**《操作系统课程设计》**

**（2023/2024学年第一学期）**

**指导教师：田秋红**

**郭奕亿**

**班级：21计算机科学与技术留学生（1）**

**学号：2021529620004**

**姓名：LOW REN HONG**

1. **题目：**

B类和C类

**二、设计目的：**

操作系统原理是计算机专业的核心课程。本课程设计的目的旨在加深学生对计算机操作系统内核的理解，提高对操作系统内核的分析与扩展能力。在课程理论教学中，较多地是讲解操作系统理论和实现原理，为将来在基于Linux的嵌入式系统开发或在Java虚拟机上的软件开发工作奠定基础。

**三、计划安排：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **时间** | | **地点** | **工作内容** | **指导教师** |
| 11.22 | 下午 | 10-413、10-414 | 任务布置，相关要求介绍与选题 | 田秋红、郭奕亿 |
| 晚上 | 10-413、10-414 | 完成选题报告和需求分析文档 | 田秋红、郭奕亿 |
| 11.29 | 下午 | 10-413、10-414 | 算法分析与基本设计 | 田秋红、郭奕亿 |
| 晚上 | 10-413、10-414 | 完成基本设计文档 | 田秋红、郭奕亿 |
| 12.6 | 下午 | 10-413、10-414 | 编写代码 | 田秋红、郭奕亿 |
| 晚上 | 10-413、10-414 | 编写代码 | 田秋红、郭奕亿 |
| 12.13 | 下午 | 10-413、10-414 | 程序测试 | 田秋红、郭奕亿 |
| 晚上 | 10-413、10-414 | 程序测试 | 田秋红、郭奕亿 |
| 12.20 | 下午 | 10-413、10-414 | 撰写课程设计报告，答辩 | 田秋红、郭奕亿 |
| 晚上 | 10-413、10-414 | 撰写课程设计报告，答辩 | 田秋红、郭奕亿 |

**四、参考资料**

1. Gray Nutt．Kernel Projects for Linux（影印版）．北京：机械工业出版社,2002
2. 李善平，郑扣根．Linux操作系统计实验教程．北京：机械工业出版社,1999
3. 印旻．Java语言与面向对象程序设计．北京：清华大学出版社，2000
4. https://blog.csdn.net/qq\_40989769/article/details/125223591超专业解析|linux文件系统的底层架构及其工作原理
5. https://www.cnblogs.com/theseventhson/p/15622853.html[linux源码解读（三）：文件系统——inode](https://www.cnblogs.com/theseventhson/p/15622853.html)
6. https://www.cnblogs.com/theseventhson/p/15643658.html[linux源码解读（五）：文件系统——文件和目录的操作](https://www.cnblogs.com/theseventhson/p/15643658.html)

**目 录**

[1.引言 2](#_Toc120274428)

[1.1任务要求 2](#_Toc120274429)

[1.2 选题 2](#_Toc120274430)

[2.需求分析与设计 2](#_Toc120274431)

[2.1 需求分析 2](#_Toc120274432)

[2.2系统框架和流程 2](#_Toc120274433)

[2.2.1 Linux文件系统源码解读绘制功能框架图 2](#_Toc120274434)

[2.2.2 本系统功能框架和流程图 3](#_Toc120274435)

[2.3 文件系统流程和模块描述 6](#_Toc120274436)

[2.3.1 类图 6](#_Toc120274437)

[2.3.2 用例图 6](#_Toc120274438)

[2.3.3 顺序图 6](#_Toc120274439)

[3.数据结构 6](#_Toc120274440)

[4.关键技术 6](#_Toc120274441)

[4.1初始化 7](#_Toc120274442)

[4.2分治法 7](#_Toc120274443)

[5.运行结果 7](#_Toc120274444)

[5.1 运行环境 7](#_Toc120274445)

[5.2 服务模式 7](#_Toc120274446)

[5.3 运行结果 8](#_Toc120274447)

[5.4 实验结果分析 9](#_Toc120274448)

[6.调试和改进 10](#_Toc120274449)

[7.心得和结论 11](#_Toc120274450)

[7.1结论和体会 11](#_Toc120274451)

[7.2进一步改进方向 11](#_Toc120274452)

[7.3 分析设计方案对系统安全的影响 11](#_Toc120274453)

[主要参考文献 11](#_Toc120274454)

# 引言

随着计算机的普及以及网络的发展，合理的处理任务的效率成了一个重要的问题。生产者和消费者是个经典的并发程序问题以及多用户文件设计允许多个用户同时访问同一个文件系统，并通过多线程的调度来尽可能地优化其效率

## 1.1任务要求

B类

用Java语言模仿生产者和消费者的问题

1. 通过Java语言中的wait（）和notify（）命令模拟操作系统中的P/V操作；
2. 为每个生产者／消费者产生一个线程，设计正确的同步算法
3. 每个生产者和消费者对有界缓冲区进行操作后，即时显示有界缓冲区的当前全部内容、当前指针位置和生产者／消费者线程的自定义标识符。
4. 生产者和消费者各有两个以上。
5. 多个生产者或多个消费者之间须共享对缓冲区进行操作的函数代码。
6. 在什么样的情况下

在多线程的环境下，要确保生产者和消费者的同步问题

1. 要做什么事或为什么要这样做

使用Java语言中的wait()和notify()命令来实现

1. 这样做有什么好处

使用wait()和notify()可以有效的进行线程之间的协调和同步，确保生产者和消费者线程安全的访问缓冲区

1. 要解决的问题

避免多个线程同时访问缓冲区导致的数据不一致的问题

避免死锁

进程的饥饿

C类

文件系统

1. 解读部分Linux文件系统源码，绘出功能结构框图；
2. 本系统为多用户系统，可定义文件或目录的访问权限；
3. 模拟文件的存储和索引，整个文件系统为树状结构；
4. 实现文件/目录的基本命令，类似于Linux终端界面；
5. 实现文件保护操作，满足读读允许、读写互斥、写写互斥；
6. 在什么样的情况下

实现一个简单的文件系统

1. 要做什么事或为什么要这样做

实现多用户系统，理解文件系统的层次结构，完成基本的文件系统create、open、close、read/write等基本功能，并实现文件保护操作。

1. 这样做有什么好处

确保了文件的隔离性和安全性，树状的结构能够用户更方便的查找文件

1. 要解决的问题

实现对文件的保护

文件的权限

系统的命令实现

## 1.2 选题

B类和C类

# 需求分析与设计

## 需求分析

* 1. **生产者与消费者**

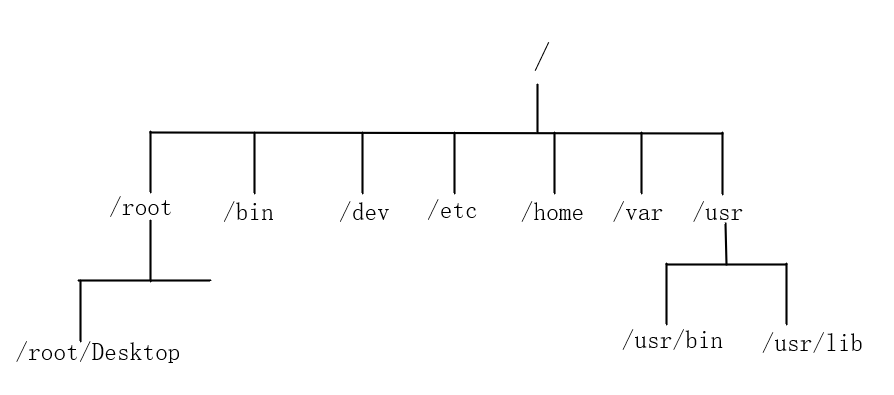
# 生产者和消费者的问题里我们会有两个主要的线程：生产者和消费者。涉及到生产者将数据放入缓冲区而消费者则从缓冲区取出数据。解决问题的方法有这几种一个是同步法还有一个是阻塞队列以及等待唤醒机制。Java里同步的方法有wait()/notifyAll()，wait()/notify()，Semaphore(信号量)，BlockingQueue阻塞队列，await()/signal()。生产者占用着存储空间，消费者等生产者让出空间才能取出数据，互相等待就会发生死锁。

* 1. **文件系统**

实现多用户文件系统，要实现Login用户登入，确保用户登入后才可以对对应的系统做出操作，进入以后是树状的目录，create、open、close、read/write等基本功能。

## 2.2系统框架和流程

2.2.1 Linux文件系统源码解读绘制功能框架图



/用来表示根目录

/root表示root用户的家

/bin表示存放系统命令，普通用户和root用户都可以执行

/dev设备文件保存设置

/etc配置文件保存位置

/home普通用户的主目录

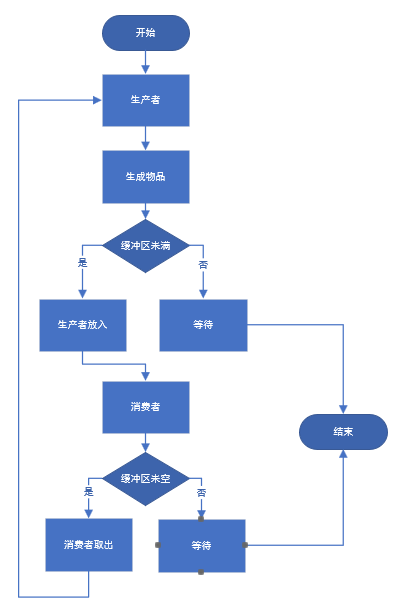
/var是个变量文件

/usr是用于存储只读用户数据

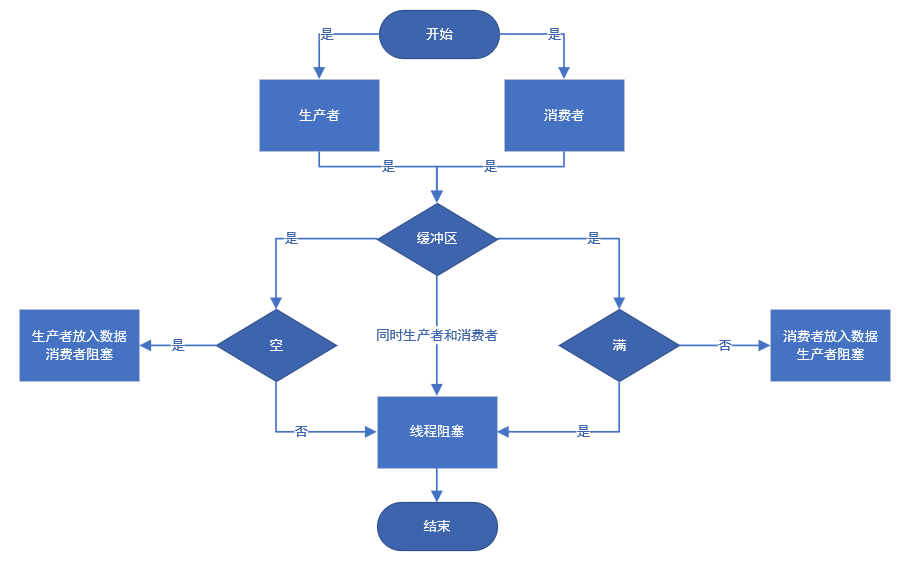
/usr/bin 存放系统命令，普通用户和超级用户可以执行

/usr/lib应用程序调用的函数库保存位置

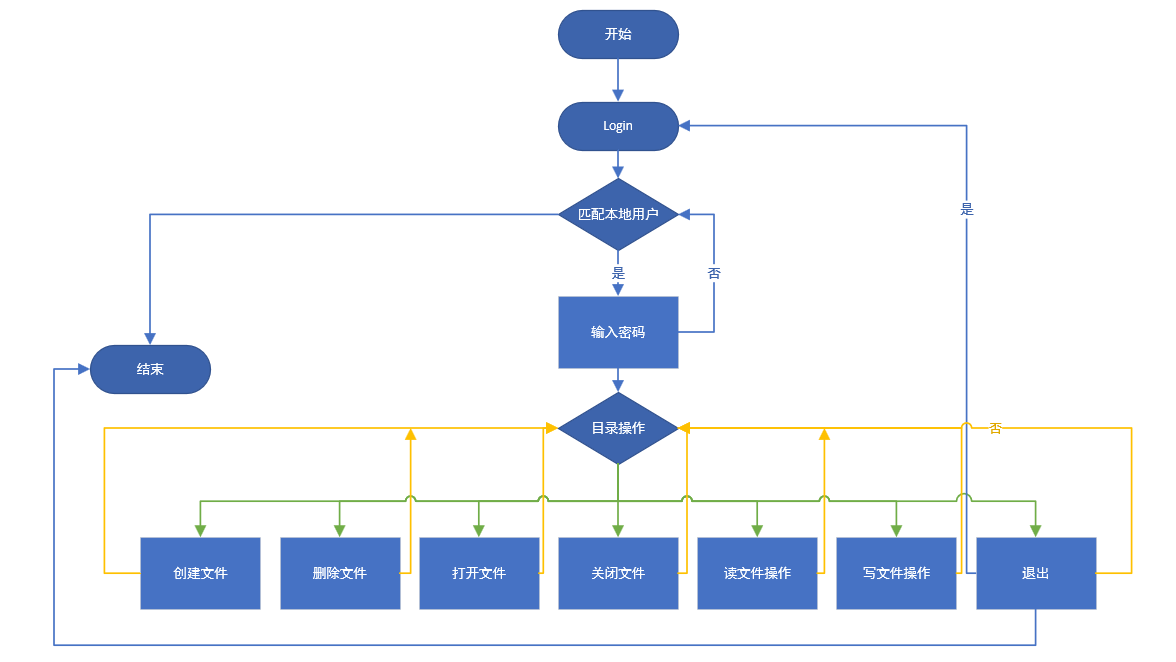
2.2.2 本系统功能框架和流程图



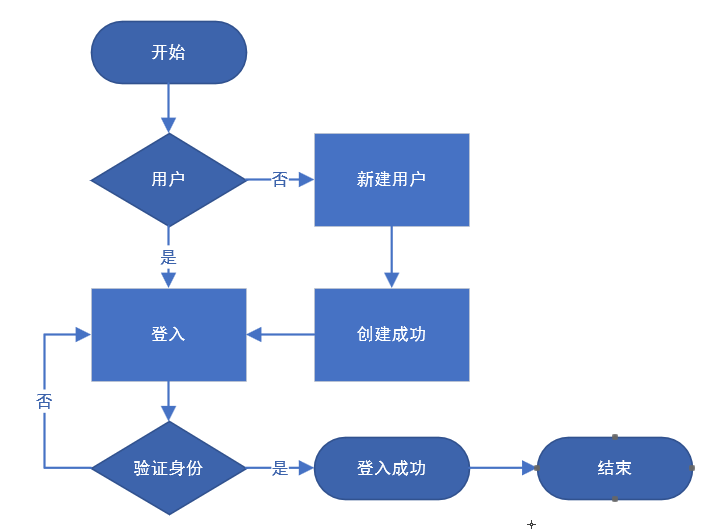
生产者和消费者的流程图顺序执行



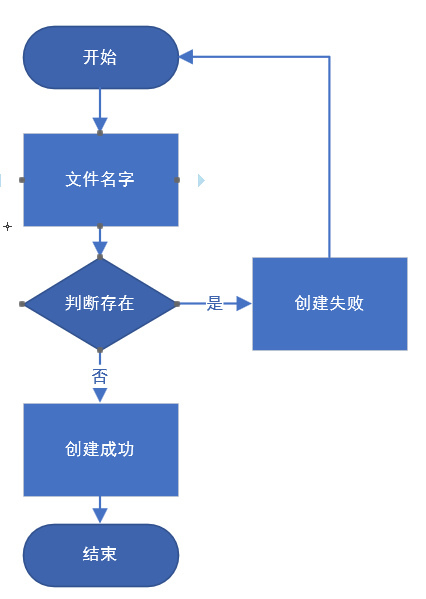
生产者和消费者的流程图并发执行



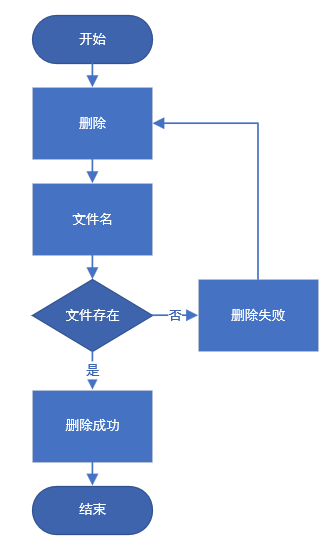
文件流程图



用户的流程图



创建新文件流程图



删除文件流程图

## 

## 2.3 文件系统流程和模块描述

文件系统需要登入功能以及注册用户用能，也要实现对于文件的增删改查的功能。

主要实现：

1. 用户登入功能
2. 新建文件
3. 删除文件
4. 修改文件
5. 列文件目录

用户管理

1. 系统设置userMap<String username , User user>充当用户数据库，在磁盘上分配以磁盘块专门存放用户数据，系统运行之后就读入内存。
2. 登录（login）进行用户名密码校验，登录后可显示用户上一次登录时间（这一点是仿造linux进行的添加），登录后默认进入每个用户的同名专属文件夹，设置内存curUser为当前用户。用户可以使用cd命令切换到其他有权限目录。
3. 注册时进行用户名重名检测，将用户名、密码添加到用户数据库，注册成功后为用户添加一个同名用户文件夹，设置权限为（rwx—）只允许用户自己访问，同时加入根目录下。
4. 用户退出时系统会自动进行数据持久化保存，保证数据不丢失。

文件操作模块：

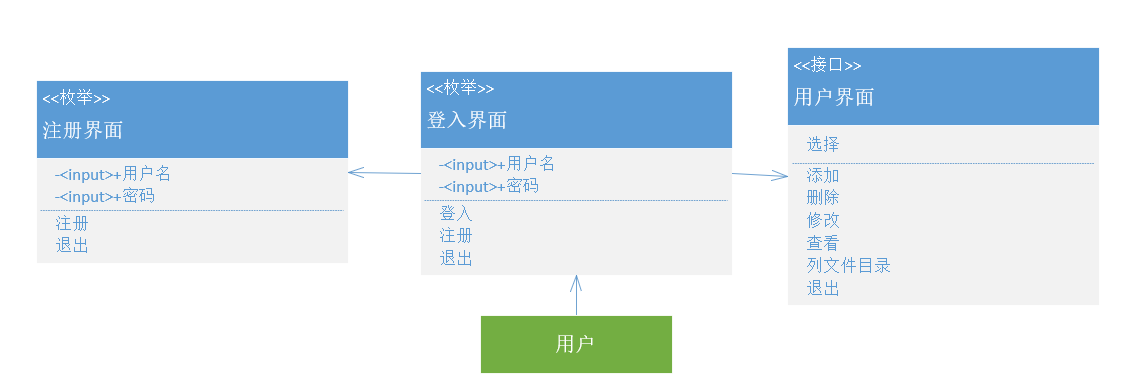
文件操作包括创建（create）文件、打开（open）文件、关闭（close）文件、显示打开的文件（show\_open）、读取（read）文件内容、覆盖写入或追加写入（write）文件、删除（delete）文件、文件重命名（rename）等操作。本系统的所有文件操作实现了跨目录操作，只要有权限以及文件存在便可以进行操作，而不是只能操作当前目录文件。

1. create：用户输入create + 文件名，系统进行文件名判空和重名检测 如无重名文件即进行创建。创建过程包括创建文件索引结点和文件控制块FCB，文件大小为空文件，将FCB存入磁盘，修改父目录的文件项，加入到父目录的儿子集合。
2. open: 用于打开文件，用于在对文件进行操作前需要先打开文件，这样可以方便操作系统分配资源使用权比如外设的管理。打开文件包括解析文件路径是否存在，解析是否是普通文件，判断用户权限，判断是否重复打开，然后封装为openFile加入到openFileList中。系统可选择打开多个文件。
3. close：用于关闭文件，只有关闭文件才可以释放资源使用权，别的进程才可以进行使用该资源，用于在进行logout操作之前需要先确保关闭了打开的文件，否则无法正常退出。关闭过程包括：解析文件路径是否存在，解析是否是普通文件，判断是否在openFileList中。
4. show\_open：用于用户查看打开了什么文件，方便用于进行正常关闭操作。
5. vim：读取文件内容并显示出来给用户看。读取包括解析文件路径是否存在，解析是否是普通文件，判断用户权限，判断是否打开，必须先打开才能进行读取，随后进从文件的FCB中读取到第一块存放的位置，然后使用FAT表到磁盘进行一一读取。
6. write：用于用户对文件写入内容。写入有两种模式：覆盖写与追加写，新文件默认是覆盖写。写入过程包括：解析文件路径是否存在，解析是否是普通文件，判断用户权限，判断是否打开，必须先打开才能进行写入。如果是覆盖写入，则先清除文件的旧FAT表分配关系，再进行分配，将用户输入内容分块写入磁盘，修改FAT表及位示图，修改父目录文件项及递归修改父目录的文件大小；如果是追加写入，则找到FAT中指定的最后一块，在此基础上继续分配。
7. delete：删除文件，允许用户删除文件及空目录。删除过程包括：解析文件路径是否存在，解析是否有权限，删除文件需要有rw权限，删除文件夹还需要执行权限。判断是否打开，如果打开则要先关闭才能删除，随后修改父目录大小及文件项，修改FAT表及位示图
8. rename：文件重命名操作。过程包括解析文件路径是否存在，解析是否是普通文件，判断用户权限，判断是否打开，判断文件名是否重复，随后修改文件的FCB中的名称字段。
9. dir操作与linux的ll指令实现效果基本一致，可以显示目录下文件的文件类型、目录权限、创建者、文件大小、修改时间、文件名。例如：drwx— 4 root 32 2022.10.1 a.txt 同时，对于不同后缀文件格式，也仿照linux进行不同颜色显示，白色表示普通文件、绿色表示可执行文件、红色表示压缩文件、蓝色表示目录文件。
10. mkdir：默认在当前目录下创建新目录项。创建过程包括目录名判空，目录名判断重复，创建索引结点，创建FCB，存入磁盘，修改父目录文件项，加入到父目录儿子集合。
11. cd：切换目录操作，可切换到用户指定的有权限目录。cd … cd …/表示切换到上级目录。切换目录包括解析目录路径，判空，判断是否是目录文件，判断权限，如果都正常即切换目录并更新内存指定的当前目录。
12. ls：用于显示当前目录的文件名。过程包括获得内存指定的当前目录FCB，遍历输出儿子集合目录项。
13. pwd：模仿linux操作，指令输入时不直接显示当前相对于根目录的全路径，而是显示当前目录名而已，使用pwd操作才可以获得全路径。过程包括逐层往上遍历直到根目录。

文件存储模块：

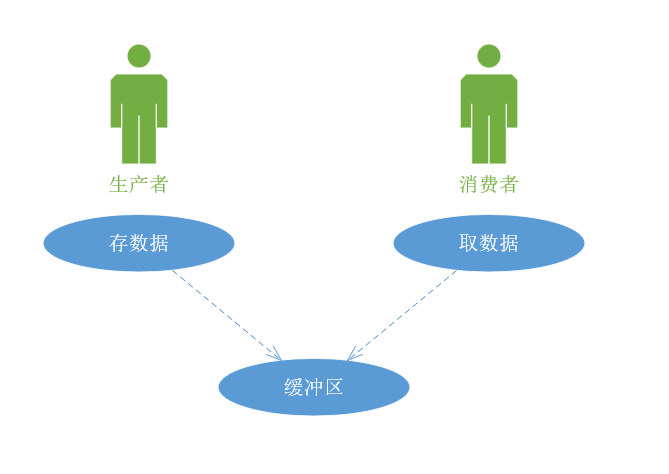
1. init：当系统找不到序列化的虚拟磁盘文件，就会进行初始化操作。初始化包括初始化磁盘数据，FCB集合数据，根目录数据，FAT表，内存数据，用户集合数据。
2. loadData：当存储路径文件存在时，系统通过字节流加载磁盘数据，同时将磁盘中的用户集合数据、根目录、当前用户、当前目录、FAT表读入内存。
3. saveData：当用户退出系统时，系统自动将内存数据持久化到磁盘中，再将磁盘存储到虚拟磁盘文件中。
4. freeFile：清除文件占有的磁盘空间，修改FAT表，递归修改父目录文件大小，清空索引结点
5. writeToDisk：将用户输入的内容存入磁盘持久化。过程包括：判断是否有足够的磁盘空间，遍历FAT表寻找空闲块，将文件内容分块存储进去，随后返回第一个块号
6. find\_empty：用于存储内容时寻找空闲块块号返回块号。

2.3.1 类图

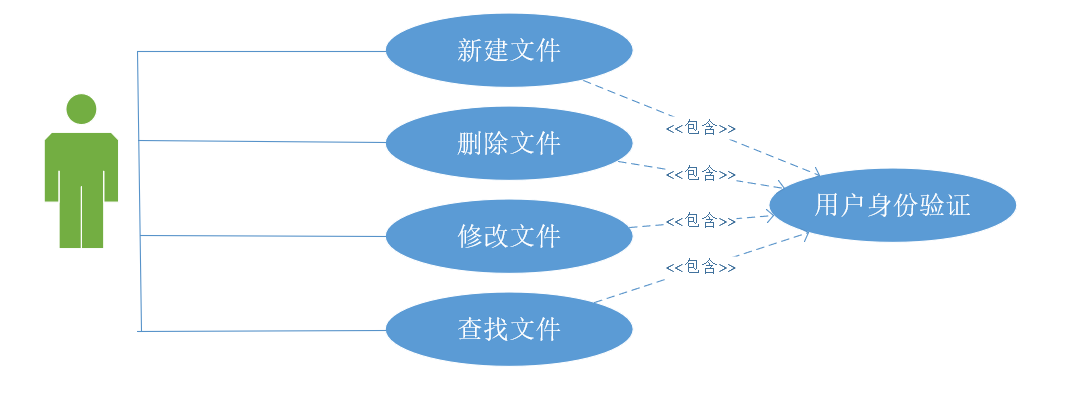


2.3.2 用例图

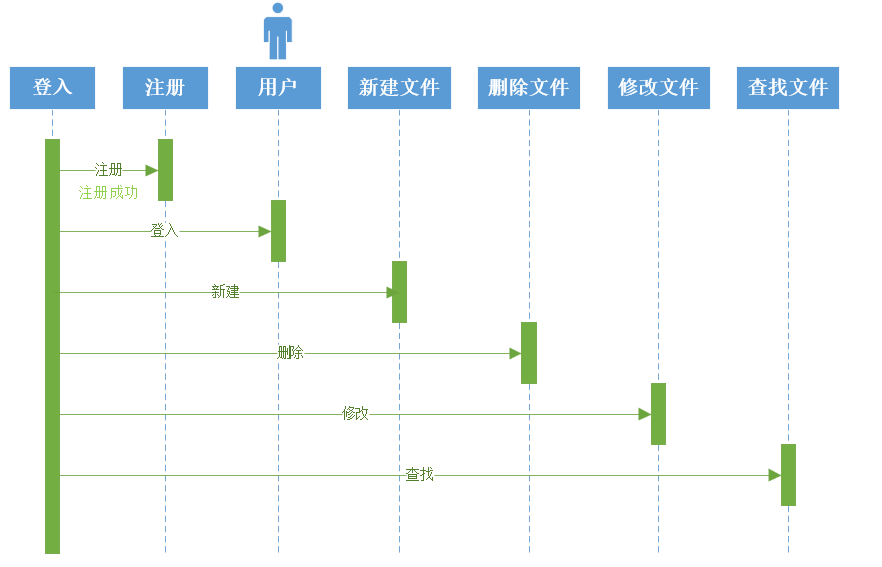
**生产者和消费者**



**文件系统**



2.3.3 顺序图



# 数据结构

# **生产者和消费者的问题**

* + 1. 缓冲区

存储列表

private final List<integer>list;

缓冲区大小

private final int MAX\_SIZE;

* + 1. 线程类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Producer类 | | Consumer类 | |
| 共享缓冲区 | List | int | mlists | int |
| 缓冲区最大容量 | MAX\_SIZE | Int | - | - |
| 构造进程 | - | Producer() | - | Consumer() |
| 模拟进程 | void | Run | void | run |

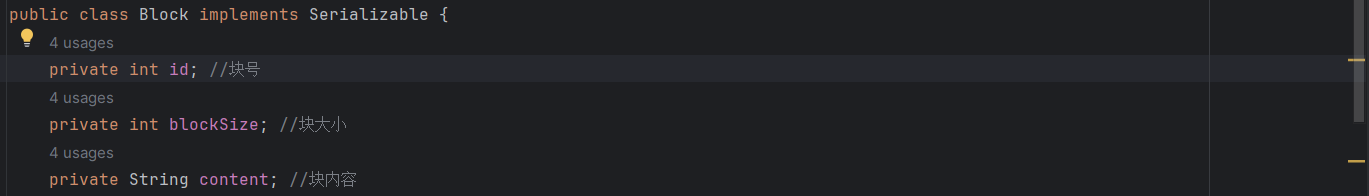
* + 1. 控制流函数

使用synchronized关键字只有一个线程访问，给共享缓冲区加锁，控制线程对缓冲区的互斥访问。生产者可以通过add()对共享缓冲区的数据进行添加，消费则则对缓冲区取出数据。同时通过wait()/notifyAll()来保证多个生产者和消费者在同一个缓冲区，只有一个数据流对缓冲区进行添加和取出。

# **文件系统**

# Block

磁盘整体内存为2MB，共分为256个磁盘块，每一物理块可存储8B的数据。



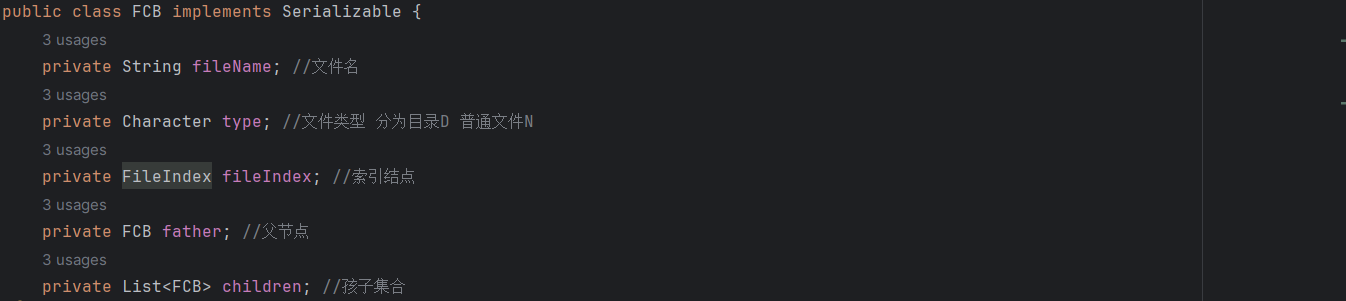
# Disk

磁盘有专门空间用于存储FCB集合、用户集合、文件分配表FAT。

# 树形结构

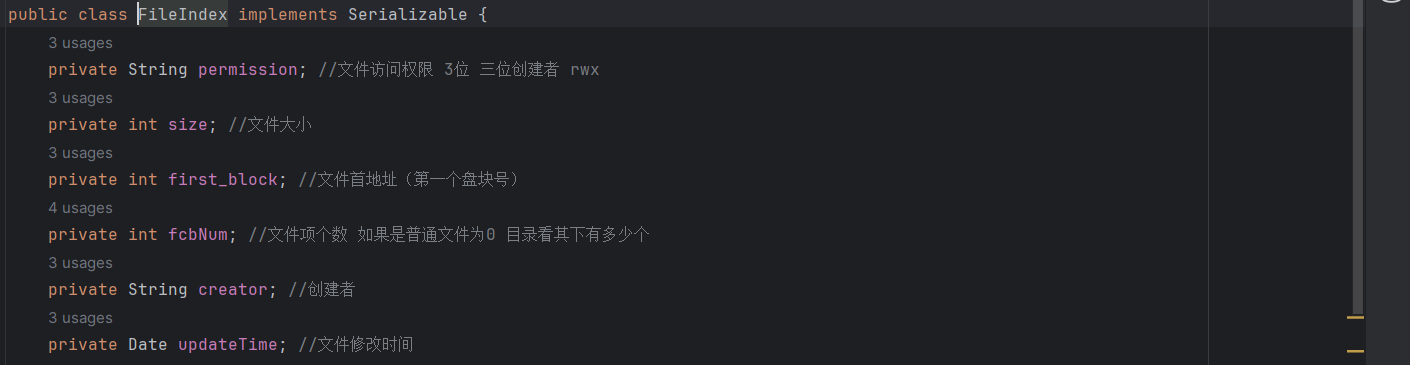
系统采用树形结构进行目录组织，根目录是所有目录及文件的根，根目录下每个用户有自己的同名文件夹，用户也可以在根目录下进行创建文件，与其他用户实现文件共享。

# FCB



每个文件（普通文件和目录）都有文件控制块FCB，凭借FCB才可以形成树形文件目录。本系统的FCB除了文件名文件类型以外的大部分内容都由索引结点进行封装，加速读取速率。每个FCB有父节点和孩子集合，采用双亲孩子表示法进行树结构的组织。

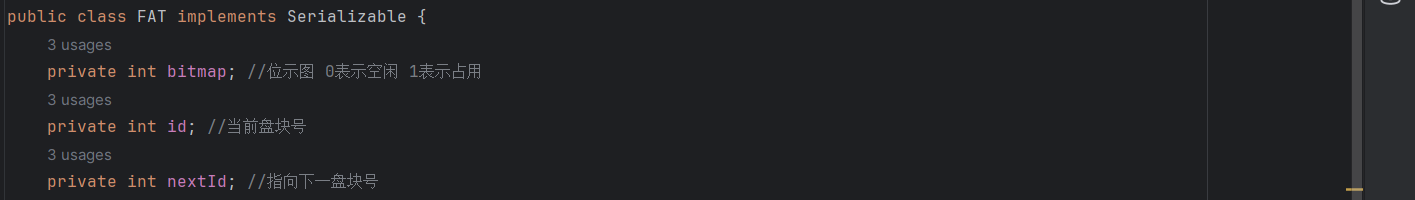
# FileIndex



文件的索引结点存放文件权限，文件大小，存放在的物理块首地址，文件项数（如果是目录才有效，文件为0）以及文件创建者和修改时间。

文件权限仿造linux，同时做了简化，这里用户可以设置文件对自己（创建者）的权限，以及其他用户的权限。权限包括读（r=4）写（w=2）和执行（x=1）。

# FAT数据结构



FAT是文件分配表，用于记录文件的分配情况，同时为节省空间，直接将位示图合并进FAT表中，FAT表只有256项，与磁盘物理块一一对应，可显示磁盘每一块的分配情况。

# **关键技术**

# 生产者消费则问题

# sychronized关键字

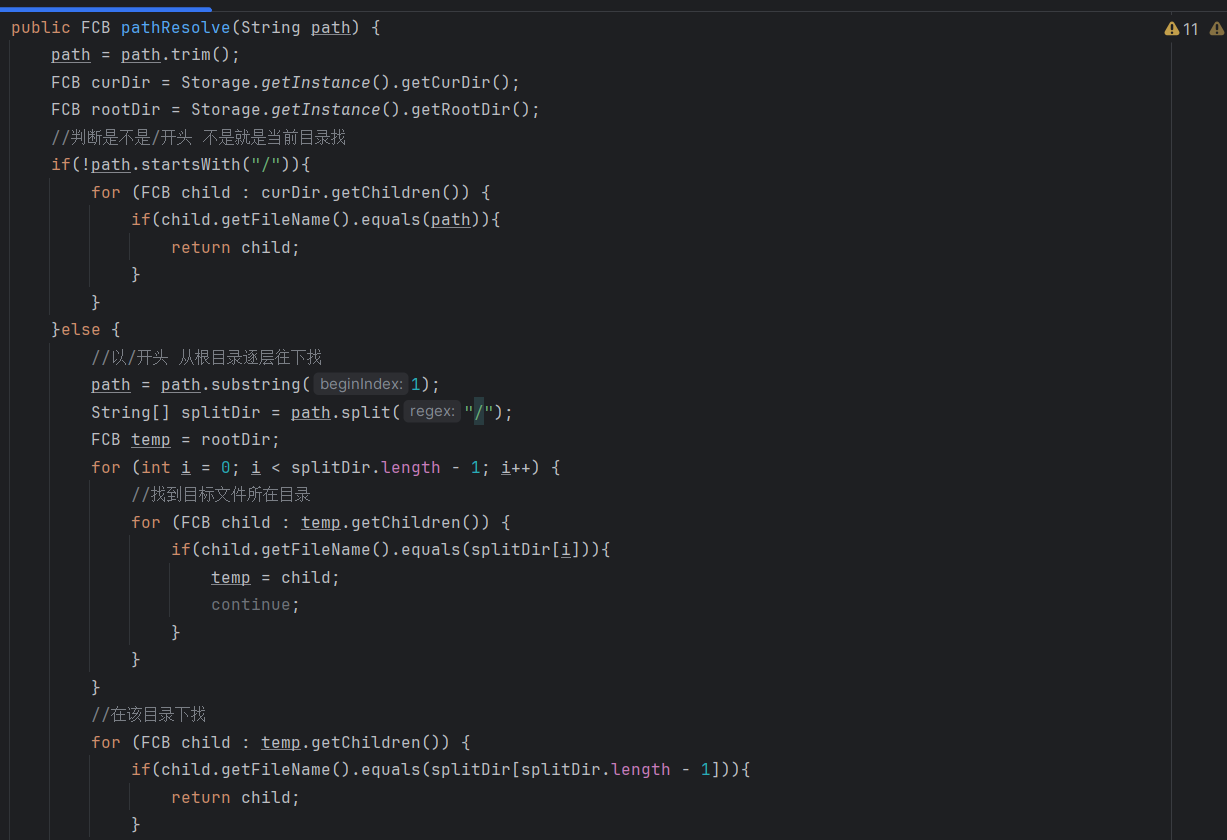
多线程同时访问一个资源的时候就会导致程序运行错误的问题。所以当线程互斥访问缓冲区，并是单一的访问。生产者访问的时候消费者就等待。消费者访问的时候就消费者等待。我们可以使用sychonized对对象进行加锁，获得了锁的线程才能进行相关的操作。只有有锁的线程才能操作共享缓冲区。

# Wait()/notifyall()方法

共享缓冲区有个上下限容量，生产者和消费者必须配合访问，生产者不可以向满的缓冲池中存入数据，消费者不可以从空的缓冲区中取出数据。当缓冲区达到上限容量时，生产者无法向缓冲区生产数据，就会陷入死循环。由于synchronized存在所以线程无法操作缓冲区。因此，需使用wait()方法来让进程进入等待状态，当其他线程操作完毕后，使用notifyAll()方法唤醒等待进程。消费者也是一样，当消费者消费光时，需要进入等待状态，等生产者线程生产数据后，才能唤醒消费者线程继续消费数据。

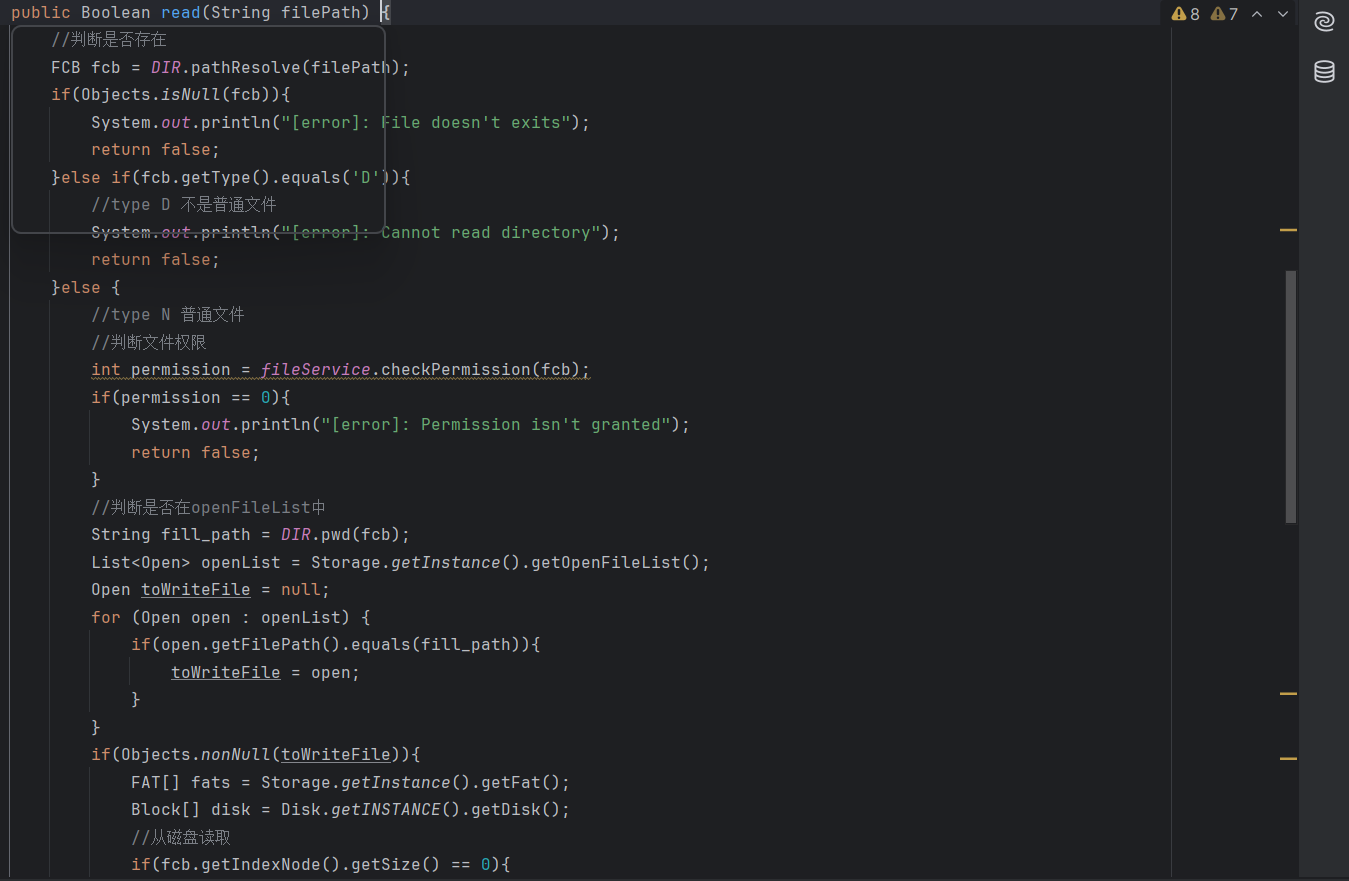
# 文件系统

# Path resolve



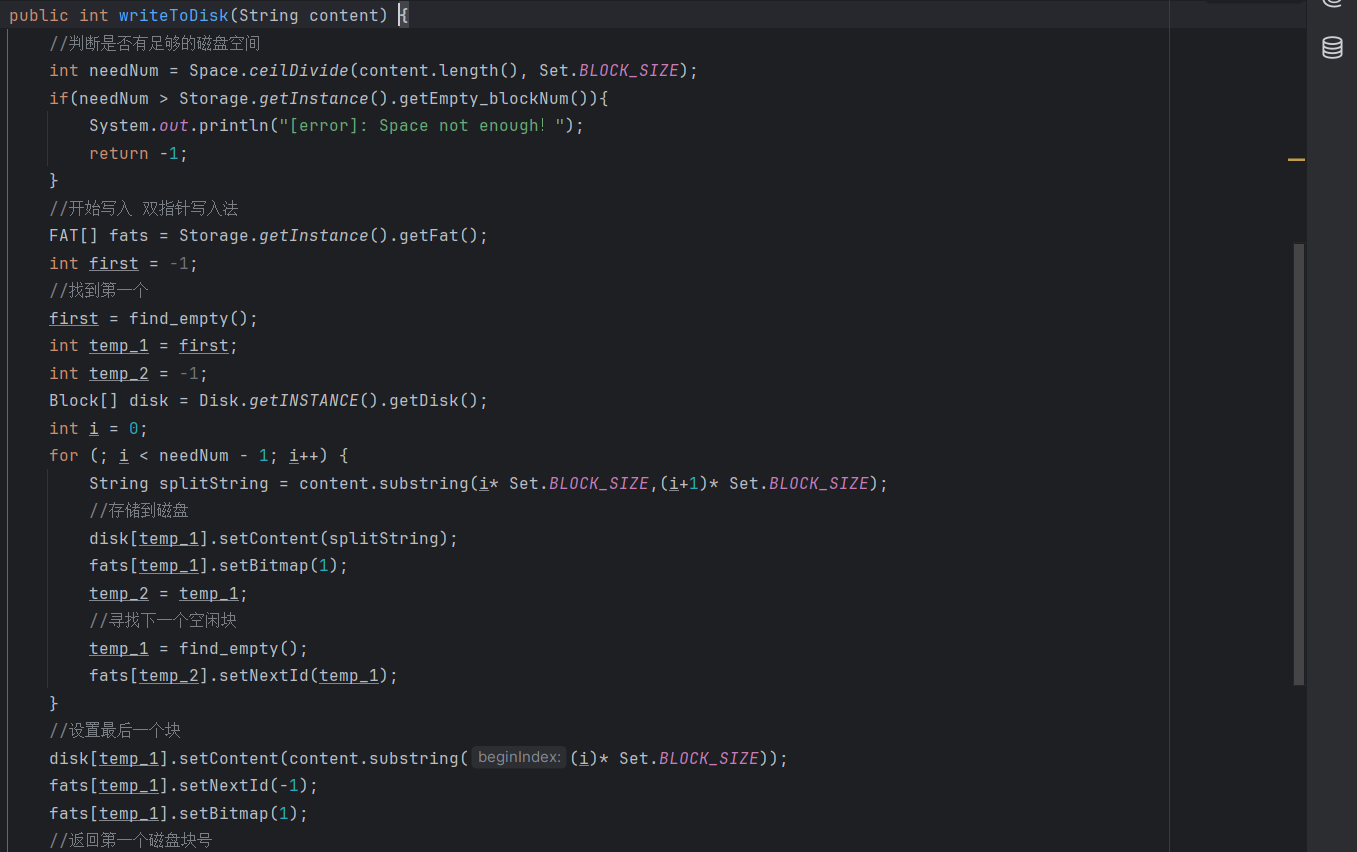
实现跨目录文件目录操作的核心。判断在根目录还是当前目录下寻找。

# Read



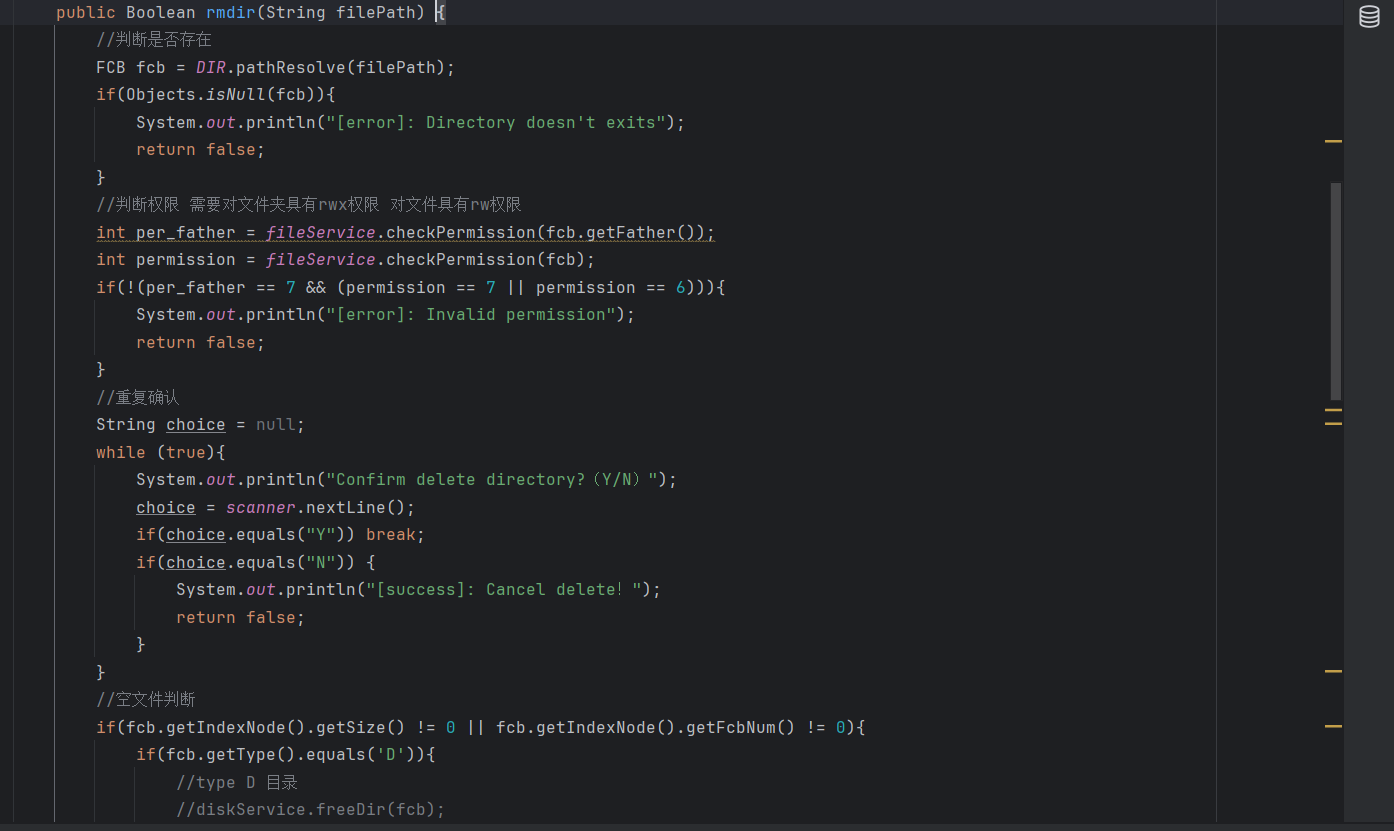
解析文件路径，判断是否是普通文件接着判断权限，找到索引结点按照FAT表读取磁盘内容

# Write to disk



判断磁盘是否够空间然后开始写入最后存入磁盘

# Delete



**删除目录**，判断是否为空后判断权限，最后重复确认，删除后要freeFile来清空磁盘。

**删除文件**的操作也是和删除目录一样类似的。

# **初始化**

//初始化磁盘

for (int i = 0; i < Constants.BLOCK\_COUNT; i++) {

disk[i] = new Block();

disk[i].setId(i);

disk[i].setBlockSize(Constants.BLOCK\_SIZE);

disk[i].setContent(null);

}

newDisk.setDisk(disk);

//初始化FCB集合

List<FCB> fcbList = new ArrayList<>();

newDisk.setFcbList(fcbList);

//初始化根目录

FileIndex fileIndex = new FileIndex("rwxrwx",0,-1,0,null,new Date());

FCB rootDir = new FCB("rootDir",'D', fileIndex,null,new LinkedList<>());

fcbList.add(rootDir);

//初始化FAT表

FAT[] fats = new FAT[Constants.BLOCK\_COUNT];

for (int i = 0; i < Constants.BLOCK\_COUNT; i++) {

fats[i] = new FAT();

fats[i].setId(i);

fats[i].setBitmap(0);

fats[i].setNextId(-1);

}

newDisk.setFat(fats);

//初始化内存

Storage storage = Storage.getInstance();

//用户集合

Map<String, User> userMap = new HashMap<>();

newDisk.setUserMap(userMap);

//初始化内存

storage.setUserMap(userMap);

storage.setCurDir(rootDir);

storage.setCurUser(null);

storage.setRootDir(rootDir);

storage.setFat(fats);

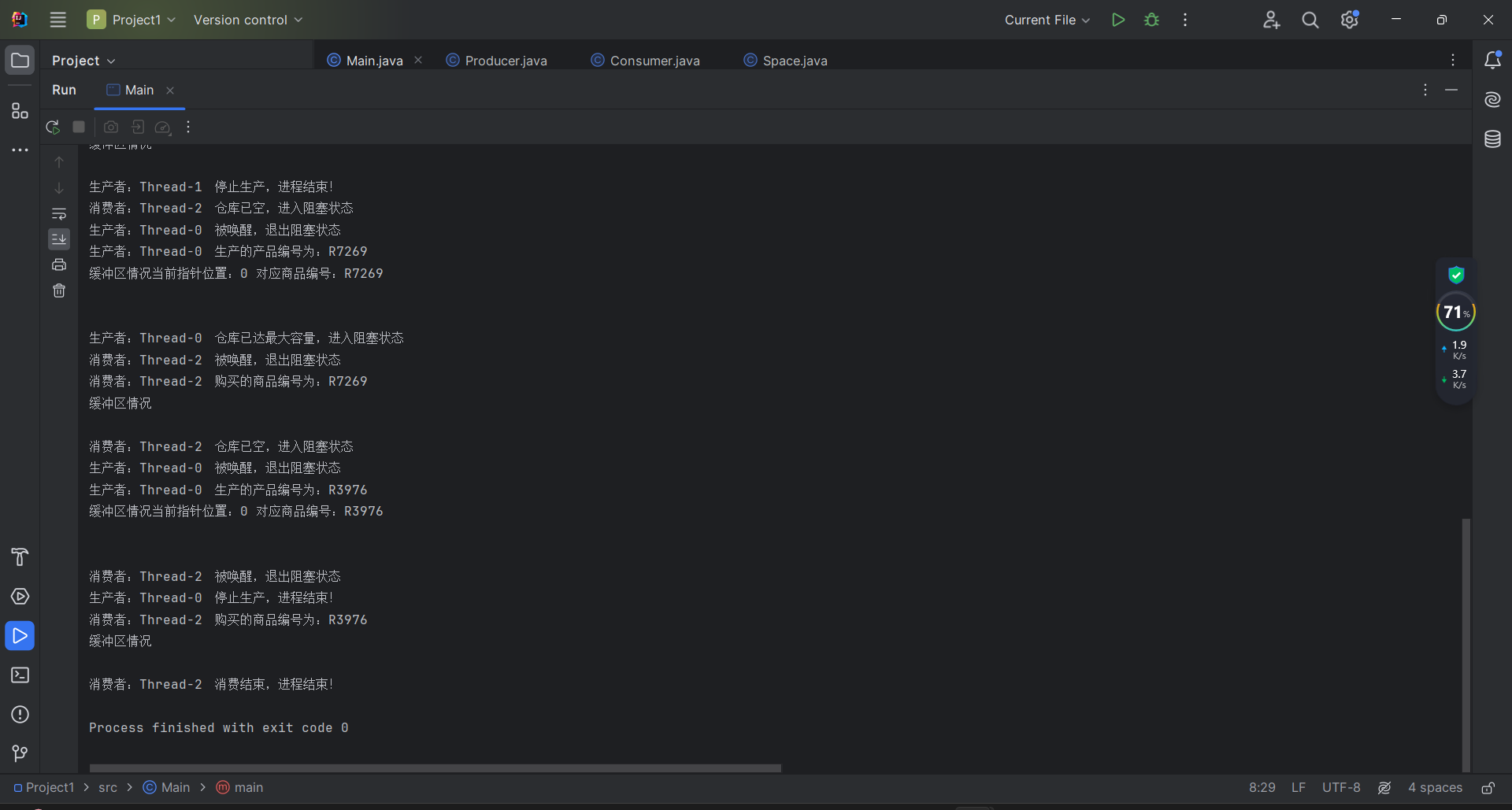
ArrayList<OpenFile> openFiles = new ArrayList<>();

storage.setOpenFileList(openFiles);

storage.setEmpty\_blockNum(Constants.BLOCK\_COUNT);

System.out.println("[success]Init success");

# 运行结果



生产者和消费者

## 

文件系统

## 10.1 运行环境

本次实验运行环境的是使用了AMD Ryzen 5的处理器以及8GB的RAM。系统类型是64 位操作系统

## 10.2 运行结果

用户登入的系统的主界面，用户可以进行注册，登入以及退出操作。注册操作需要用户输入账号和密码后才能登入。登入操作需要用户提供正确的账号和密码才能登入。



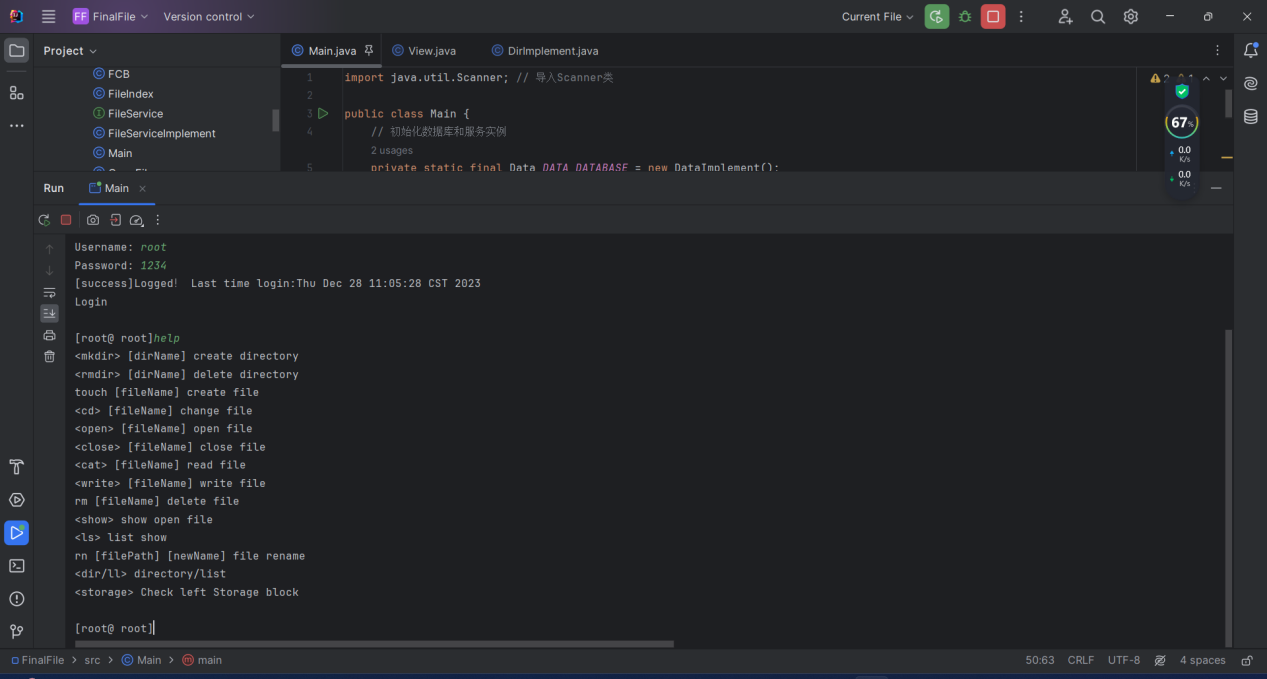
10-1.系统主界面

用户注册主界面需要用户输入账号和密码，确保账号的密码后才能登入。要是用户没注册就需要用户先注册后再登入。



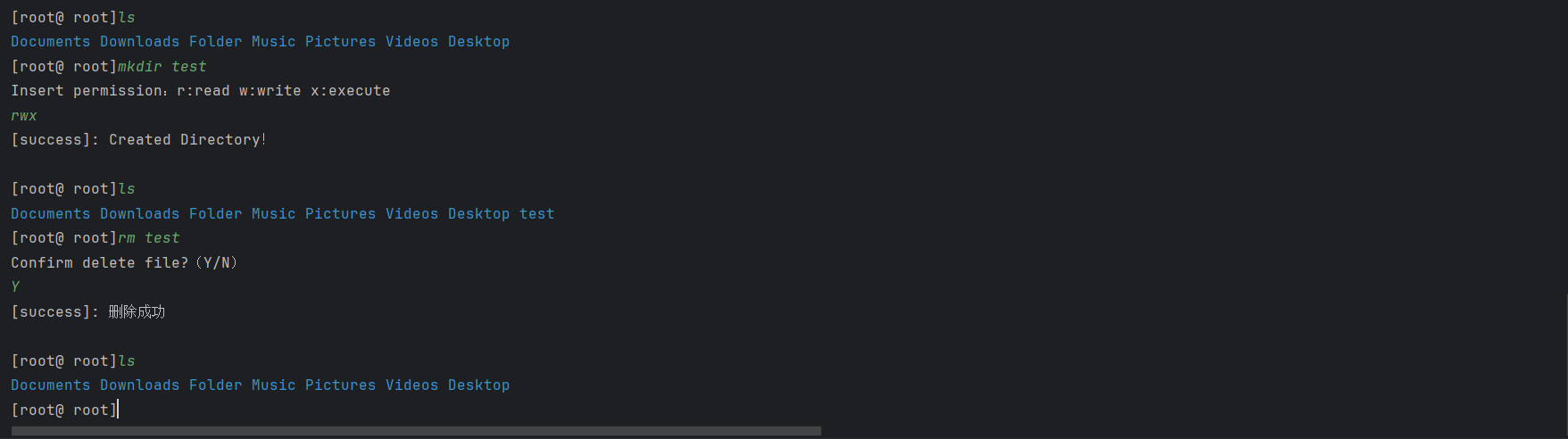
10-2.用户注册主界面

用户可以输入help来获取相关的帮助



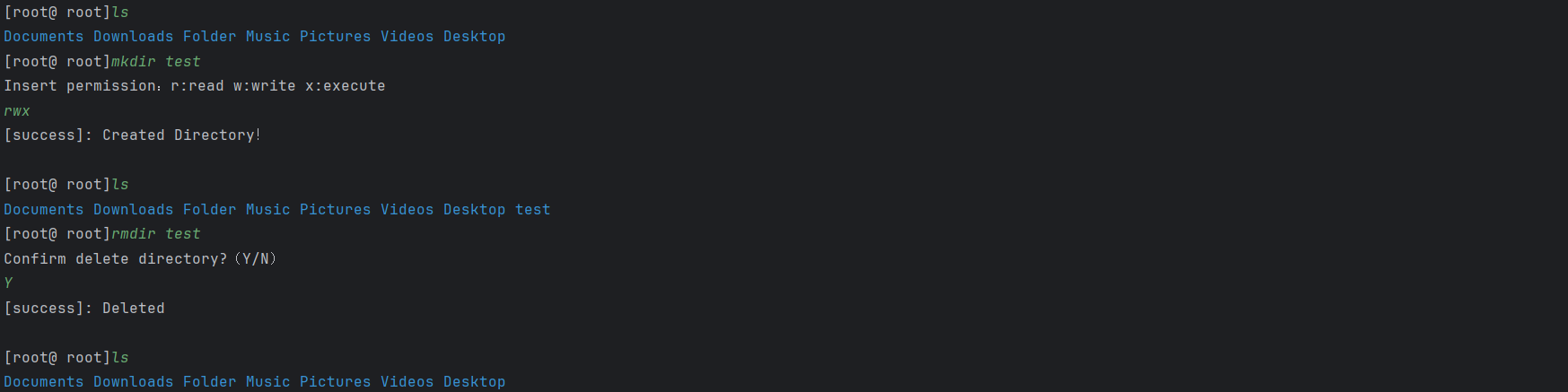
10-3.对应的操作

用户可以使用mkdir[filename]来创建新的目录



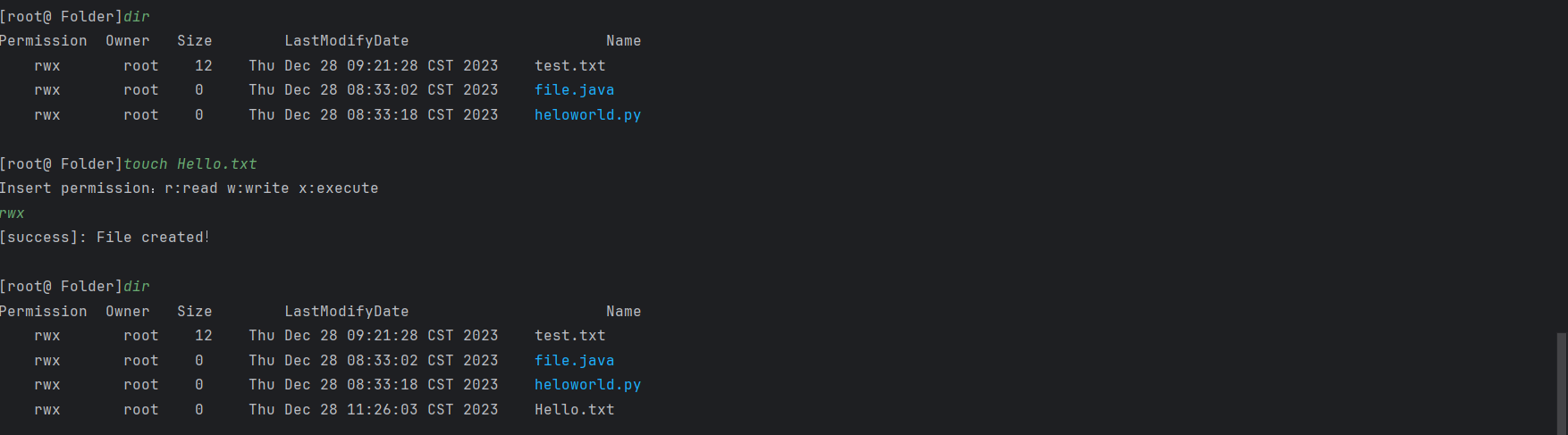
10-4.mkdir操作

用户可以使用rmdir [filename]来进行对目录的删除



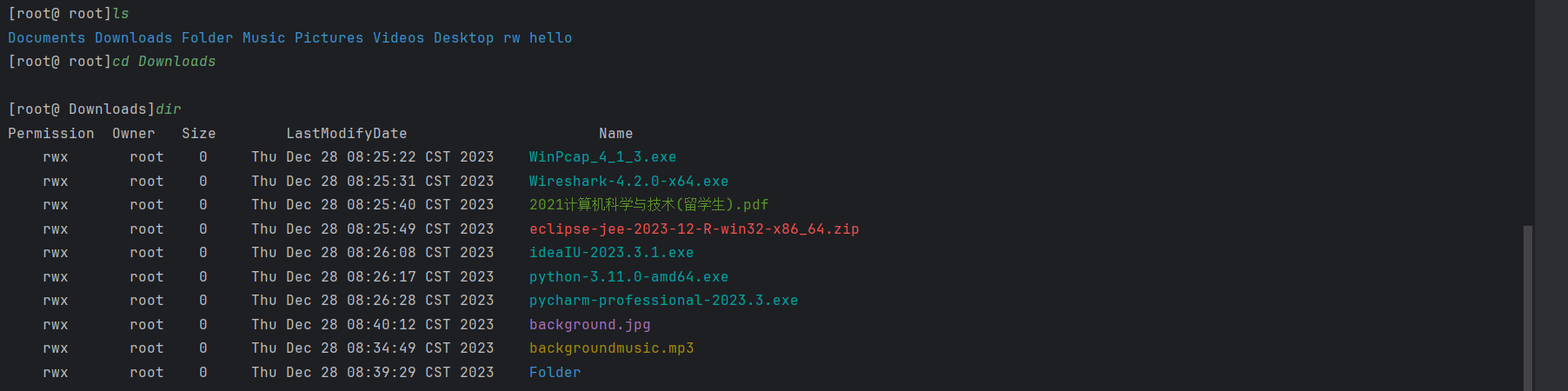
10-5.rmdir操作

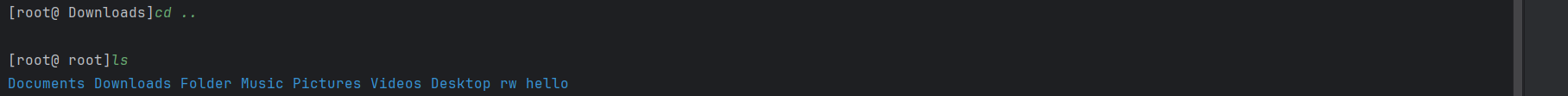
用户可以使用touch来创建文件



10-6.touch操作

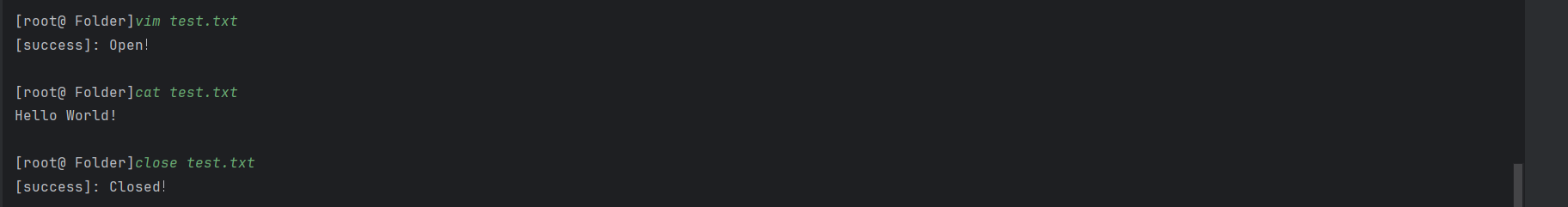
用户输入cd directorypath 可以进入相应的操作，也可以使用cd ..和cd .来返回上一级目录





10-7.cd操作

用户可以可以打开文件后使用read操作来读取文件内容然后再关闭文件



10-8.open,read,close操作

用户可以使用write操作来实现输入，要是里面已经有内容了会提示用户是否要继续写还是覆盖之前的内容。



10-9.write操作

# 调试和改进

遇到了要删除文件但是删除不了一直报错，然后就上网找了原来是permission那边写错了没给到用户权限去删除从而导致删除不了。还有一个问题是要做可视化界面的尝试使用JavaSwing和JavaFx但还是弄不了就没弄可视化界面了。我觉得是可以做可视化界面的，毕竟这是面向用户的操作，所以也要有可视化界面。这里也可以添加管理员操作，我这里知识用户对用户的操作罢了。

# 心得和结论

## 结论和体会

完成了生产者和消费者共享缓冲的问题，使用了wait()/notifyAll()。在多用户管理系统使用了Block以及FAT和FCB来管理用户的资料。优点Disk存储都存在一个地方。缺点的话没可视化界面，以及没连接数据库，没有管理员。虽然没连接数据库但是是使用的系统设置userMap<String username , User user>充当用户数据库，在磁盘上分配以磁盘块专门存放用户数据，系统运行之后就读入内存。

## 进一步改进方向

还可以添可视化界面，增强系统对文件的保护比如使用加密的方法。

# 主要参考文献

1. 北京市政府鼓励商场开网上商店, <http://news.iresearch.cn/0468/20090513/94180.shtml> ，访问日期：2018年3月6日.
2. Garey M R, Johnson D S.Computers and Intractability: A Guide to the Theory of Np-Completensess[M]. New York: WH Freeman & Co, 1979.
3. 樊学豹、程素萍、秦鹏渊，白云铁矿主采场西端岩石运输系统方案设计，包钢科技，第33卷第4期，2007年8月。
4. Desrosiers J, Soumis F, Desrochers N, Routing with time windows by column generation[J], 1984, 14:545-565.
5. https://blog.csdn.net/qq\_40989769/article/details/125223591超专业解析|linux文件系统的底层架构及其工作原理
6. https://www.cnblogs.com/theseventhson/p/15622853.html[linux源码解读（三）：文件系统——inode](https://www.cnblogs.com/theseventhson/p/15622853.html)
7. https://www.cnblogs.com/theseventhson/p/15643658.html[linux源码解读（五）：文件系统——文件和目录的操作](https://www.cnblogs.com/theseventhson/p/15643658.html)
8. 计算机操作系统， 汤小丹等 ，西安电子科技大学出版社
9. 操作系统实验指导书，傅秀芬，广东工业大学（自编）
10. 计算机操作系统教程 ( 第二版 )， 张尧学、 史美林，清华大学出版社
11. 现代操作系统，A.S.Tanenbaum 著，陈向群等译机械工业出版社

**考核成绩评定表**

|  |  |
| --- | --- |
| 指导教师考核成绩 |  |
| 答辩成绩 |  |
| 总成绩 |  |

签字：

**年 月 日**