**操作系统课程实践实验报告**

实验题目：简单文件系统

目 录

[一 题目介绍 1](#_Toc515888058)

[二 实验思路 1](#_Toc515888059)

[三 遇到问题及解决方法 2](#_Toc515888060)

[四 核心代码及实验结果展示 4](#_Toc515888061)

[4.1 实验小组分工 4](#_Toc515888062)

[4.2 核心代码及实验结果 4](#_Toc515888063)

[五 个人实验改进与总结 5](#_Toc515888064)

[5.1 个人实验改进 12](#_Toc515888065)

[5.2 个人实验总结 13](#_Toc515888066)

[六 参考文献 14](#_Toc515888067)

# 一 题目介绍

目的

# 通过具体的⽂件存储空间的管理、⽂件物理结构、⽬录结构和⽂件操作的实现，加深对⽂件系统内部数据结构、功能以及实现过程的理解。

内容

1. 在磁盘上创建一个文件作为磁盘镜像，在其上实现⼀个简单的基于多级⽬录的单⽤户单任务系统中的⽂件系统。

2. ⽂件物理结构可采⽤混合索引结构。

3. 空闲磁盘空间的管理可选择索引表、位示图或其他办法。

4. ⽂件⽬录结构采⽤多级⽬录结构。每个⽬录项应包含⽂件名、物理地址、⻓度等信息，还可以通过⽬录项实现对⽂件的读写保护。

5. 实现基本的功能

实验内容

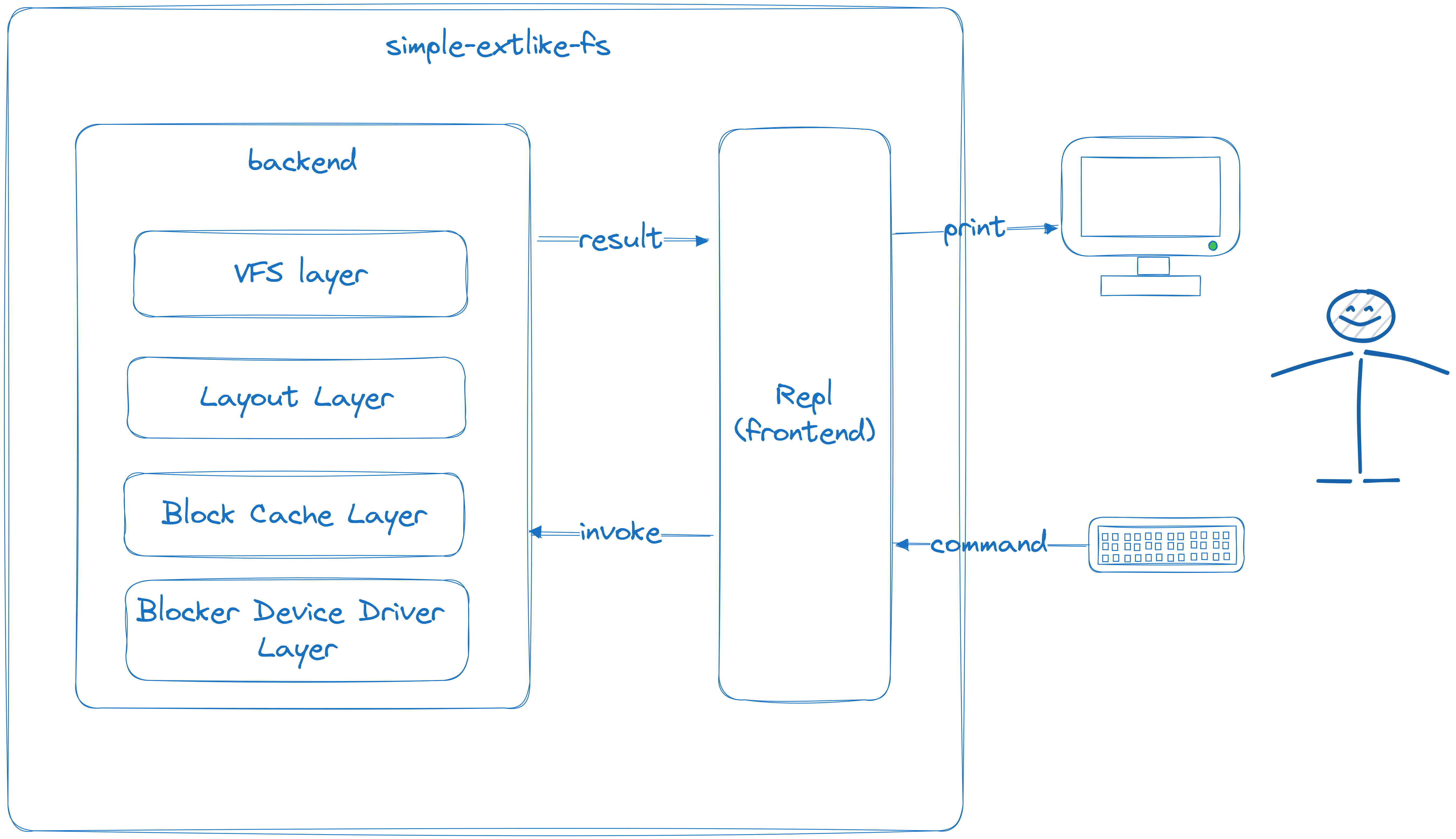
实现操作文件系统的基本命令

1. cd <new-dir> 进入一个新目录
2. pwd 打印当前工作目录
3. ls 列出所有目录项
4. close <fd> 关闭对应的fd表项
5. open <filename> <flags> 用对应的<flags>打开<filename> flags可以是r(允许读), w(允许写), a(把读写指针定位到文件末尾), c(如果文件不存在则创建对应的文件), t(把文件大小截断到0)的任意组合。打开后打印对应的fd
6. read <fd> <size-to-read> 从对应的fd中读入<size-to-read>字节的数据并打印在终端
7. write <fd> 从终端读入字符串，写入对应<fd>
8. mkdir <new-dir> 创建新目录
9. rm <filename> 删除对应文件
10. create <filename> 创建文件
11. flush 把缓存中的内容全部写到磁盘
12. exit 把缓存中的内容全部写到磁盘，然后退出交互环境

# 二 实验思路

本文件系统是一个extlike的文件系统，代码架构参照rcore设计。

代码架构如下



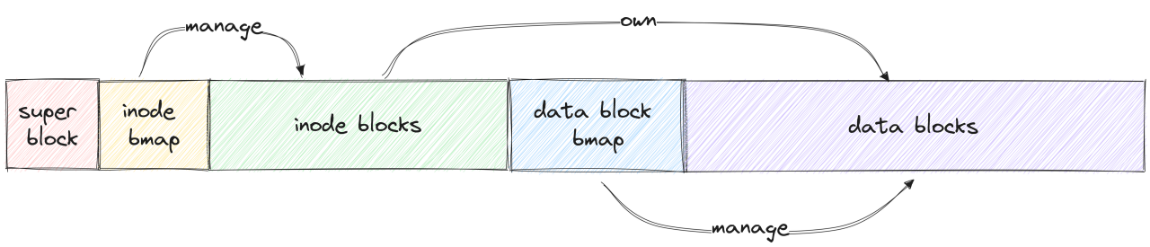
本文件系统大致可以分为前后端。前端也是就是外壳是和用户交互的交互式环境。解析用户输入的命令并调用后端的接口，再把结果打印在终端。后端也就是内核是代码的核心部分，实现文件系统的接口，供前端调用。

后端代码大致可以分为四层：块设备驱动层、块缓存层、布局层、虚拟文件系统层。

块设备层实现磁盘按块读写的设备驱动接口。

块缓存层实现一层缓存。上层代码并不直接访问磁盘。因为磁盘的访问性能不如访问内存。把磁盘数据缓存在内存内能大大提升文件系统读写性能。

布局层对块设备进行划分，大致划分五类区域



超级块是块设备的第0块，主要用来描述文件系统的整体信息：

一个魔数用来判断文件系统的有效性，

块总数，

inode位图占的块数，

inode块占的块数，

数据块位图的块数，

数据块的块数，

根inode的块数

inode位图块用来管理inode的分配与回收： inode位图块中的每一个位都代表对应inode是否已经被分配出去，分配inode时找到为0的位，置1并返回对应inode号。回收时把inode号对应的位置0.

inode块存储所有inode：每一个inode都是对应文件的fcb存储了对应文件的大小，文件的类型（普通文件或者是目录），和一组混合索引管理inode管辖的数据块。

数据块位图：数据块位图的每一位都代表对应的数据块是否已经被分配出去。

数据块：存储文件数据

vfs层：在下面三层的基础上实现文件系统的基本接口 读写文件、删除创建文件目录、获取某目录下的目录项...

# 三 遇到问题及解决方法

# 混合索引管理文件所属的数据块的代码出错，导致逻辑数据块号转换成物理代码块号的过程出错。修改对应虚实块号转换的函数

1. 位图分配器分配回收代码出错，导致同一个数据块被多次分配，修改对应的分配回收逻辑
2. 在同一个进程中用同一把锁lock了两次，导致进程死锁，梳理并发逻辑，修改死锁错误

# 四 核心代码及实验结果展示

## **4.1 实验小组分工**

## 21051331叶福伦一人完成

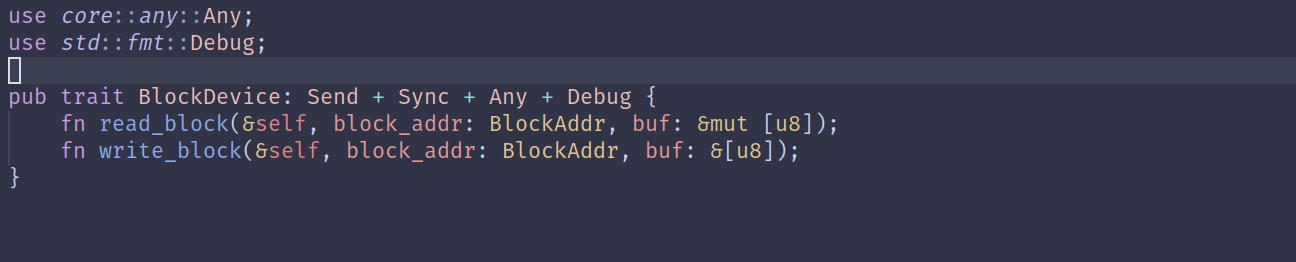
## 4.2 核心代码及实验结果

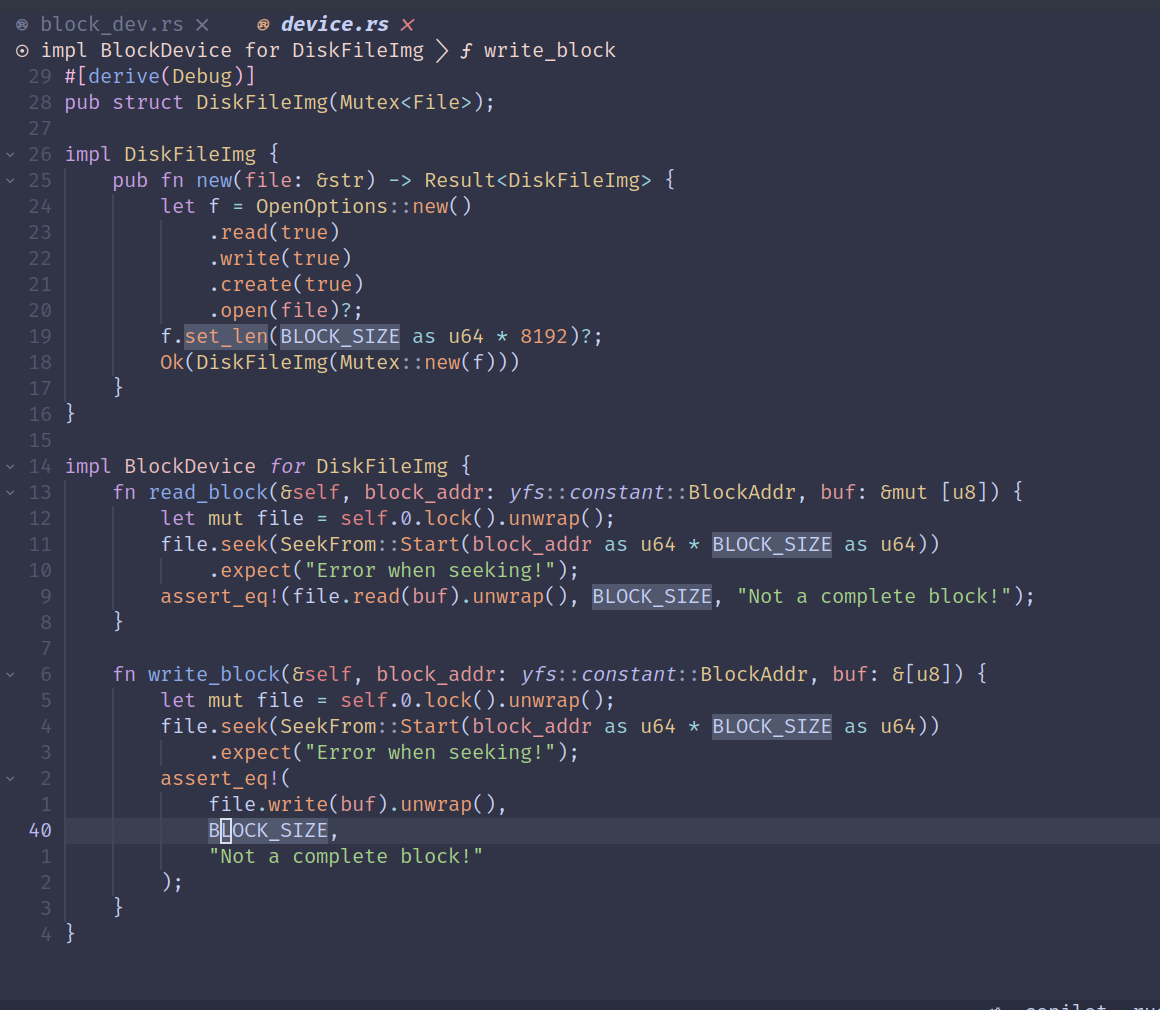
实验结果

# 

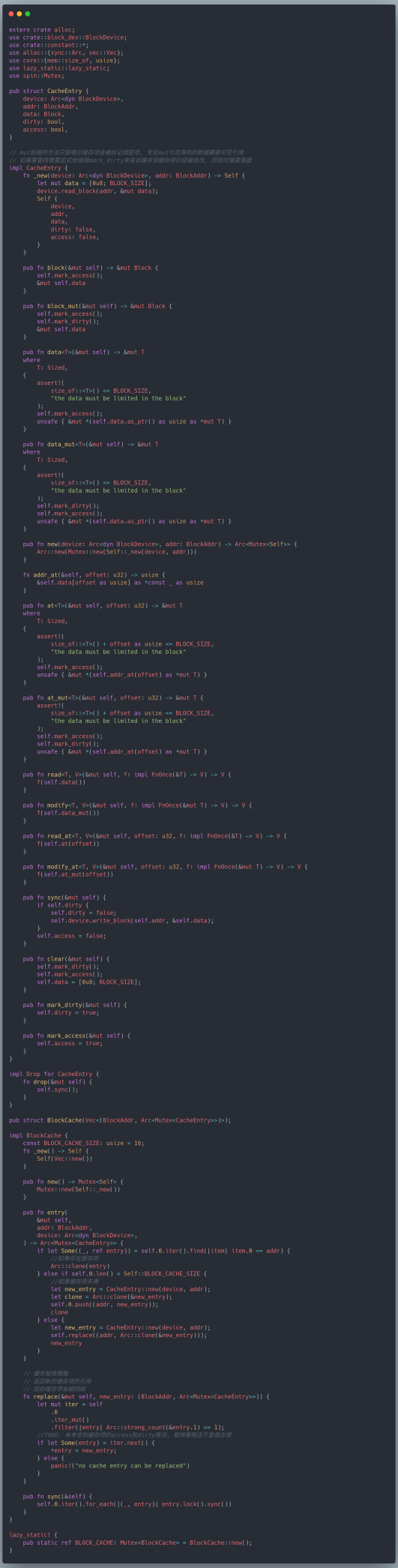
# 核心代码：

设备驱动层：

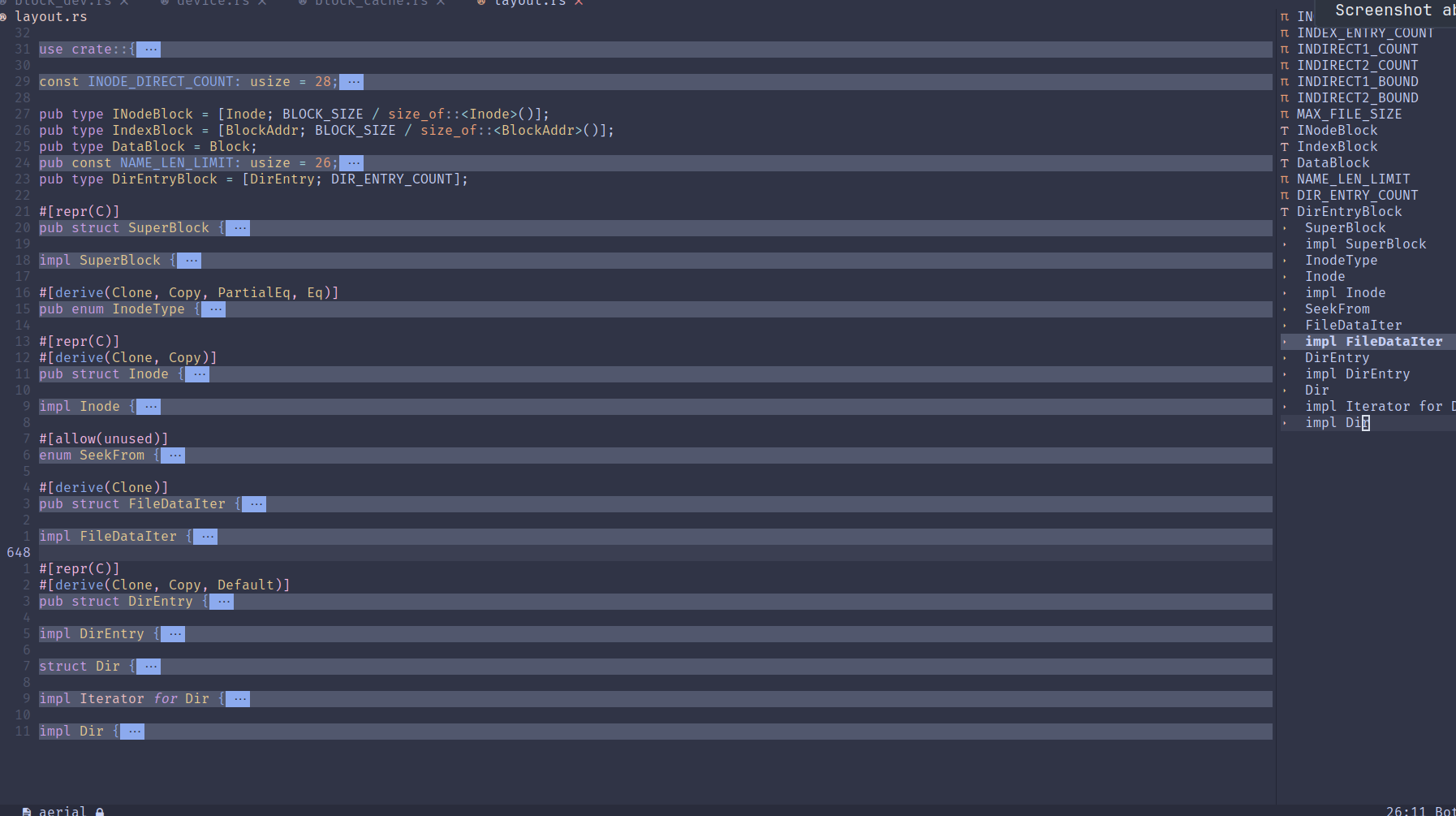


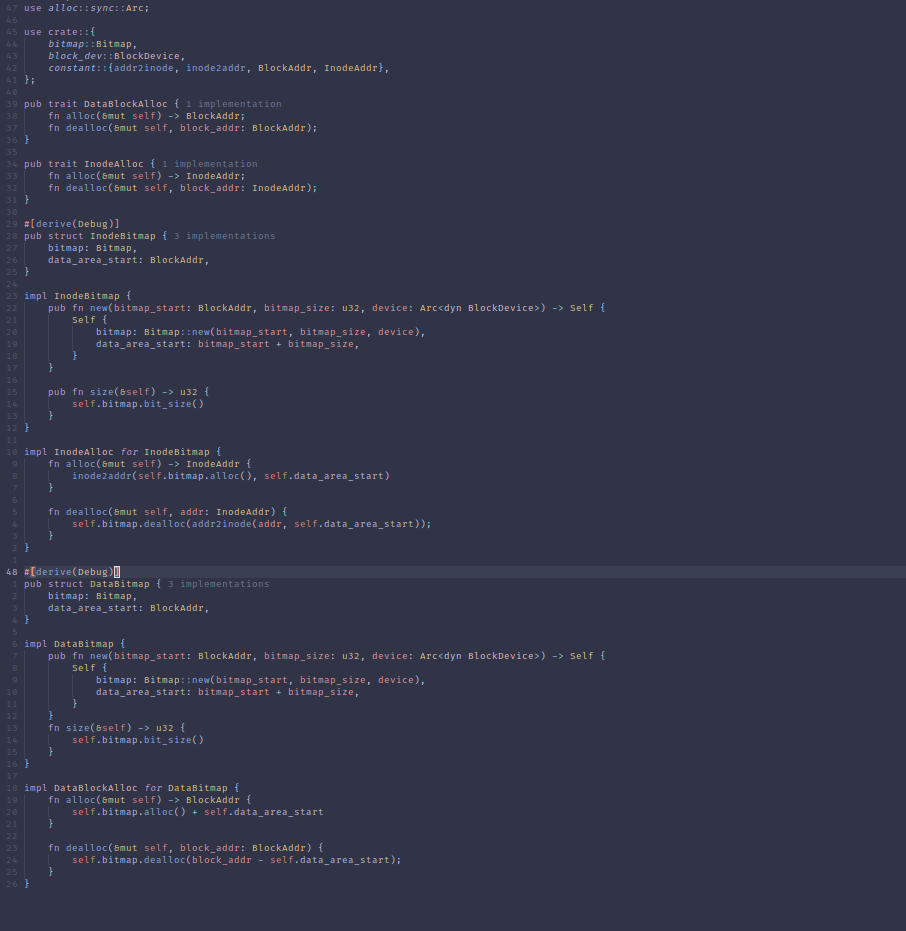
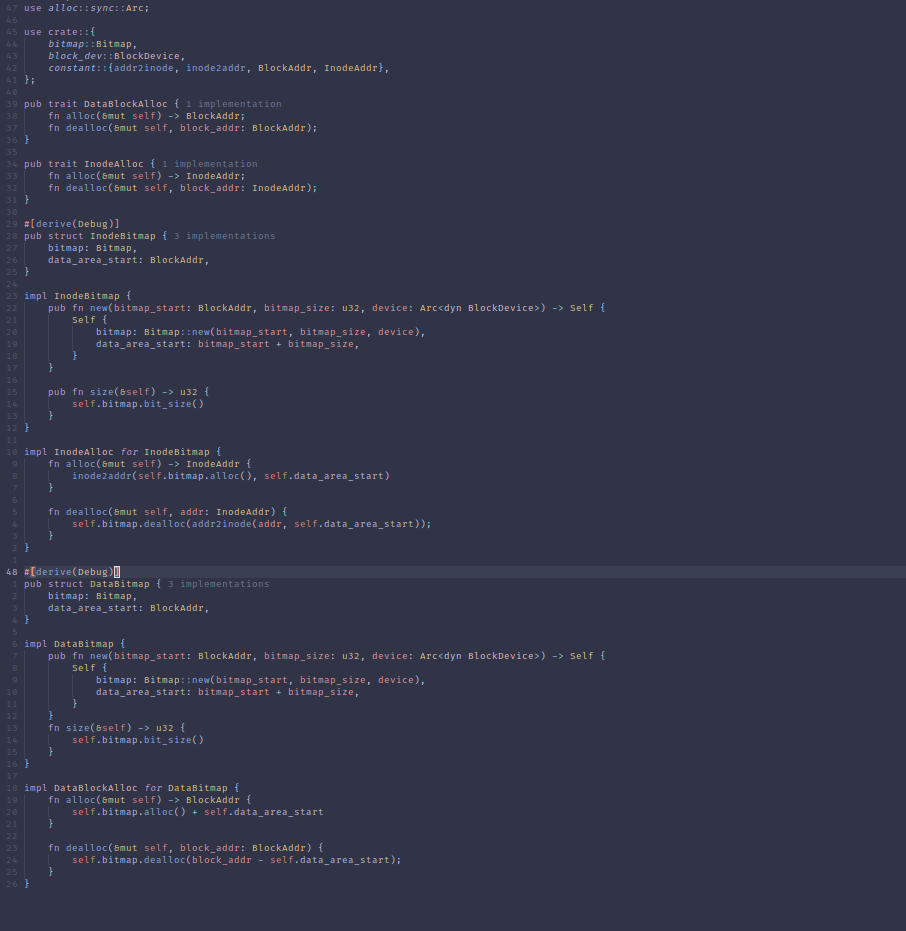


块缓存层：



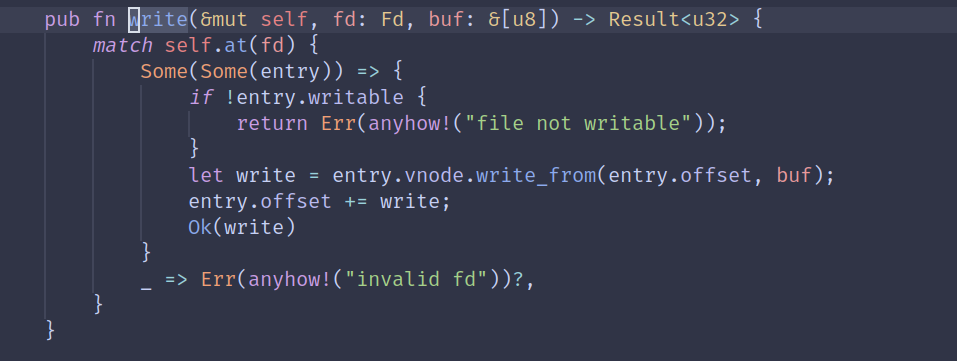
布局层：

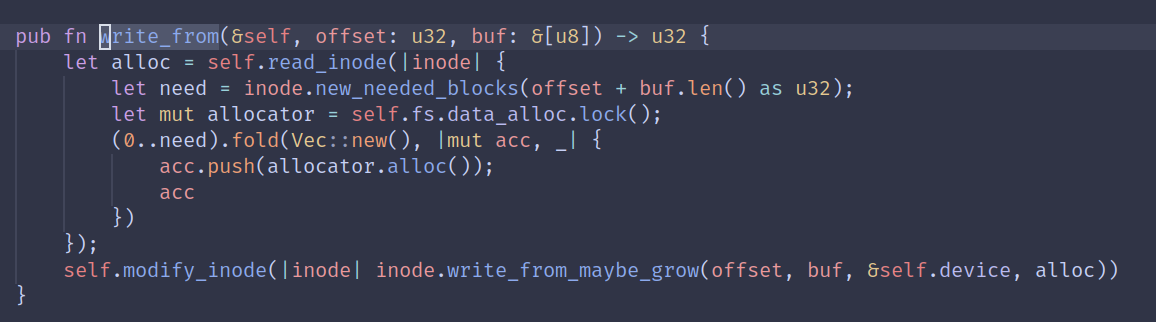


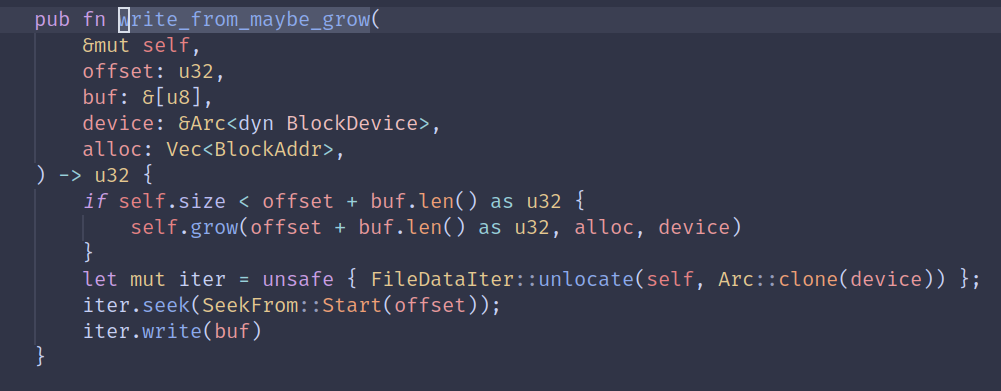


vfs层：

写文件：

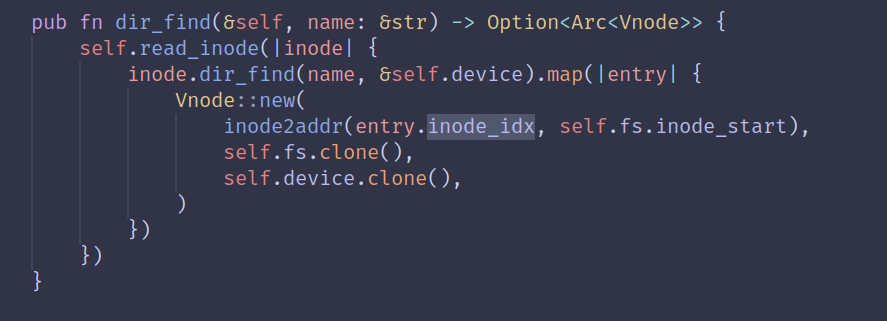


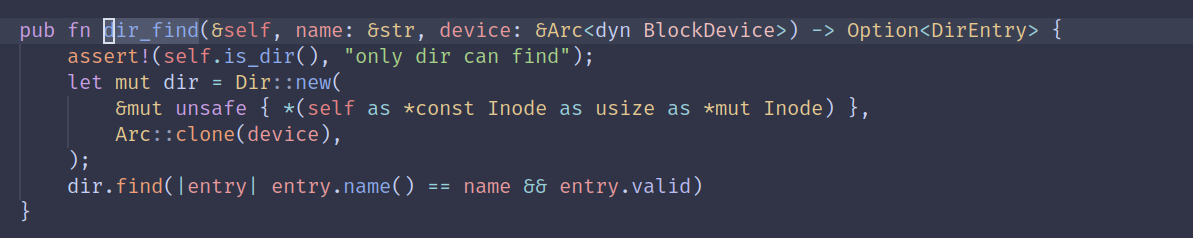




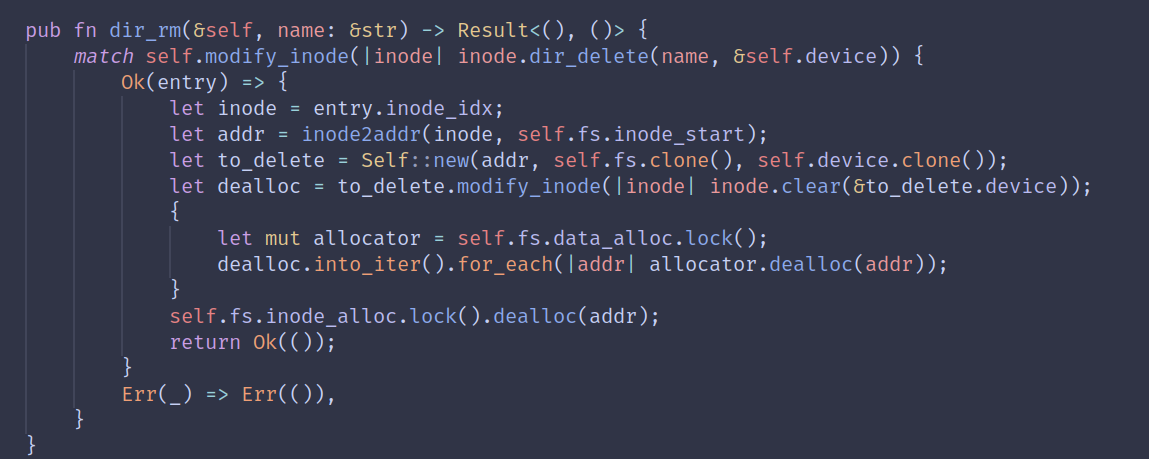


查找目录项：





删除文件：



# 五 个人实验改进与总结

## 5.1 个人实验改进

相比书本上的FAT模拟系统有一些明显的不同，我使用ext作为参考对象，使用文件作为磁盘镜像和实现了缓存层这两个方面。选择了使用文件作为磁盘镜像，而不是在内存中模拟一个大块的磁盘。这种方式更接近实际的文件系统实现，因为真实的文件系统通常是存储在硬盘或其他媒体上的文件。模仿ext文件系统的实现方式。这意味着使用了更复杂的数据结构和算法，例如inode表、块组描述符等。实现了一个简单的缓存层来提高文件系统性能。这是一个很好的优化，可以显著提高文件系统的访问速度。在实际的文件系统中，缓存通常用于减少对磁盘的频繁访问，提高文件的读写性能。

## 5.2 个人实验总结

在完成操作系统实验课的文件系统设计与实现过程中，我收获颇丰，不仅深化了对文件系统原理的理解，还锻炼了编码和系统设计的能力。以下是我在这次实验中的一些心得体会：

1. 实际应用能力提升：通过实现一个文件系统，我深刻理解了文件系统在操作系统中的关键作用。这次实验不仅是对课堂知识的巩固，更是将理论知识应用到实际中的宝贵机会。

2. 设计灵活性与抽象能力： 在实现文件系统时，我面临了许多设计决策，例如文件分配策略、目录结构等。这促使我学会权衡不同设计选择，提高了设计灵活性和抽象能力。

3. 错误处理： 在实验中，我重视了错误处理机制。这帮助我更好地理解如何处理异常情况，并为系统出现问题时提供了可靠的故障排除手段。

4. 代码质量和可维护性：编写文件系统代码的过程中，我注重了代码的清晰结构和良好的注释。这对于他人理解我的代码、以及未来的修改和维护都具有重要意义。

5. 实验挑战与成就感： 面对一项复杂的任务，我经历了挑战和困难。但通过不断的努力和思考，我成功地完成了文件系统的实现，这带给我了巨大的成就感。

总的来说，这次实验不仅仅是技术水平的提升，更是培养了解决问题和项目管理的能力。通过这个过程，我更深刻地理解了操作系统中文件系统的核心概念，并在实际应用中得到了验证。这为我未来在系统开发领域的探索奠定了坚实的基础。

# 六 参考文献

[https://rcore-os.cn/https://rcore-os.cn/](https://rcore-os.cn/https:/rcore-os.cn/)

<https://github.com/MoefulYe/ycore>

<https://github.com/MoefulYe/os-lab>