



文件系统基础

中国科学院大学计算机学院

2025-12-24





内容提要

- FS基础
 - 用户视图
 - 基本概念
- FS内部结构



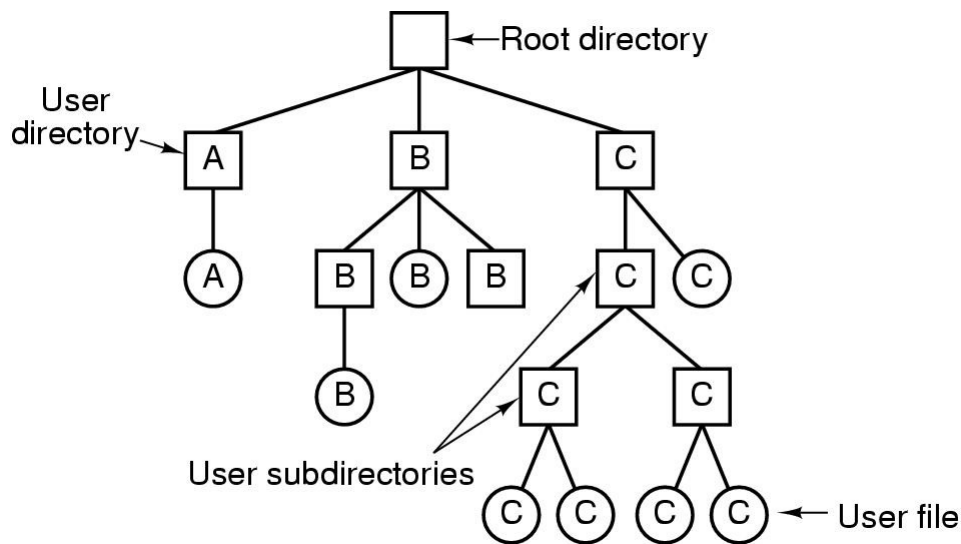
为什么需要文件系统 (File System, FS)

- 持久化保存数据需求 (persistence)
 - 进程结束、关机/关电、宕机/掉电
 - 使用持久化存储设备：磁盘、SSD等
- FS是对持久化数据存储的抽象
 - 给用户/程序开发者提供一个逻辑上的持久化存储
 - 文件、目录形式
 - 简单、易理解、操作方便
 - 将复杂的、公共的管理功能从用户程序中移出
 - 存储设备管理，例如磁盘
 - 数据管理，即程序持久化数据的组织和增删改查
- 对FS的基本需求
 - 能够保存大量 (复杂多样) 的信息 → 管理问题
 - 多个进程同时访问 → 并发控制问题
 - 多用户共享数据 & 私有数据 → 安全保护问题



文件系统的用户视图

- 文件：数据组织的单位
 - 文件是命名的字节数组
 - 用户将数据组织成文件，根据文件名来访问对应的数据
 - FS不感知文件的内容：使用文件的进程负责解析内容
- 目录：文件组织的单位
 - 一组文件和目录的命名集合
 - 父目录、子目录
 - 同一目录下无重名
- 名字空间：树形层次结构
 - 文件系统的逻辑视图





文件

- 文件名：由字母、数字及某些特殊字符组成的字符串
 - 用户根据文件名来访问文件
 - 文件扩展名：描述文件的用途
- 文件属性
 - 文件大小、所有者、时间戳、访问权限
 - 文件逻辑地址：0 .. fsize-1，指示数据在文件中的位置
- 文件内容：无结构
 - OS将文件视为无结构的字节数组
 - 程序开发者可以定义任意结构的文件
- 文件的类型
 - 常规文件、目录文件、设备文件、可执行文件



文件

- 文件的访问
 - 打开文件 & 文件描述符
 - 当前位置：文件内部的逻辑地址，范围是 $[0, \text{fsize}-1]$
 - 访问方式（Access mode）：读、写、执行



文件访问接口 (syscall)

- 创建文件 : `fd = creat(fname, mode);`
- 删除文件 : `unlink(fname);`
- 打开文件 : `fd = open(fname, flags, mode);`
- 关闭文件 : `close(fd);`
- 读文件 : `rn = read(fd, buf, count);`
- 写文件 : `wn = write(fd, buf, count);`
 - 追加写 : 用 `O_APPEND` 打开文件
- 定位文件 : `lseek(fd, offset, whence);`
- 同步写回文件 : `fsync(fd);`
- 截断文件 : `truncate(fname, length);`
- 获取属性 : `stat(fname, attbuf);` 或 `fstat(fd, attbuf);`
- 重命名文件 `rename(oldpath, newpath);`



文件访问模式 (Access pattern)

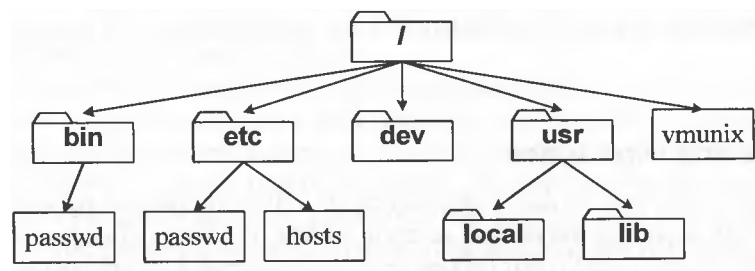
- 顺序访问
 - 从头到尾依次访问每个文件块
 - 例子：观看一部电影，读一篇文章
 - 顺序访问文件 ≠ 磁盘上顺序访问扇区
- 随机访问
 - 每次随机访问一个文件块
 - 例子：只看电影的某些片段
- 按关键字访问
 - 查找包含关键字的文件及段落
 - FS自身没有提供此功能，可以借助应用程序完成，例如grep
 - 与之对比，数据库自身可以实现按关键字查找记录



目录

- 路径

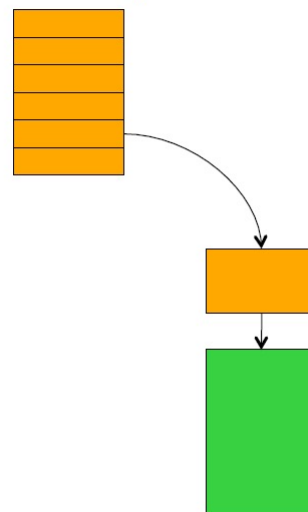
- 根目录 & 当前工作目录
- . : 当前目录
- .. : 父目录
- 绝对路径 vs. 相对路径



- 目录：一种特殊的文件

- 名字 & 属性
- 目录和文件用相同的数据结构：inode
 - 用一个标志 (i_mode) 来区分文件与目录
- 目录内容：描述它所包含的目录和文件集合
 - 有结构：逻辑上是一张表
 - 目录项：每个成员一项
 - 不同FS采用不同的结构
 - 由FS负责维护和解析目录内容
 - 访问目录 vs 访问文件：不同的syscall

Directory





目录访问接口 (syscall)

- 创建目录 `mkdir(dirname, mode);`
- 删除目录 `rmdir(dirname);`
- 打开目录 `fd=open(dirname, flags);`
- 关闭目录 `close(fd);`
- 读目录 `readdir(fd, direntbuf, count);`



-
- ```

graph TD
 Root[Root directory] --> A1[A]
 Root --> B1[B]
 Root --> C1[C]
 A1 --> A2((A))
 B1 --> B2[B]
 B1 --> B3((B))
 B1 --> B4[B]
 B2 --> B5((B))
 B4 --> C2[C]
 B4 --> C3((C))
 C1 --> C4((C))
 C1 --> C5((C))
 C3 --> C6((C))
 C3 --> C7((C))
 C5 --> C8((C))
 C5 --> C9((C))
 C2 --- C4

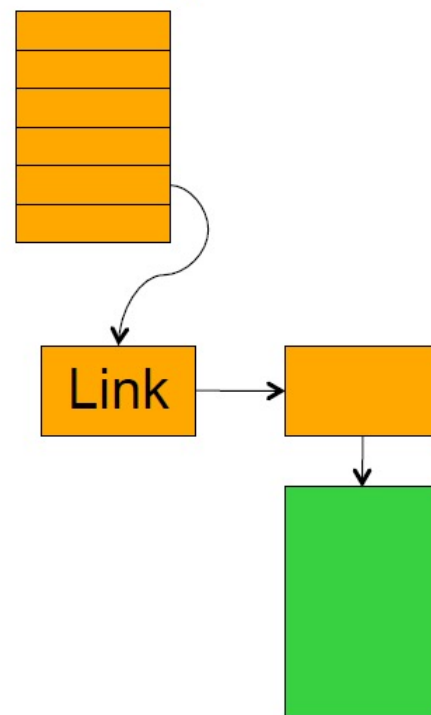
```
- Shared file



# 符号链接

- 另一种文件共享的手段
  - `symlink(srcpath, linkpath)` 系统调用 或 `ln -s` 命令
  - `ln -s source link_name`
  - 创建一个普通文件 `link_name`，该文件内容为 `source` 的路径，具体取决于 `source` 使用的是绝对路径还是相对路径

Directory



符号链接与硬链接有什么区别？



# 内容提要

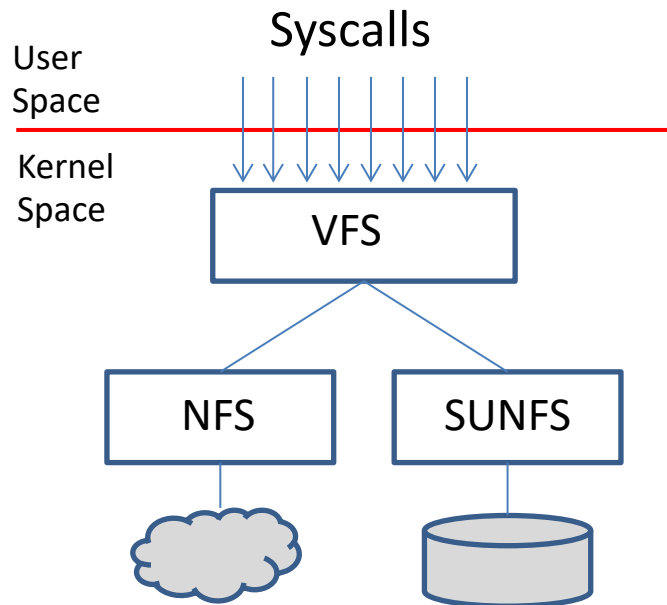
---

- FS基础
- FS内部结构
  - 虚拟文件系统与物理文件系统
  - 主要数据结构
  - 磁盘布局与超级块



# 虚拟文件系统 ( VFS )

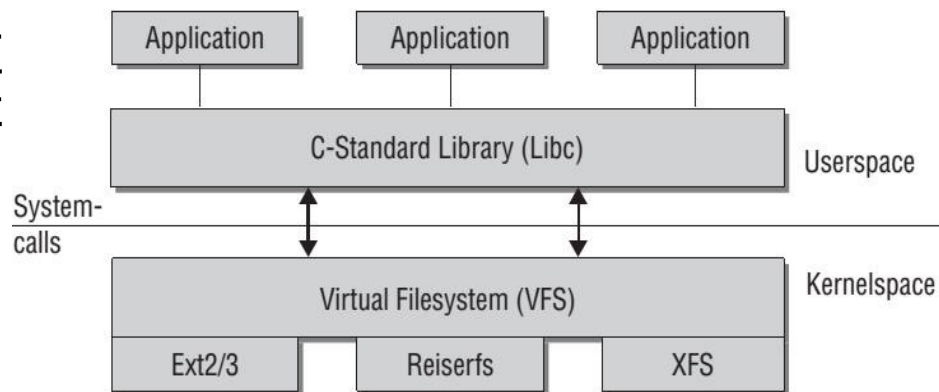
- 同时挂载不同类型的FS
  - SUNFS访问本地的磁盘
  - SUN NFS访问远端服务器的FS





# 虚拟文件系统 ( VFS )

- 虚拟文件系统
  - 实现FS接口和通用功能
- 物理文件系统
  - 磁盘布局、数据结构、磁盘空间管理、名字空间管理等
- 虚拟文件系统开关表
  - 用于物理文件系统的挂载与卸载
  - 每一种类型的文件系统有一个表项
    - 文件系统类型的名字
    - 初始化函数指针，用于mount
    - 清除函数指针，用于umount





# FS相关的接口（系统命令）

- 创建文件系统      `mkfs`：创建磁盘布局：初始化超级块、bitmap、根i-node等
- 删除文件系统      `rmfs`
- 挂载(安装) FS      `mount -t fstype dev dir`
- 卸载文件系统      `umount dev|dir`
- 显示已安装FS      `mount` 或者 `mount dev|dir`
- 获取文件系统属性 `df` (`statfs`系统调用)
- .....
- 创建、删除、安装、卸载只允许特权用户调用





# mount

例子：`mount -t ext4 /dev/sdb /home/os`

- 前提
  - 文件系统类型ext4必须事先已加载到内核
  - 挂载目录/home/os必须已经创建好
- 步骤
  - 根据文件系统类型，查VFS开关表，找到该ext4类型FS的初始化函数，即ext4\_mount()
  - 调用ext4\_mount
    - 读取超级块
    - 读取根目录i-node
  - 初始化一些内存数据结构：超级块、根i-node等



# FS主要数据结构

---

- i-node : 描述文件/目录 , 也称为文件元数据
- dentry : 目录项 , 记录文件和inode对应关系
- 打开文件表 : 记录进程打开的文件信息
- 超级块 : 描述文件系统基本信息



# i-node

- i-node
  - 每个文件用一个i-node来描述
  - 文件元数据
  - ino: inode number , 即i-node的ID , 唯一标识一个文件 (在一个FS内)
- 文件属性信息
  - mode: 文件类型和访问权限
  - size: 文件大小
  - nlinks: 硬链接数
  - uid: 所有者的user ID
  - gid: 所有者的group ID
  - ctime: 文件创建的时间戳
  - atime: 上一次访问文件的时间戳
  - mtime: 上一次修改文件的时间戳



# i-node

- 文件块的索引信息: 文件块的磁盘位置信息
  - $\langle \text{offset}, \text{count} \rangle \rightarrow$  磁盘上的位置
    - ↓
    - 文件块#       $\rightarrow$       磁盘逻辑块# LBN
    - ↓
  - 不同FS采用不同的索引机制



# 目录项 ( dentry )

- 目录内容为它所包含的所有子目录和文件的名字及其ino
  - 不包含子目录的内容
- 逻辑上，目录是一张映射表
  - 目录项 ( dentry ) : 文件名 → ino
- 物理上，目录是一个字节数组
  - 文件名不等长，数组每一项不等长
- 路径解析
  - 根据路径名，获得其ino
  - 逐级目录查找
  - 例子：“/home/os/fs01.ppt”
    - 在根目录下，查找“home”的ino
    - 在“home”目录下，查找“os”的ino
    - 在“os”目录下，查找“fs01.ppt”的ino
    - 根据“fs01.ppt”的ino，找到其数据块，进行读写操作

| 文件名       | ino |
|-----------|-----|
| .         | 125 |
| ..        | 1   |
| vm01.ppt  | 137 |
| homeworks | 153 |
| vm02.ppt  | 138 |
| dev01.ppt | 139 |
| fs01.ppt  | 140 |



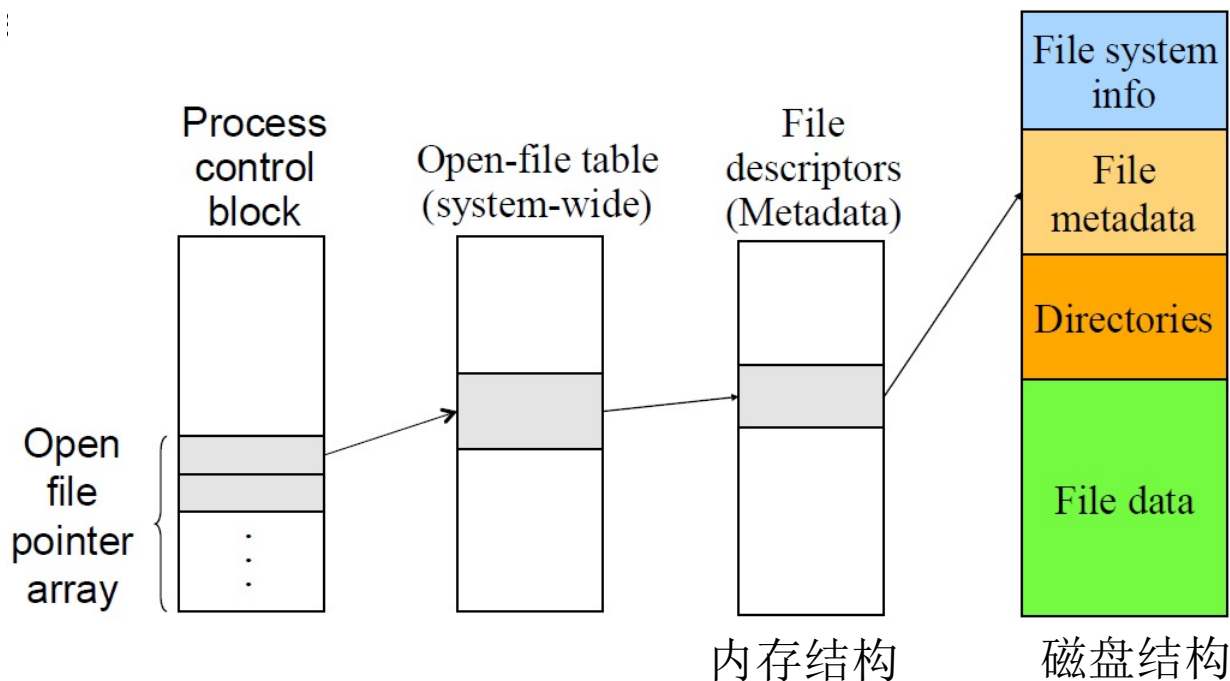
# 打开文件：fd=open(path, flags, mode)

- 打开文件表：Open-file table
  - FS使用打开文件表（在内存中）把进程与文件的i-node进行关联
  - 通过路径名解析和权限检查，得到待访问文件的ino，读出它的 i-node（保存在磁盘上）
  - 将磁盘i-node拷贝至一个内存i-node结构中，在打开文件表中增加一项，包含以下内容
    - 文件的reference count
    - 当前文件的偏移量
    - 文件的访问模式
    - 内存inode结构的指针
    - ...



# 打开文件：fd=open(pname, flags, mode)

- 文件描述符表：file descriptor table
  - 每个进程有一个文件描述符表（open file pointer array/file descriptor table）
    - 指针数组，每个指针指向打开文件表中的一项，表示一个打开文件
    - 该指针在文件描述符表中的下标，即文件描述符fd





# 文件系统物理结构：磁盘布局

|            |             |                 |         |          |                       |
|------------|-------------|-----------------|---------|----------|-----------------------|
| Boot block | Super block | Free space mgmt | I-nodes | Root dir | Files and directories |
|------------|-------------|-----------------|---------|----------|-----------------------|

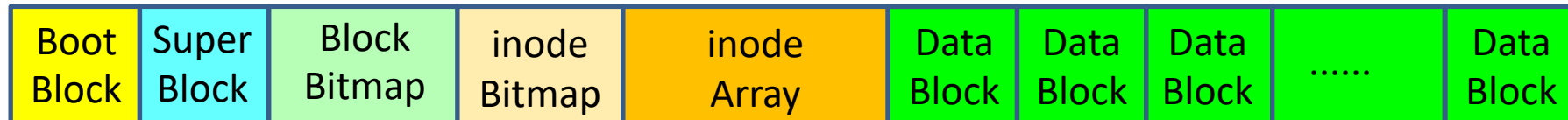
|            |             |              |              |             |            |            |            |       |            |
|------------|-------------|--------------|--------------|-------------|------------|------------|------------|-------|------------|
| Boot Block | Super Block | Block Bitmap | inode Bitmap | inode Array | Data Block | Data Block | Data Block | ..... | Data Block |
|------------|-------------|--------------|--------------|-------------|------------|------------|------------|-------|------------|

- 引导块
  - 启动OS的代码
- 超级块：定义一个FS
  - FS的相关信息
- 空闲空间管理相关的信息
- i-node表
  - 每个i-node描述一个文件或目录
- 数据块
  - 文件块或目录块





# 超级块



- 定义一个文件系统
  - 数据块的大小
  - i-node的大小
  - 数据块总数
  - i-node总数
  - 根目录ino
  - i-node表的起始地址
  - 空闲数据块 ( Block bitmap ) 起始地址
  - 空闲i-node ( i-node bitmap ) 起始地址
- 当前状态
  - 数据块使用情况：已使用的块数、预留的块数、剩余的块数...
  - i-node使用情况：已使用的个数、剩余的个数...



# 总结

---

- FS名字空间是由目录和文件构成的树形层次化结构
  - 文件是无结构的命名字节数组
  - 目录是一种特殊的文件，包含文件和子目录的命名集合
  - 文件和目录有各自的访问接口
- FS结构
  - 虚拟文件系统：提供统一的访问接口，对接不同的物理文件系统
  - 物理文件系统：实际管理磁盘、文件、目录等



# 总结

- FS主要数据结构
  - 超级块：记录一个文件系统的基本信息
  - i-node：定义一个文件/目录，根据ino定位inode的磁盘位置
  - 目录项（dentry）：目录文件中的内容，记录文件名→ino的映射关系
  - 打开文件表：记录目前打开文件的信息，例如访问位置、访问模式，内存inode指针等
  - 超级块：记录文件系统的基本信息