- 文件系统的功能
- 文件 (File)
  - 文件的命名和属性、文件的顺序和随机访问模式、与文件相关的操作
  - inode (Index Node)
  - 打开文件表 (Open-File Table)
- 目录 (Directory)
  - 目录的层级结构
  - 硬链接 (Hard Links) 和符号链接 (Symbolic Links)
- 文件和目录的角色：Two Key Abstractions

一、文件

1、文件系统功能（简答）

- 数据：存储，提供访问接口，分类和索引
- 磁盘管理
- 访问权限，保护
- 安全可靠，不能读数据

2、文件命名

主要方便用户，对用户屏蔽磁盘细节

一般合法文件名：1-8个字母组成的字符串，允许数字

> UNIX区分大小写

.后面的是扩展名，作用是给OS提示如何处理的

3、文件属性7（文件元数据）

保存数据外相关信息，用文件控制块FCB（文件目录项，必须连续存放）来维护

- 文件保护相关：访问权限、合法操作、口令、创建者（ID）、所有者（ID）
- 标记：隐藏、存档、系统、、临时、随机访问、加锁
- 名称
- 类型

- 位置：指向设备和设备上文件的指针
- 大小：
- 时间：创建时间、最后一次修改时间、最后一次存取时间

4、文件顺序&随机访问模式

磁盘存储可以不按顺序读取、按关键字
随机很慢，指针索引比较快

5、文件基本操作

复制=创建+读+写

- create：为新文件分配空间+目录中创建目录项
- delete：目录检索，释放空间，删除目录条目
- write：系统调用，查找位置，维护写指针
- read：系统调用，查找位置，维护读指针（同一指针）

- append：write阉割版，只能在末尾写
- rename（非必须）
- seek：对随机访问文件指定从何处开始
- get/set attributes：读取、设置文件属性

6、文件操作-打开文件表

为避免多次重复检索目录，维护一个包含所有打开文件信息的表（节省再次搜索开销）
系统调用close从表中删除这一条目
打开文件的信息：

- 文件指针
- 文件打开计数
- 文件磁盘位置
- 访问权限：创建，只读，读写，添加

二、文件存储实现（物理结构）

文件分配：磁盘非空闲块

inode索引结点

检索只需文件名，不需要其他信息也导入内存，因此把文件名和文件描述信息分开

文件描述信息单独形成一个数据结构：inode（i结点，索引结点）

目录项=文件名+inode

1、连续分配

> 访问磁盘1次

优点：

- 实现简单，存取速度快
- 寻道数和寻道时间最小

缺点：（参考固定长度的数连续组）

- 文件长度不易增加，需要大量移动盘块
- 很难维持有序性
- 反复增删产生外部碎片
- 难确定需要的空间，只适用于长度固定的文件

2、链接分配

> 访问磁盘n次

优点：

- 无外部碎片，提高磁盘利用率
- 动态分配，不需知道文件大小
- 插入，删除，修改方便

隐式链接

缺点：

- 只能顺序访问，随机访问效率低，查找要多次方盘
- 可靠性差：链表指针丢失难恢复

改良，使用簇包含几个盘块，产生一些内部碎片，但减少访问时间

显示链接--FAT文件分配表

表只有一张，把链接各物理块的指针显式存放在内存的链接表中（系统启动时就会读入内存）

优点：提高检索速度，减少访问磁盘次数

缺点：FAT占用较大内存空间，且并没必要把整个FAT调入内存

- 文件的第一个盘号再FCB的物理地址中，其余查FAT
- -1表示最后一块。-2表示空闲

3、索引分配

> m级索引访问磁盘m+1次

直接索引分配

将每个文件所有盘块号集中构成索引块（表）

优点：直接访问，无外部碎片

缺点：索引分配增加系统存储空间开销，索引块太小就无法支持大文件

解决方法：链接索引，多层索引，混合索引

多层索引

计算max：4KB索引块有1024个4B指针，两级索引支持2^20个数据块（4B），即4G大小的最大文件

混合索引

> 假设盘块4KB

- 小文件（40KB）：文件的盘块地址直接放进FCB，直接寻址
- 中文件（4MB）：单级索引：FCB中的索引表中获得盘块地址（一次间址）
- 大文件：两（4GB）或三级（4TB）

- 文件系统布局 (File System Layout)
- 文件存储的实现 (Allocation Method)
  - Contiguous
  - Linked List
  - FAT Table
  - Indexed (inode 的多级索引设计)
- 目录的实现 (Directory Implementation)
  - 目录项中包含的信息、不同长度文件名的处理
  - 在目录中查找文件

三、目录

1、层级结构

单级目录

访问、新建、删除都要遍历文件名

缺点：

- 查找速度慢
- 不允许文件重名
- 不便于文件共享

两级目录

- 主文件目录：不同用户
- 用户文件目录：FCB

优点：

- 提高了访问速度
- 解决了不同用户重名问题
- 在目录上实现访问限制

缺点

- 缺乏灵活性，不能分类

树形目录

绝对目录：从跟出发

Absolute path name: path from the root directory to the file

Windows	\usr\ast\mailbox
UNIX	/usr/ast/mailbox
MULTICS	>usr>ast>mailbox

当前目录（工作目录）：用“."表示

优点

- 分类。层次清晰
- 有效管理、保护
- 存取权限方便

缺点

- 查找中间结点多，增加磁盘访问次数
- 不便于文件共享

无环图目录

增加指向同一结点的有向边

为什么：为每个结点设置共享计数器，用户删除时，另一个用户还可以访问。仅当计数器为0才删除结点，否则只删除共享链

共享文件：一个人修改，所有人都能看见

缺点：实现了文件共享，但管理复杂

2、目录的实现

目录项信息

提供查找文件磁盘块所需信息：文件属性

- 磁盘地址
- 块首地址
- 索引节点

不同长度文件名处理

- 不能限制长度，浪费空间
- 法一：固定格式数据+任意长度文件名
- 缺点：移走后会有长度可变空隙，不一定适合下一任
- 法二：目录项固定长度，文件名放到目录后面的堆中（下一任肯定适合空隙）

目录查找文件

目录查询要在磁盘反复搜索、不断IO，所以需要把当前目录复制到内存

- 线性列表：实现简单，但查找费时，太长不行
- 哈希表：查找非常迅速，插入和删除也简单，但要防止多对一映射，缺点是管理复杂（目录有上千文件才用）
- 查找结果存入高速缓存

3、文件共享

系统中只保留一个副本，节省空间

硬链接：基于索引结点

- 索引节点中：count链接计数（多少个用户）+文件属性信息（不再放在目录项）
- 文件目录中：文件名+指向索引结点的指针
- 只要还有一个，索引节点就不会删掉
- 优点：简单，快
- 缺点：仅用于单个文件系统、不能跨越文件系统；可文件共享但不能目录共享

创建者拥有文件，但不能删除文件（否则留下其他用户的悬空指针），删除后只是count-1

软链接：利用符号链

只有文件主才有指向其索引节点的指针，其他人只能通过符号链看到路径名。文件主删除后，其他人访问失败，符号链删除

优点：链接不同文件系统文件，链接目录；链接不同机器文件

缺点：

- 慢
- 根据路径名逐个查找目录，多次读磁盘，开销大

4、文件和目录两个重要抽象Two key Abstractions？？？

- **File:** Array of bytes that can be read and written
  - Associate bytes with a user readable name
  - Also has a low-level name (inode number)
  - Usually OS does not know the exact type of the file
- **Directory:** A list of files and directories
  - Each entry either points to file or other directory
  - Contains a list of user-readable name to low-level name, like <foo, inode number 10>

文件：可被读写的数据

- OS不知道文件类型，文件有可读的名字和索引结点

目录：关于目录和文件的列表

- 指向了文件或其他目录
- 包含可读文件名和索引结点

- 磁盘空闲空间管理 (Disk Space Management)
  - 不同数据块大小对文件系统的影响
  - 记录空闲块：Bit Map、Free List
- 文件系统的性能
  - 缓存 (Cache)
  - 快速文件系统 (Fast File System)
- 文件系统的一致性
  - 文件系统一致性检查 (fsck)
  - 日志文件系统 (Journaling)
- 虚拟文件系统 (Virtual File System)

四、文件系统

1、文件系统布局

卷：包含文件系统的分区

- 卷中文件区和目录区（FCB）分离

- 主引导记录MBR：磁盘0号扇区
- Partition Table：每个分区的起始
- 引导块：启动OS的信息
- 超级块：包含文件系统所有关键参数

磁盘中：

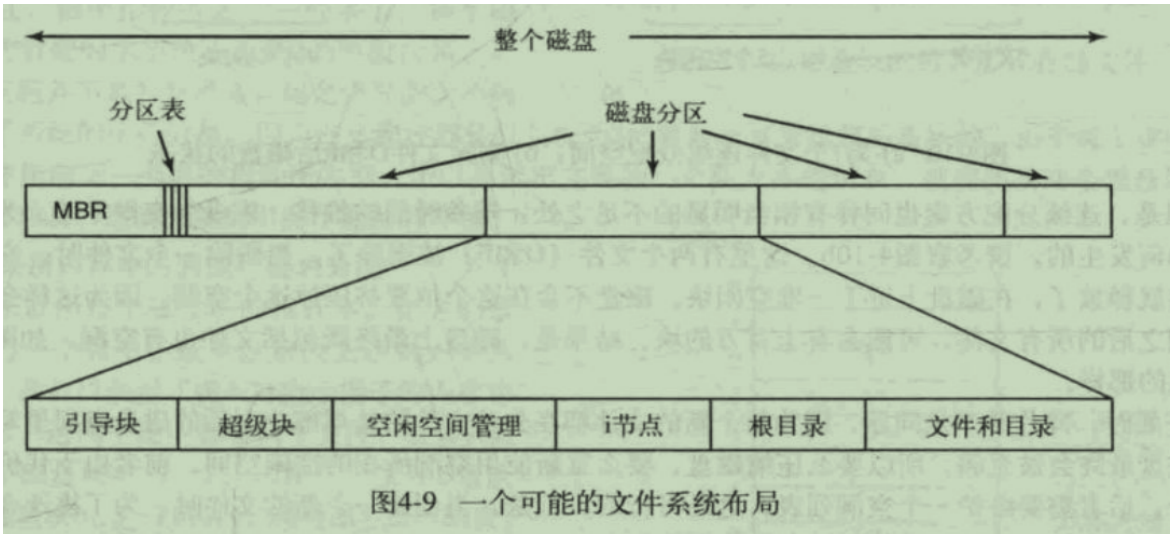

图4-9 一个可能的文件系统布局

内存中：

内存中的信息用于管理文件系统并通过缓存来提高性能。这些数据在安装文件系统时被加载，在文件系统操作期间被更新，在卸载时被丢弃。这些结构的类型可能包括：

1）内存中的安装表（mount table），包含每个已安装文件系统分区的有关信息。

2）内存中的目录结构的缓存包含最近访问目录的信息。对安装分区的目录，它可以包括一个指向分区表的指针。

3）整个系统的打开文件表，包含每个打开文件的 FCB 副本及其他信息。

4）每个进程的打开文件表，包含一个指向整个系统的打开文件表中的适当条目的指针，以

2、磁盘空闲空间管理

不同数据块大小的影响（简答）

How about the **block size?**

- Too large: waste a large amount of disk space for small files
- Too small: waste multiple seeks and rotational delays to read files (each file will consist of many blocks)

太大浪费磁盘空间，太小寻道和旋转等待长，读文件慢

块4kb比较好

记录空闲块2

- 空闲表法和空闲链表法都不适用于大型文件系统，因为表太大

位视图法

二进制表示m*n个盘块状态

矩阵对应盘块号：n（i-1）+j

成组链接法

> 结合空闲表法和空闲链表法，克服表太长的缺点

成组链块：存放一组空闲盘块号的盘块
分配过程从前往后，先分配第一组，然后分配第二组……
回收过程正好相反，从后往前分配，先将释放的空闲块放入第一组，第一组满了，再开辟一组，之前的第一组变为第二组……

3、虚拟文件系统

面向对象的思想，抽象出一个通用的文件系统，定义通用文件系统都支持的接口，屏蔽不同文件系统的差异和操作细节
并非实际的文件系统，只存在于内存中
提高系统性能：最近最常使用的目录项在高速缓存的磁盘缓存中，可以加速路径转换过程

4、文件系统性能

高速缓存

上千块，散列表
页面置换算法
块提前读提高命中率（只适用于顺序读取）

快速文件系统

- 提高block大小
- 柱面组提升数据IO效率
- 策略：如何分配文件和目录

5、文件系统一致性（简答）

产生原因：管理空闲块出错；分配回收程序出错
问题严重后果：丢失盘块；破坏信息

文件系统一致性检查fsck？？？

检查块+文件

日志文件系统

具有故障恢复能力的文件系统，因为对目录以及位图的更新信息，总在原始磁盘日志被更新之前，被写到磁盘上的一个连续的日志上，保证了数据的完整性

T：文件系统题型

- 目录项空间=磁盘块-链指针空间
- 隔层提速：建立链接文件，提拔上来
- 别人的手下你未必有权限访问
- 硬链接的文件和他的父目录都不能删，软连接可以

1、磁盘访问次数（链式）

层数*层磁盘数（层磁盘数最小为1，最大看目录下最多有多少文件，要占多少个磁盘块，这一步是访问到目标的FCB）+索引层数（FCB到地址，最大三级索引）+1（读内容，读连续文件一定是一次，非顺序算文件大小除磁盘块）

2、fsck

3、文件占磁盘空间

还有简介索引表也占磁盘块
超10个物理块的注意

4、max

根目录下能装多少文件：看根目录区能装多少目录项

文件系统能装多少文件：看inode区能装下多少inode

索引：单文件最大长度：索引表区含的索引项数*磁盘块大小

链接：单文件最大长度：
存储块1kb，连接指针4b，则索引范围2^32，每块大小1024-4=1020b

FAT最大长度&单文件最大长度
FAT表项2B，簇4kb则可索引2^16个表项，最大2^16*2B
文件2^16*4kb