- 第4章 文件管理
  - **basic**
    - concept(3
    -
    - 文件控制块 (**FCB**)→ 索引节点 (**inode**)
    -
    - **操作**(3 2 1
    - **保护**(rwx | ogo
    -
    - 逻辑结构(2
    - 🧮 **物理..**(3分配 + 1索引|count
  - **目录**
    - concept
    - **结构** (3
    -
    - 操作
    - **实现**(2
    -
    - 文件共享(2link
  - **文件系统**
    - **结构**(1 3 1
    - **布局**(2块 3区
    -
    - 🧮 **外存空闲空间管理**(位示图0/1
    -
    - 虚拟文件系统 **(VFS)** 统一
    - 文件系统挂载 (1

好的，各位同学，我们来到第四章《文件管理》。这是操作系统三大核心（进程、内存、文件）之一，也是考研的**高分重灾区**，计算题模式固定，概念题重在理解。本章我们要扮演一个“图书管理员”，搞清楚计算机是如何管理硬盘上成千上万个文件的。

第4章 文件管理

本章的核心是解决两个问题：文件内部数据怎么组织（逻辑与物理结构），以及文件之间怎么组织（目录）。

---

basic

concept(3

- **核心考点:**
  - 文件的定义、属性。
- **形象记忆法:**
  - 文件**系统** = **图书馆**
  - **文件** = **书**
  - 文件**内容** = 书的**正文**
  - 文件**属性** = 书的**版权页** (书名、作者、出版社、ISBN号等)。
    - **名称:** 唯一标识符。
    - **类型:** .txt, .jpg, .exe...
    - **位置:** 文件在硬盘上的存放地址。
    - **大小:** 文件占了多少空间。
    - **保护:** 谁能读、谁能写。
    - **时间:** 创建、最后修改、最后访问的时间。

文件控制块 (FCB)→ 索引节点 (inode)

- **核心考点:**
  - FCB的作用。
  - 索引节点 (inode) 对FCB的改进。
- **形象记忆法:**
  - **FCB** (文件控制块) = 图书的**索引卡片**
    - **内容:** 包含了文件所有的元数据（文件名、属性、物理地址等）。
    - **存放:** 所有的FCB集合在一起，就形成了**文件目录**

        - **缺点:** 如果目录里文件很多，每张卡片信息又全又大，找一本书（文件）就要翻很久的卡片，效率低。

    - 索引节点 (**inode**): 卡片瘦身术
        - **思想:** 把文件名和其它信息分开。
        - **目录项:** 只存放 `文件名` + `索引节点编号 (inode号)`。目录项变得很小。
        - **inode表:** 单独存放所有`inode`，每个`inode`包含了除文件名外的所有其它文件属性。
        - **优点:** 目录检索速度大大加快
            - 现在只需要在一个小本子（目录文件）里根据文件名找到编号
            - 再去对应的仓库（inode表）里取详细信息。

操作(3 2 1

- **核心考点:**
    - 基本文件操作。
    - 文件`打开(open)`和`关闭(close)`的内部过程，以及`文件描述符`的作用。
- **形象记忆法 (图书馆借书):**
    - **基本操作:**
        - `create`: 办一张新的索引卡片，找个空书架。
        - `delete`: 撕掉索引卡片，回收书架。
        - `read/write`: 读/写书的内容。
    - ** `open` **操作 (借书流程):
      a. 用户提供**文件名** (书名)，发起`open`请求。
            - 操作系统（图书管理员）根据书名去**目录**（总索引卡）里查找，找到对应FCB/**inode**
                - 把这张卡片的信息copy到内存里的“**打开文件表**”（今日借阅登记表）中
      b. 管理员返还给用户一个**数字**，叫**文件描述符** (借书证上的编号)。

    - `close`**操作 (还书流程):**

        - 用户提供**文件描述符**，系统根据编号找到“打开文件表”中的条目，释放它，并将可能已修改的元数据写回磁盘。

    - **为什么需要 `open` / `close` ？**
        - 后续的 `read/write` 操作，用户只需提供**文件描述符**（一个数字），系统直接查内存里的“打开文件表”即可，**无需每次都去硬盘上遍历目录**，极大提高了效率。

保护(rwx | ogo

- **核心考点:**
    - 访问控制列表 (**ACL**)。
    - 精简访问控制（拥有者、组、其他）。
- **形象记忆法 (图书馆门禁):**
    - **目的:** 控制谁能对这本书（文件）做什么操作。
    - **访问类型:** `读(r)` 、 `写(w)` 、 `执行(x)` 。
    - **精简访问控制:**
        - **拥有者 (Owner):** 这本书是你捐的，你有最高权限。
        - **组 (Group):** 你所在学院的师生可以借阅。
        - **其他 (Others):** 校外人员只能看，不能借。

逻辑结构(2

- **核心考点:**
    - 文件的两种逻辑结构。
- **形象记忆法 (用户视角看书的内容):**
    - **无结构文件 (流式..):** 书就是一长串没有章节的文字流。如 `.txt` 文件。想找特定内容？从头读到尾。
    - **有..文件 (记录式..):** 书由一个个逻辑单元（记录）组成。如学生信息表，每条记录就是一个学生的信息。

🧮 物理..(3分配 + 1索引count

- **核心考点:**
  - 三种文件分配方式（物理结构）的原理、优缺点对比（**选择题、大题高频**）。
  - **索引**分配的**文件大小**计算 (大题核心)
- **形象记忆法 (硬盘上如何存放书):**
  - **连续分配:**
    - **原理:** 一本书的所有页必须存放在**连续**的书架上。
    - **优点:** 顺序读写速度极快（磁头不用动）。
    - **缺点:** 容易产生**外部碎片**；放书前必须知道书有多厚，不易增删。
  - **链接..:**
    - **原理:** 书的每一页可以放在**任意空闲**的书架位置，每页末尾都写着“下一页在xxx书架”。
    - **优点:** 无外部碎片，书的厚度可动态增减。
    - **缺点:** 只支持**顺序访问**，不支持随机访问（想看第100页，必须先翻前99页）；指针占用空间；一页损坏，后面的都找不到了。

  - **索引..:**
    - **原理:** 每本书配一个**索引块**（目录页），这个索引块记录了书中每一页分别存放在哪个书架上。
    - **优点:** 支持**随机访问**，无外部碎片，书的厚度可动态增减。
    - **缺点:** 索引块本身要占用书架空间。

- **计算题模板 (混合索引文件最大容量):**
  - **场景:** UNIX系统中，一个inode有13个地址项`i.addr[0]~i.addr[12]`。
    - `0-9`: 10个**直接地址**
    -
    - `10`: 1个**一级间接地址**
    - `11`: 1个**二..**
    - `12`: 1个**三..**

- 已知: 磁盘块大小 `B` (如 4KB)，地址项大小 `P` (如 4B)。
- **计算步骤:**
  a. 一个**索引块**能存多少地址: $\boxed{N = B/P}$ (如 `4KB/4B = 1024`)。
  b. 计算**各级**索引能表示的文件大小:
    - **直接地址:** $大小_{直} = 10 \times B$
    - **一级间接:** $大小_{一} = N \times B$
    - **二..:** $大小_{二} = N^2 \times B$
    - **三..:** $大小_{三} = N^3 \times B$
  c. **sum**计算最大文件大小:
    - $最大容量 = 大小_{直} + ..._{一} + ..._{二} + ..._{三}$

目录

concept

- **核心考点:**
  - 目录是实现“按名存取”的关键。
- **形象记忆法:**
  - 目录: 图书馆的**索引目录册**，它的核心作用是记录了“书名”到“书的物理位置”的映射。

结构 (3

- **核心考点:**
  - 单级、两级、树形目录结构的特点。
- **形象记忆法:**
  - **单级目录:** 整个图书馆只有**一本索引册**，所有书的卡片都在里面。**缺点:** 书名不能重复、查找慢。
  - **两级..: 一个总索引册**，下面为**每个读者**配一个私人的索引册。**解决了书名重复问题**。
  - **树形..: 索引册里可以套子索引册**（按类别分）。层次清晰，分类方便。现代OS都用这个。
    - **绝对路径:** 从图书馆大门口 (`/`) 开始的完整寻路指南。

- **相对..:** 从你当前所在阅览室（当前工作目录）开始的寻路指南。

操作

- **核心考点:**
  - 创建、删除、显示目录等基本操作。
- **形象记忆法:**
  - 就是对“索引册”本身的操作，比如增加一个分类、删除一个分类等。

实现(2

- **核心考点:**
  - 线性列表和哈希表两种实现方式。
- **形象记忆法 (索引册的内部实现):**
  - **线性列表:** 索引册里的条目**按顺序排列**。查找时从头翻到尾。
  - **哈希表:** 根据书名通过一个函数**直接算出**它在索引册的第几页。查找速度快，但可能存在“冲突”（两本书算出同一页码）。

文件共享(2link

- **核心考点:**
  - 硬链接 软链接（符号链接）的实现原理和区别（**选择题高频**）。
- **形象记忆法:**
  - **硬链接 (Hard Link):**
    - **原理:** 多个目录项中的**inode**号 指向**同一个inode**。
    - **类比:** 给一本书起了两个不同的书名，但它们指向的是图书馆里**同一本实体书**。
    - **特点:**
      - inode中有一个**链接计数 (link count)**，记录有多少个硬链接指向它。
      - 删除文件时，只是删除目录项，`link count--`。只有当 `link count` 变为0

时，才真正删除文件数据。
        - 不能..（图书馆），不能..（索引册）

- **软.. (Symbolic ..):**
    - **原理:** 创建一个**特殊的文件**，文件内容是**另一个文件的路径名**。
    - **类比:** Windows的“**快捷方式**”。它是一张纸条，上面写着“要找的书在XX分类的XX索引册里”。
    - **特点:**
        - 它有自己独立的inode。
        - 删除**源文件**后，软链接**失效** (纸条还在，但按图索骥找不到了)。
        - 可以**跨**文件**系统**，可以**链接目录**。

文件系统

结构(1 3 1

- **核心考点:**
    - 层次结构
- **形象记忆法:**
    - 类似于I/O软件的层次，从上到下为：
        - user -> **逻辑**文件系统 -> 文件组织**模块** -> **基本**文件系统 -> I/O控制层。
    - 每一层都调用下一层提供的服务，并向上一层隐藏细节。

布局(2块 3区

- **核心考点:**
    - 文件系统在磁盘上的整体布局
- **形象记忆法 (磁盘分区蓝图):**
    i. **引导**块 (Boot Block): 分区的“启动器”，存放启动该分区OS的代码。

ii. **超级**块 (Super ..): ..“说明书”，记录整个文件系统的核心信息（块大小、inode数量、空闲块信息等）。
    - 其**损坏**，文件系统就**崩溃**。

---

iii. **空闲空间管理区:** 记录哪些盘块是空闲的（如位示图）。
iv. **inode..:** 集中存放所有文件的inode
v. **数据..:** 存放所有文件的实际数据

🧮 外存空闲空间管理(位示图0/1

- **核心考点:**
    - 位示图法的原理和计算。
- **形象记忆法:**
    - **位示图 (Bitmap):** 用一个二进制位串来表示所有磁盘块。
        - `0` 代表该块**空闲**。
        - `1` ..**已占用**。
        - **优点:** 查找连续的空闲块非常方便。

- **计算题模板 (位示图定位):**
    - 已知:
        - 磁盘块号 `i`
        - 每行/每字的位数`w` (如32)
    - **key:** 块号`i`是从0 or 1开始
        - 从**0**开始:
            - 在位示图的第 `j` 行 (字): $j = \frac{i}{w}$
            - 在第 `j` 行的第 `k` 位: $k = i\%w$
        - ..**1**..:
            - ..: $j = \frac{(i-1)}{w}$
            - ...: $k = (i-1)\%w$

虚拟文件系统 (VFS) 统一

- **核心考点:**
  - ..作用
- **形象记忆法:**
  - VFS: 标准化**接口**or**翻译官**
  - **作用:**
    - Linux/Windows等系统支持多种文件系统（如FAT32, NTFS, EXT4）。
    - VFS提供了一个**统一的、抽象的接口**。用户程序的所有文件操作都先发给VFS，VFS再根据具体的文件系统类型，“翻译”成对应的操作指令发给具体的文件系统驱动。
  - **目的:**
    - 屏蔽底层不同文件系统的差异，使用户程序可以用同样的方式操作任何文件系统的文件。

文件系统挂载 (1

- **核心考点:**
  - 挂载的概念
- **形象记忆法 (接入新书架):**
  - 你有一个主图书馆（根文件系统`/`）。现在运来一个装满书的移动书架（U盘）。
  - **挂载 (Mount):** 把这个移动书架推到图书馆某个空房间（如`/mnt/usb`这个空目录），并挂上牌子。
  - 之后，访问`/mnt/usb`目录，看到的就是这个移动书架上的书了。

第四章的内容与实际使用联系紧密，通过“图书馆”这个核心比喻，可以将文件、目录、FCB/inode、打开/关闭等概念串联起来。计算题主要集中在索引文件大小和位示图定位，掌握模板即可轻松应对。祝大家复习顺利！