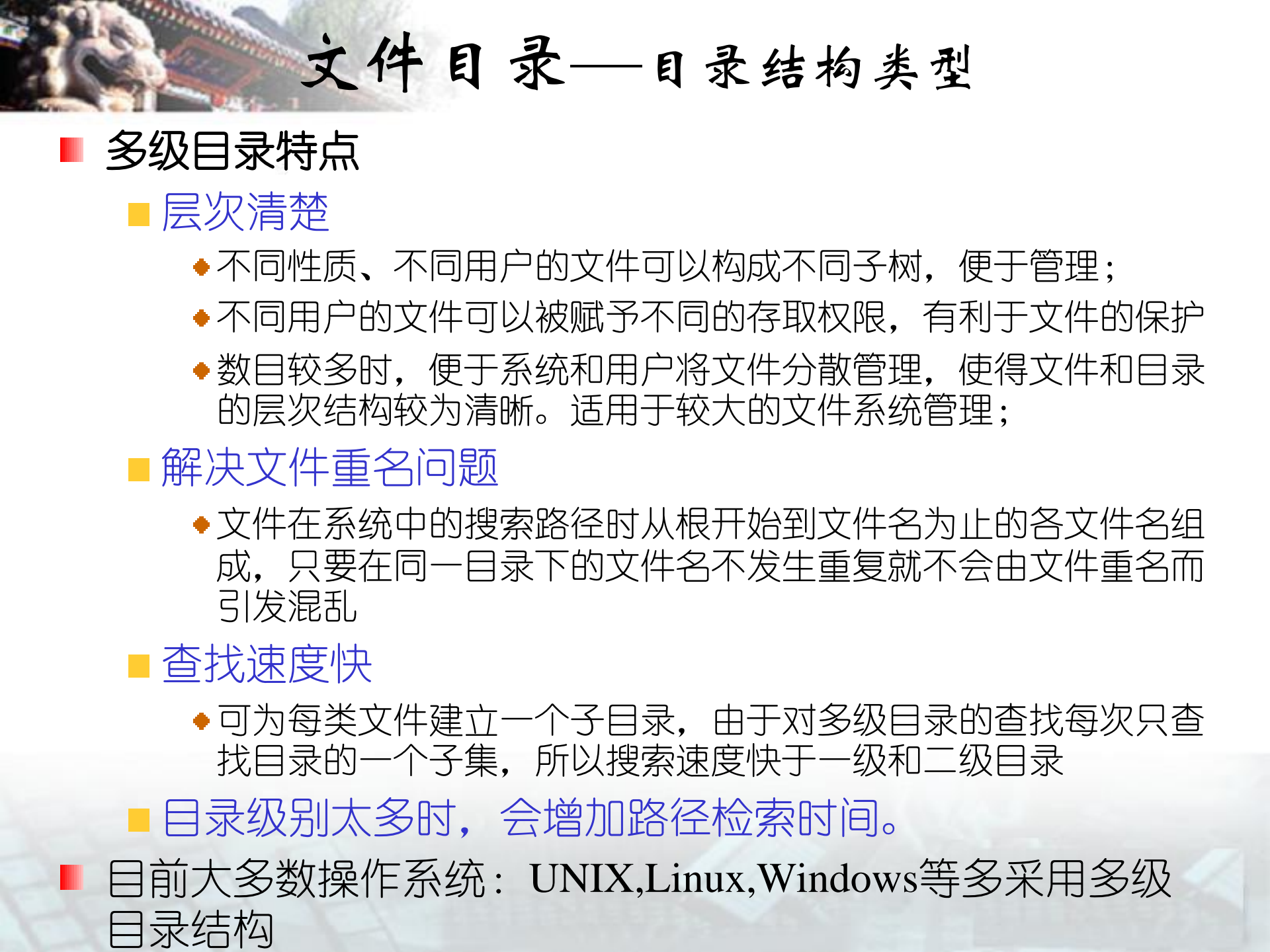
\D \B\P

NIX系统中文件路径名

/A/R/T /B/IU/I

/D /B/P

- NIX系统表示linux, unix, sun, BSD等系统
- win32系统表示win9x/me/NT/2k/XP



文件目录——目录结构类型

■ 多级目录特点

■ 层次清楚

- ◆ 不同性质、不同用户的文件可以构成不同子树，便于管理；
- ◆ 不同用户的文件可以被赋予不同的存取权限，有利于文件的保护
- ◆ 数目较多时，便于系统和用户将文件分散管理，使得文件和目录的层次结构较为清晰。适用于较大的文件系统管理；

■ 解决文件重名问题

- ◆ 文件在系统中的搜索路径时从根开始到文件名为止的各文件名组成，只要在同一目录下的文件名不发生重复就不会由文件重名而引发混乱

■ 查找速度快

- ◆ 可为每类文件建立一个子目录，由于对多级目录的查找每次只查找目录的一个子集，所以搜索速度快于一级和二级目录

■ 目录级别太多时，会增加路径检索时间。

■ 目前大多数操作系统：UNIX, Linux, Windows等多采用多级目录结构



文件目录——目录的实现

目录的主要功能：根据用户给出的ASCII形式的文件名（路径名），迅速地定位到相应的文件控制块。目录的实现需解决以下三个问题：

- 目录项的内容；
- 长文件名问题；
- 目录的搜索方法。







1. 目录项的内容

- 不同的系统采用不同的实现方法，一般分为两类：
 - ☞ 直接法：目录项=文件名+FCB（属性信息、在外存上的存放位置）。如MS-DOS/Windows；
 - ☞ 间接法：目录项=文件名+FCB的地址（索引号）。如Unix（inode）；
- 不管是何种方法，给定一个文件名，即可返回相应的FCB。

直接法

文件名	FCB
Games	FCB1
Mail	FCB2
News	FCB3
Work	FCB4

间接法

文件名	FCB索引
Games	 FCB1
Mail	 FCB2
News	 FCB3
Work	 FCB4

2. 长文件名问题

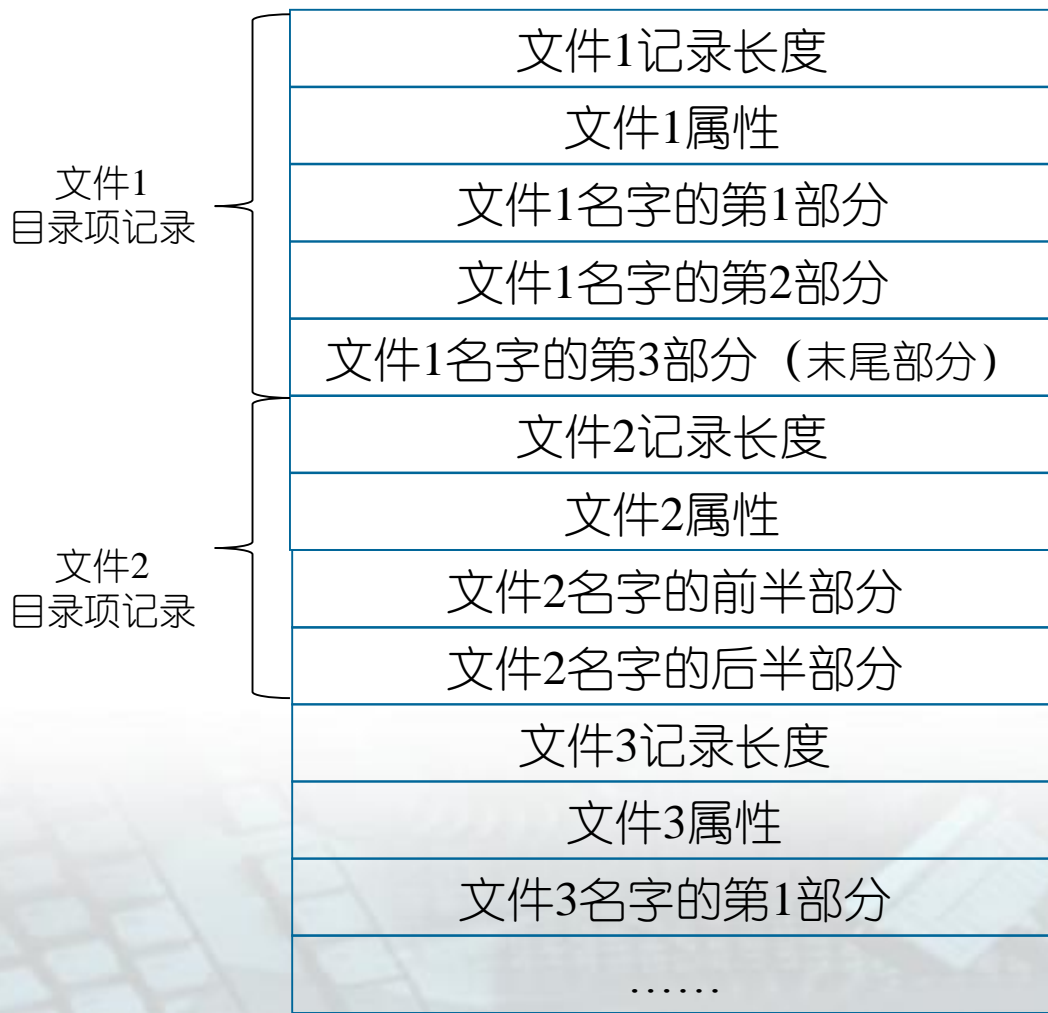
在MS-DOS中，文件名的最大长度为8个字符，扩展名的最大长度为3个字符；在Unix V7中，文件名的最大长度为14个字符。但是在现代OS当中，一般都支持更长的、可变长度的文件名。实现方法有：

- 最简单的方法，不支持长文件名，会激怒用户
- 方法1：在目录项中，将文件名的长度固定为255个字符。缺点：浪费空间，很少文件用很长的名字；



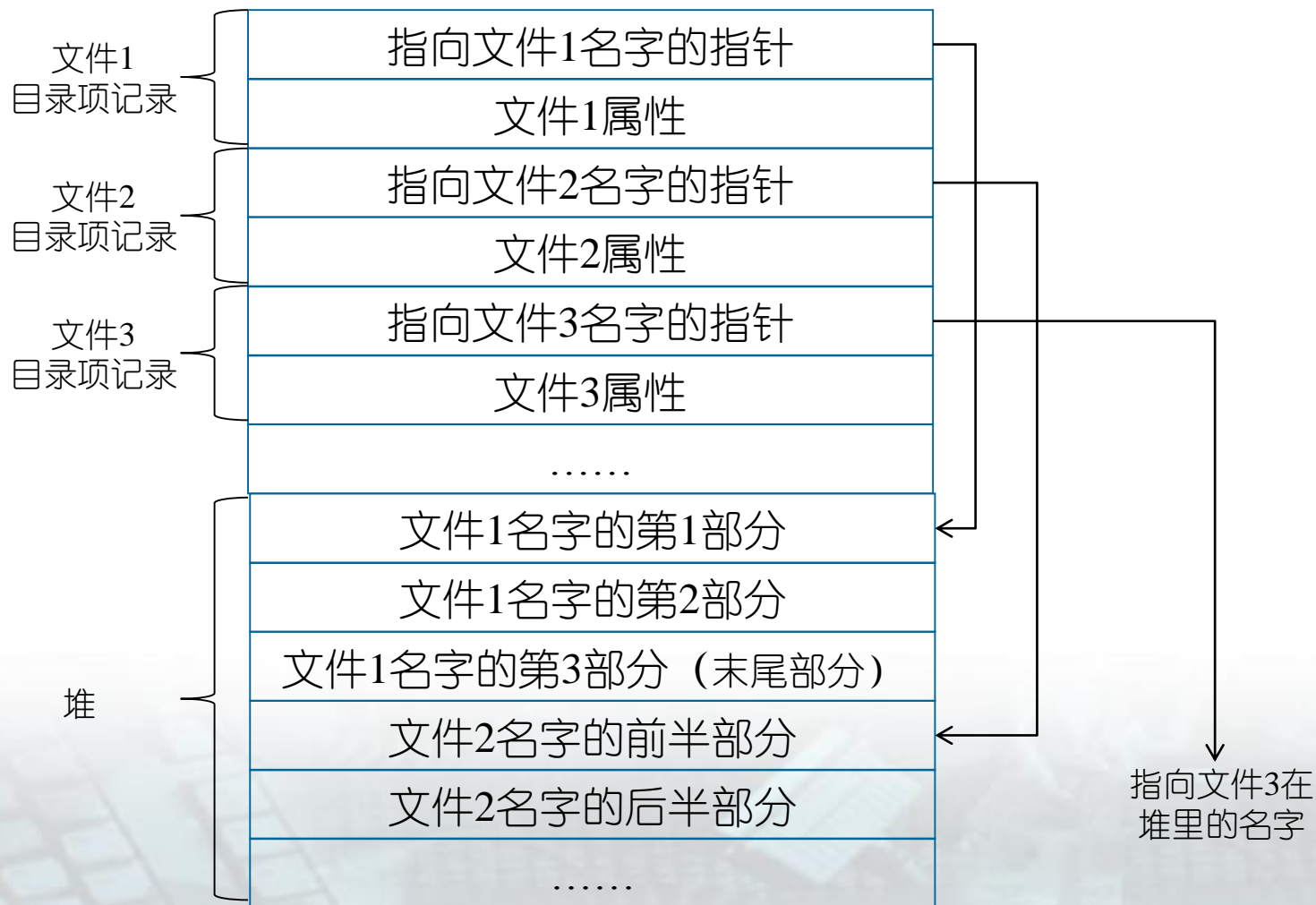
2. 长文件名问题

- 方法2：每个目录项的长度可变，分为三部分：目录项长度、文件的属性信息（此两项长度固定）、文件名（长度可变）。缺点：文件被删除后，该目录项所占用的空间不太好回收利用；



2. 长文件名问题

- 方法3：目录项本身的长度固定，把长度可变的文件名统一放在目录文件的末尾。





文件系统的实现

- 在磁盘上如何安排文件和目录存储、如何管理磁盘空间、怎样使文件系统有效而可靠地工作等
- 存储空间的分配与回收
 - 位示图、空闲块表、空闲链表、成组链表



文件系统的实现——文件存储空间分配 (file allocation)

- 新创建文件的存储空间（文件长度）分配方法
 - 预分配(preallocation): 创建时(这时已知文件长度)一次分配指定的存储空间，如文件复制时的目标文件
 - 动态分配(dynamic allocation): 需要存储空间时才分配（创建时无法确定文件长度），如写入数据到文件



文件系统的实现——外存空闲空间管理

- 外存空闲空间管理的数据结构通常称为**磁盘分配表** (disk allocation table)，分配的基本单位是物理块，空闲空间的管理数据结构：
 - **位图** (bitmap)：每一位表示一个物理块，取值0和1分别表示空闲和占用。
 - **空闲空间链接** (chained free space)：每个空闲物理块中有指向下一个空闲物理块的指针，所有空闲物理块构成一个链表。不需要磁盘分配表，节省空间。每次申请空闲物理块只需取出链表开头的空闲物理块即可。
 - **空闲空间索引** (indexed free space)：在一个空闲物理块中记录其他几个空闲簇的位置。
- 注：可以上述结构结合，应用于不同的场合。如：**位图**应用于索引结点表格，**链接和索引**结合应用于文件区的空闲空间。



文件系统的实现——文件的操作

在文件系统的内部，是如何来实现
open、close、read、write等各种
系统调用函数的？



1. 数据结构

- 位于**外存**上的数据结构：
 - ➡ **目录结构**：用来组织文件，通过文件名来寻找其FCB；
 - ➡ **文件控制块FCB**：记录了文件的各种属性信息和文件所在的物理块信息；
- 位于**内存**中的数据结构：
 - ➡ **系统内打开文件表**：记录了系统中已打开文件的FCB和共享计数等信息；
 - ➡ **进程内（用户）打开文件表**：在进程内部打开的文件所组成的一个表格，包括打开方式、读写指针、该文件在系统打开文件表中的索引。该表保存在PCB中，或在PCB中记录了它的起始地址；



系统内打开文件表与进程内打开文件表之间的关系：

- 两表分离的目的是为了实现文件的共享，即一个文件可以同时被多个进程访问；
- 系统内打开文件表记录的是被打开文件的一些共性信息，而进程内打开文件表记录的是各个进程对该文件访问的个性信息；
- 进程文件表指向系统文件表，如果多个进程同时打开了某一个文件，则在各自的进程文件表项中，指向相同的系统文件表项。

打开方式	读写指针	系统表指针
...
...
...

进程P1的打开文件表

打开方式	读写指针	系统表指针
...
...
...

进程P2的打开文件表

系统内打开文件表

共享计数	FCB
...	...
2	...
...	...



2. 打开文件

任何一个文件在使用前都必须先打开，打开文件的系统调用为：**fd = open（文件路径名，打开方式）**

1. 根据文件路径名一层层地去查找各级目录结构，找到该文件所在的目录项；
2. 根据打开方式、共享说明等信息检查访问合法性；
3. 查找系统内打开文件表，看文件是否已经被其他进程打开，若是，将文件的共享计数值加1，转S4；
若否，**将该文件的FCB从外存读入内存**，保存在系统打开文件表的某个空表项中，共享计数置为1；
4. 在进程打开文件表中增加一项，填写访问方式、当前读写指针等，并指向系统打开文件表对应表项；
5. 返回一个指针给fd，指向进程打开文件表中的相应表项，以后的文件操作均通过该指针来完成。

目录的逐级搜索示例

文件路径名

/Ann/mail/B

根目录

Name	D	SFID
Ann	Y	1034
Bob	Y	179
Yao	Y	7182

Ann目录

Name	D	SFID
mail	Y	2165
A	N	5797

mail目录

Name	D	SFID
sent	Y	434
B	N	2459
C	N	25

目标目录项

用户空间 `fd=open` (文件路径名, 打开方式)

内核空间

查找目录项并
验证合法性

若需要, 把
FCB读入内存

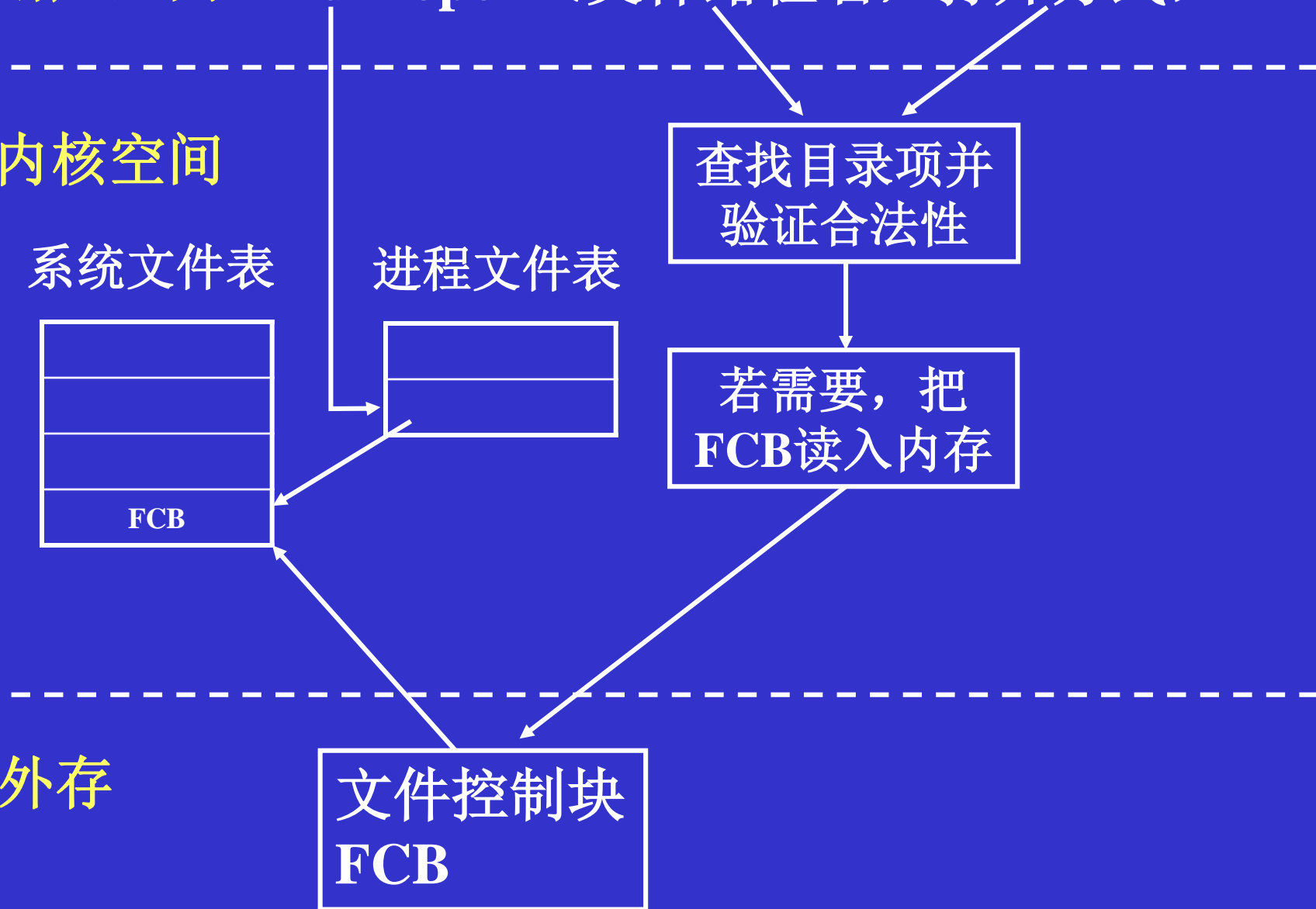
系统文件表

进程文件表

FCB

文件控制块
FCB

外存





3. 关闭文件

文件在使用后必须关闭，相应的系统调用：**close (fd)**

1. 根据**fd**，将进程打开文件表中的相应表项删除；
2. 将系统打开文件表中相应表项的共享计数值减1，如果该值仍大于0，说明还有其他进程在使用该文件，此次操作结束，返回；否则转S3；
3. 把更新后的文件信息复制回外存中的目录结构，然后把系统打开文件表中的相应表项删除。

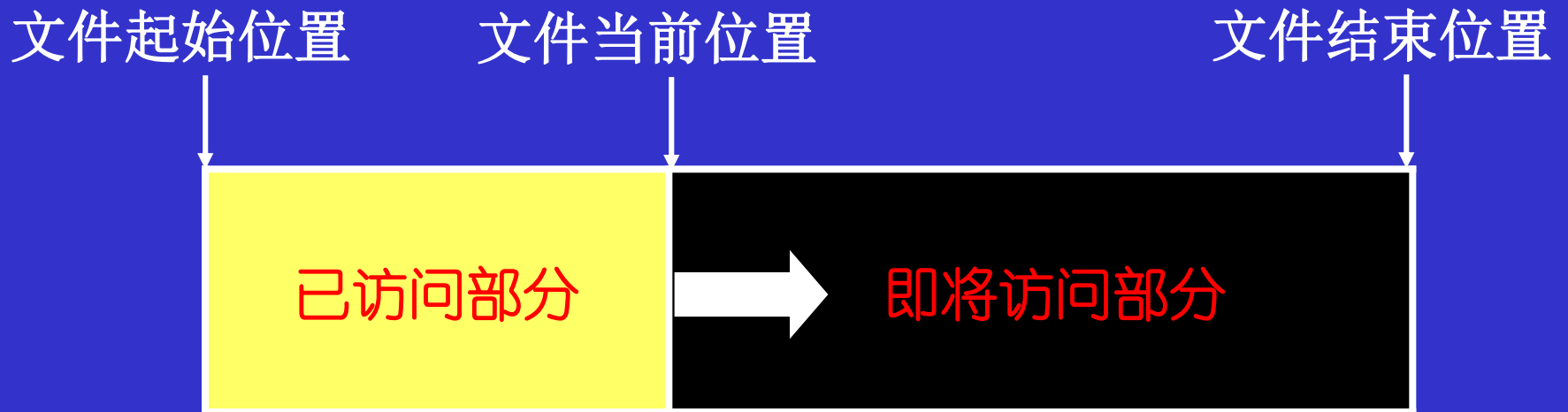


4. 读取文件

系统调用函数 **int read (fd, userBuf, size)**：从文件fd的当前位置开始，顺序读取大小为size的数据块，并保存在userBuf中。

- 通过fd可以访问进程打开文件表中的相应表项，如文件的当前位置，并由此可访问系统打开文件表中的相应表项，即该文件的FCB，由此可知文件的各个逻辑块在外存上的存储位置；
- 需要验证本次操作的合法性，包括读取权限、数据块大小是否越界，等等；
- 以块为单位来访问外存，需将用户给出的文件地址转换为逻辑块，再转换为相应的物理块，然后根据物理块编号去访问外存。

文件的读操作（用户）



文件的逻辑地址空间



其他的一些系统调用：

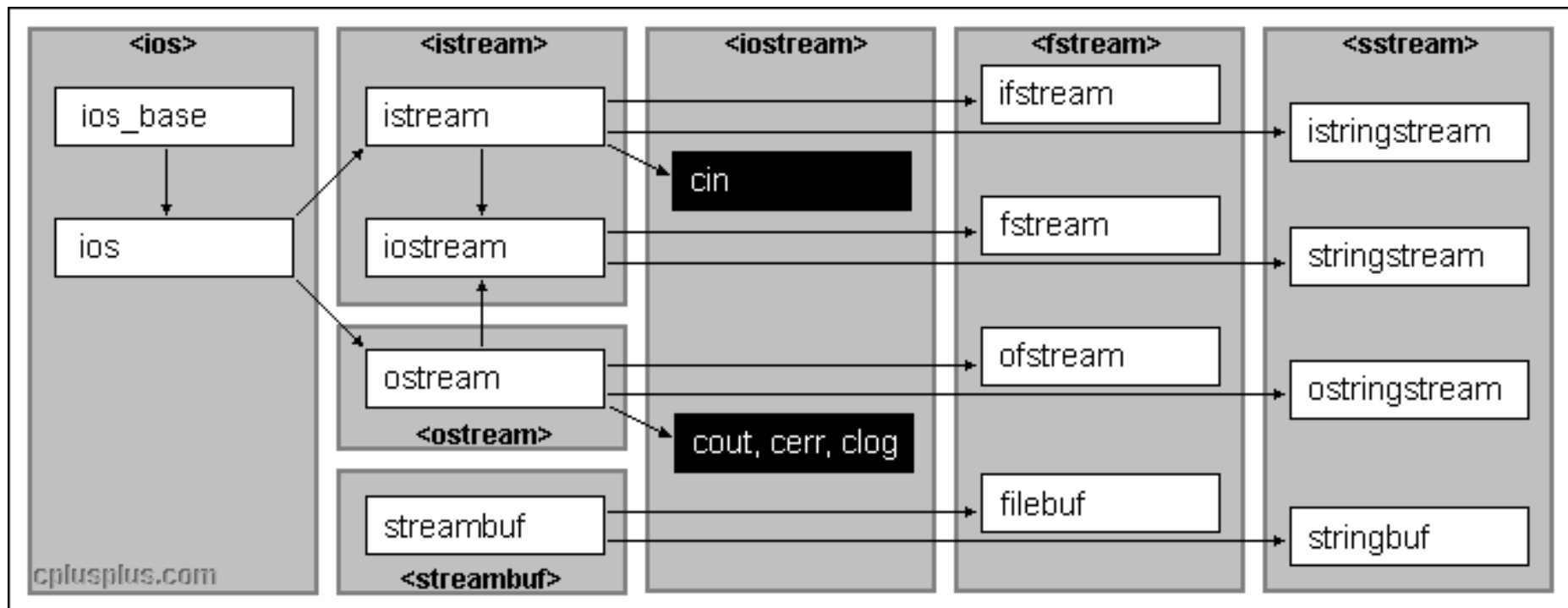
- 创建文件**create**;
- 写文件**write**;
- 添加**append**;
- 文件定位**seek**;
-



带缓冲区的文件


- 操作系统提供的有关文件的系统操作是没有缓冲区的
- 在此基础上，大多数编程环境都提供带缓冲区的文件
 - 设立缓冲区作为中介，化零为整
 - fopen、fclose、fread、fwrite、fgetc、fputc、fgets、fputs、freopen、fseek、ftell

C++ 中的 iostream





主要内容

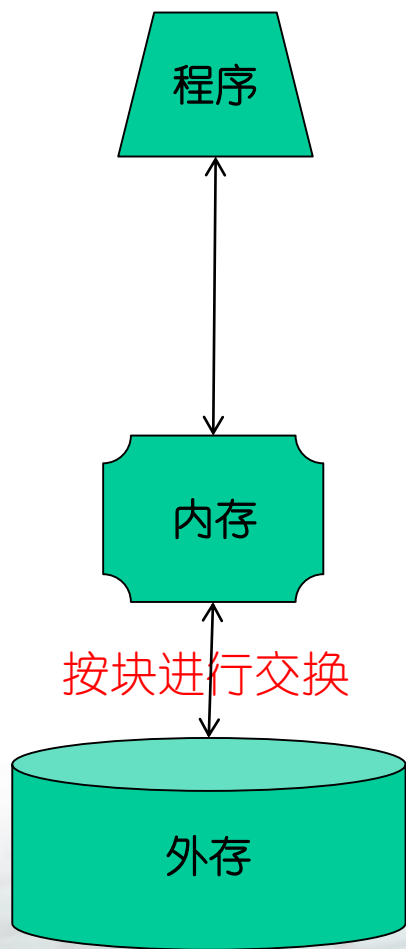
- 存储器介绍
 - 文件系统与文件操作
 - **I/O算法简介**
- 



为什么需要研究I/O算法

- 虽然计算机的内存一直在增长，但需要处理的数据增长更快
- 很多算法主要考虑所有数据均可放入内存的情况，难以适应数据不能完全放入内存的情况
- 虚拟内存是一种统一的解决方案，但没有针对具体问题优化
- 磁盘操作通常比内存操作慢 10^3 至 10^6 倍

外存模型



N = 问题实例中需要处理的数据量

B = 每个磁盘块能容纳的数据量

M = 内存能容纳的数据量

O = 输出的数据量

I/O : 内外存交换的磁盘块数

通常以 I/O 的多少作为衡量标准



例子1：遍历链表

■ 假设

■ $N=12$, $B=3$, $M=6$ (一共12个数据, 每个块能放3个, 内存能放6个)

Case 1:

1	5	9	2	10	6	3	7	12	4	11	8
---	---	---	---	----	---	---	---	----	---	----	---

Case 2:

1	3	2	5	6	4	8	7	9	10	12	11
---	---	---	---	---	---	---	---	---	----	----	----

■ Case 1需要读外存12次 (N 次)

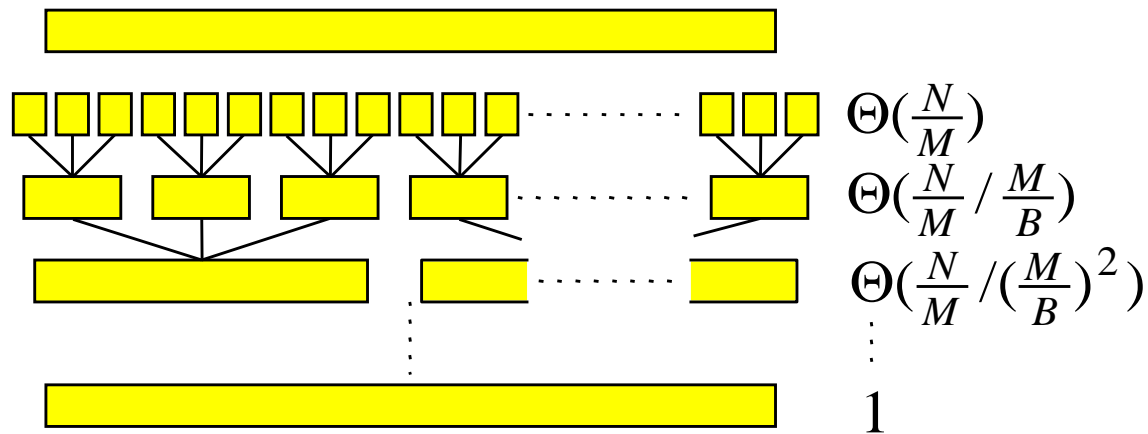
■ Case 2需要读外存4次 (N/B 次)

■ 两者相差 B 倍 (实际中 B 可能是几千或几万)

例子2：排序

■ 归并排序

- 建立 N/M 个内存大小 排好序的列表
- 不断把排好序的列表归并



- 共 $O(\log_{M/B} \frac{N}{M})$ 趟，每趟 $O(N/B)$ 次 I/O
- 所以一共 $O(\frac{N}{B} \log_{M/B} \frac{N}{B})$ I/O



一些基础的下界

	内存算法	外存算法
线性扫描	N	$\frac{N}{B}$
排序	$N \log_2 N$	$\frac{N}{B} \log_{M/B} \frac{N}{B}$
搜索	$\log_2 N$	$\log_B N$
置换	N	$\min \{ N, \frac{N}{B} \log_{M/B} \frac{N}{B} \}$