## 

**《操作系统课程设计》**

**（2019/2020学年第一学期第17周）**

**指导教师：田秋红**

**郭奕亿**

**班级：17计科2班**

**学号：2017331200181**

**姓名：陈焕**

目录

[1. 引言 4](#_Toc9473)

[1.1任务要求 4](#_Toc5605)

[1.2 选题 4](#_Toc17534)

[用JAVA语言模仿“生产者—消费者问题 4](#_Toc18323)

[文件系统 4](#_Toc6561)

[2. 需求分析与设计 5](#_Toc14787)

[2.1 需求分析 5](#_Toc24042)

[2.1.1 系统的用户群体 5](#_Toc19834)

[2.1.2产品的功能性需求 5](#_Toc28080)

[1.Web端 6](#_Toc30610)

[2.1.3产品的非功能性需求 8](#_Toc13518)

[2.1.3.1软件总体要求 8](#_Toc23836)

[2.1.3.2性能要求 9](#_Toc12827)

[2.2系统框架和流程 10](#_Toc16281)

[2.2.1 系统结构图 10](#_Toc20261)

[2.2.2流程图 11](#_Toc8222)

[3. 数据结构 12](#_Toc30537)

[3. 1 数据表 12](#_Toc30898)

[3.2 文件节点 13](#_Toc28713)

[4. 关键技术 13](#_Toc9561)

[4.1 DFS算法 13](#_Toc18174)

[4.1.1算法思想 13](#_Toc8868)

[4.1.2 DFS的运用 16](#_Toc353)

[4.2 分治思想 16](#_Toc13775)

[4.2.1 算法思想 16](#_Toc30080)

[4.2.2分治思想的运用 16](#_Toc29356)

[4.3乐观锁机制 17](#_Toc3291)

[4.3.1乐观锁思想 17](#_Toc30180)

[4.3.2 乐观锁的运用 18](#_Toc3577)

[4.3 Markdown的文本编辑格式 19](#_Toc7773)

[4.3.1 Editor.md 19](#_Toc11447)

[4.3.2 Editor.md的嵌入使用 19](#_Toc9931)

[5. 运行结果 20](#_Toc24093)

[5.1 运行环境 20](#_Toc18603)

[5.2 服务模式 20](#_Toc5093)

[5.3 运行结果 21](#_Toc7230)

[5.3.1 登陆界面 21](#_Toc25868)

[5.3.2 主界面 21](#_Toc17730)

[5.3.3 文件查看界面 22](#_Toc5166)

[5.3.4 文件修改界面 22](#_Toc14962)

[5.3.5文件分享界面 23](#_Toc10618)

[5.4实验结果分析 24](#_Toc9756)

[5.4.1 文件的浏览 24](#_Toc17785)

[5.4.2 文件的查看与编辑 24](#_Toc4170)

[5.4.2 文件的分享 24](#_Toc15117)

[6. 调试和改进 24](#_Toc18258)

[7. 心得和结论 25](#_Toc12871)

[7.1 结论和体会 25](#_Toc25035)

[7.2 进一步改进方向 25](#_Toc25350)

## 引言

我们的服务器每天都在不知疲倦的运行着，也会产生很多垃圾，然而我们有时候毫不知情，有时候也对这些黑白像素组成的命令行感到头疼，针对目前服务器文件访问形式单一，结构信息抽象难以分析的现状，QAQ团队准备研制一款轻量级的多用户文件管理系统，具有可视化Web界面，脱离设备的限制，随时随地打开浏览器就能访问、管理服务器中的QAQ文件系统，致力于降低文件管理乃至服务器管理的难度。

### 1.1任务要求

1. 初始数据的获取：创建file队列采用BFS算法获取每一层的数据，并保存在TreeNode中
2. 遍历算法：采用DFS算法的方式，能够快速的找到某个节点
3. 文件的增删改、打开关闭：将文件预读入内存修改完覆盖即可
4. 图形界面的设计：目前有三种备选方案：1.JavaSwing， 2.Vue.js(即Web项目)，3.shell-bash（命令行）前两者为图形化界面，第三个为个人偏爱形式，最终会根据进度选择一种实现。
5. 多用户访问，根据进度确定是否实现用户权限的控制。
6. 文件保护，为修改中的文件上锁，表示不可同时写操作、根据用户权限确认对某文件是否拥有修改权力

### 1.2 选题

##### 用JAVA语言模仿“生产者—消费者问题

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。

##### 文件系统

设计一个多用户文件系统，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。实现以此为基础加入自己设计功能的小型文件系统 。

## 需求分析与设计

### 需求分析

本系统是一个基于Springboot微服务框架的轻量级远程文件管理系统，专门针对市面上没有可视化界面的服务器，如Centos、Debian等，试图针对此类管理抽象困难的问题提出一种解决方案。

#### 2.1.1 系统的用户群体

QAQ文件管理系统，拥有Web端后台管理，为普通管理员和超级管理员提供对应服务。普通管理员为系统的主要使用者，也是服务器的使用者，超级管理员为系统的管理者，也是服务器的运维、管理人员。系统的用户群体图如下表 2-1 所示。

表2-1用户群体表

|  |  |
| --- | --- |
| 角色名称 | 描述 |
| 普通管理员 | 为系统的主要用户，可在WEB进行操作，可管理非root权限文件 |
| 超级管理员 | 为运维人员，即管理并操作后台管理系统的管理员，可管理所有文件 |

#### 2.1.2产品的功能性需求

QAQ文件管理系统提供了Web端服务，通过数据共享紧密联系，系统功能图见图 2-1。

QAQ文件管理系统

Web端

普通管理员

超级管理员

个人中心

文件浏览

非Root文件修改

非Root文件删除

新增文件

用户管理

文件浏览

文件修改

文件删除

新增文件

图2-1系统功能图

##### 1.Web端

Web端的使用者为后台管理员。Web端为系统管理员提供了用户管理、文件管理和两个功能模块。以下是功能模块的具体介绍。



用户管理

文件管理

文件查看

文件修改

新增文件

删除文件

<<include>>

<<include>>

<<include>>

<<include>>

用户权限管理

<<include>>

文件分享

<<include>>

* 1. Root管理员用例图



* 1. Root管理员序列图



文件管理

文件查看

<<include>>

非Root文件修改

新增文件

非root文件删除文件

<<include>>

<<include>>

<<include>>

文件分享

<<include>>

* 1. 普通管理员用例图



* 1. 普通管理员序列图

①文件管理

系统管理员根据等级、权限的不同，可进行不同的操作，例如管理管理员账号、设置权限等；管理员对由本系统提供的模板图片进行统一管理；管理员可管理付费服务内容。

②用户管理

超级管理员可查询系统用户的基本信息，用户的操作日志，并可查询用户的消费、充值信息。并对用户的部分文件进行管理。

#### 2.1.3产品的非功能性需求

##### 2.1.3.1软件总体要求

1. 可靠性原则

系统设计能有效的避免单点失败，在设备的选择和关键设备的互联时，应提供充分的冗余备份，一方面最大限度地减少故障的可能性，另一方面要保证系统能在最短时间内修复。

1. 可使用性

对格式和数类型限制的数据，进行验证，包括客户端验证和服务器验证，并采用错误提醒机制，提示用户输入正确数据和正确的操作系统。

1. 可管理性原则

整个系统的设备应易于管理，易于维护，操作简单，易学，易用，便于进行系统配置，在设备、安全性、数据流量、性能等方面得到很好的监视和控制，并可以进行远程管理和故障诊断。

1. 可扩展性原则

系统应具有高扩展性，能够方便集成不同系统的数据，并预留接口以适应未来拓展需要。

1. 接口良好性原则

系统能够提供比较良好的接口，便于系统的维护与修改，同时可比较方便的进行业务流程的修改。

1. 安全性原则

严格设置了安全级别和权限，只有合法用户才能登录使用系统，对每个用户都有权限设置、对登录名、密码、以及用户重要信息进行加密，保证账号信息安全。对于合作企业，我们实施严谨的身份认证体系。

1. 规范性原则

系统设计所采用的技术和设备应符合国际标准、国家标准和业界标准，具备为系统的扩展升级以及与其他系统的互联的良好基础。

1. 界面设计原则

系统界面设计遵循简易性、人性化原则，使用户看起来轻松自然，界面逻辑结构简单清晰，让用户便于使用、便于了解、并减少用户发生错误选择的可能性。界面更加简洁清晰，方便用户的使用。

##### 2.1.3.2性能要求

1. 数据精确度

在精度需求上，根据实际需要，数据在输入、输出及传输的过程中要满足各种精度的需求根据关键字精度的不同。如：查找可分为精确查找和模糊查找，精确查找可精确匹配与输入完全一致的查询结果，模糊查找，只要满足与输入的关键字相匹配的输入即输出，可供查找。

1. 时间特性

系统响应时间应在人的感觉和视觉范围内(<1s)，系统响应时间足够迅速(<5s)，能够满足用户要求。

1. 系统界面友好

界面美观大方，使用简便，支持较高的灵活性操作。

1. 实时性强

数据查询、统计、更新做到及时、准确。查询统计不影响业务系统同时运行。

1. 可维护性

本系统采用的 B/S 模式，结构清晰，便于维护人员进行维护。

### 2.2系统框架和流程

#### 2.2.1 系统结构图

系统为多个用户提供一个Web端的文件管理的平台，其运行原理如图2-6系统框架图所示：

用 户（Web）

文件信息缓存

Mysql用户数据库

身份验证接口

文件查看接口

文件修改接口



新增文件接口

文件删除接口



权限验证

文件写/删（锁）

定时扫描更新

文件修改

Druid连接池

更新

磁盘

文件分享接口

图2.6 系统框架图

#### 2.2.2流程图

在图2.7 用户登陆浏览流程图和图2-7 用户请求处理流程图中，在数据库权限验证后，访问缓存、磁盘是同时进行的，其原理类似于快慢表，当缓存找不到接收内存中的内容。

用户登陆

信息是否正确?

F

数据获取

用户输入账号密码

访问数据库

T

用户输入访问路径

访问磁盘

界面显示

访问缓存

Web数据处理

登陆成功

图2.7 用户登陆浏览流程图

端口监听

缓存访问线程

控制器接收请求

数据库访问

缓存访问线程

用户线程存在？

请求处理

新建用户线程

结果分装、返回

T

F

图2-7 用户请求处理流程图

## 数据结构

### 3. 1 数据表

**表3-1管理员信息表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 伦理名 | 物理名 | 属性 | 主键 | 外键 |
| 用户名 | Username | Varchar(255),not null | Y |  |
| 用户密码 | Password | Varchar(255),not null |  |  |
| 用户权限 | rank | Int(1),not null |  |  |

User表配合User类用于权限管理功能的实现。

### 3.2 文件节点

|  |
| --- |
| public class FileInfo {  //文件名  private String fileName;  //绝对路径  private String absolutePath;  //父级路径  private String parentPath;  //修改日期  private long lastModified;  //文件类型（是否为文件夹、目录）  private boolean isDir;  ｝ |
| 该数据结构包含两部分：1.文件基本信息，2.子节点信息。  用于文件树的构造，文件基本信息的获取，属于共享资源（不可同时写、保存）。 |

## 关键技术

### DFS算法

深度优先搜索属于图算法的一种，英文缩写为DFS即Depth First Search.其过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个[节点](https://baike.baidu.com/item/%E8%8A%82%E7%82%B9" \t "_blank)只能访问一次.

#### 4.1.1算法思想

深度优先搜索在搜索过程中访问某个顶点后，需要递归地访问此顶点的所有未访问过的相邻顶点。

初始条件下所有节点为白色，选择一个作为起始顶点，按照如下步骤遍历：

a. 选择起始顶点涂成灰色，表示还未访问

b. 从该顶点的邻接顶点中选择一个，继续这个过程（即再寻找邻接结点的邻接结点），一直深入下去，直到一个顶点没有邻接结点了，涂黑它，表示访问过了

c. 回溯到这个涂黑顶点的上一层顶点，再找这个上一层顶点的其余邻接结点，继续如上操作，如果所有邻接结点往下都访问过了，就把自己涂黑，再回溯到更上一层。

d. 上一层继续做如上操作，知道所有顶点都访问过。

用图可以更清楚的表达这个过程：

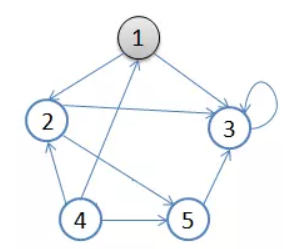


图4-7 DFS思想图1

1.初始状态，从顶点1开始

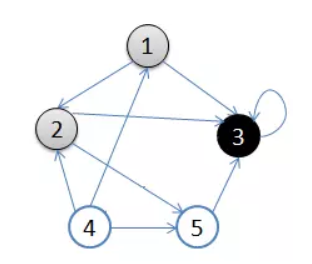


图4-8 DFS思想图2

2.依次访问过顶点1,2,3后，终止于顶点3

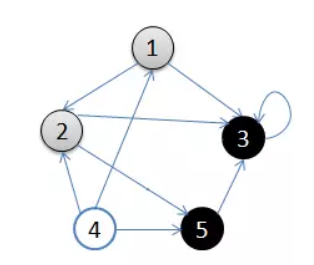


图4-9 DFS思想图3

3.从顶点3回溯到顶点2，继续访问顶点5，并且终止于顶点5

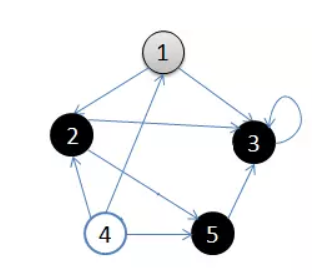


图4-10 DFS思想图4

4.从顶点5回溯到顶点2，并且终止于顶点2

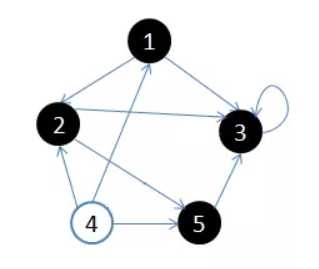


图4-11 DFS思想图5

5.从顶点2回溯到顶点1，并终止于顶点1

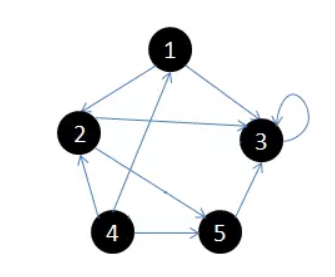


图4-12 DFS思想图6

6.从顶点4开始访问，并终止于顶点4

#### 4.1.2 DFS的运用

系统初始化的在时候需要读取根目录及子节点，同时构造文件树，在读取的时候采用图论中的深度优先算法。

由于系统文件的结构是树状分布，此时用深度优先算法（递归实现）可以很的构建出文件树。

### 分治思想

#### 算法思想

当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。如果这些子问题还较大，难以解决，可以再把它们分成几个更小的子问题，以此类推，直至可以直接求出解为止。这就是[分治策略](https://baike.baidu.com/item/%E5%88%86%E6%B2%BB%E7%AD%96%E7%95%A5" \t "_blank)的基本思想。

#### 4.2.2分治思想的运用

1.在构建文件结构树的时候采用递归的形式，将文件树的构造分解成构造子节点的过程，如此一来一方面大大减少代码量，另一方面使得结构更加清晰，拥有较强的可维护性。下列代码即构造FileInfo节点，files.add(new FileInfo(new File(absolutePath + "\\" + li)));递归调用构造函数，使得文件结构体系得以快速成型。

|  |
| --- |
| public FileInfo(File file) {  files = new ArrayList<>( );  fileName = file.getName( );  absolutePath = file.getAbsolutePath( );  parentPath = file.getParent( );  date = new Date(file.lastModified( ));  String[] list = file.list( );  if (list != null) {  for (String li : list  ) {  // System.out.println(li);  // System.out.println(new File(absolutePath + "\\" + li));  files.add(new FileInfo(new File(absolutePath + "\\" + li)));  }  } else {  files = null;  }  } |

1. 在目录删除的时候，需要分治思想，先删除目录下的文件，再删除目录

|  |
| --- |
| /\*\*  \* 递归删除目录下的文件以及目录  \*  \* @param file  \* @return  \*/  static boolean forDelFile(File file) {  if (!file.exists( )) {  return false;  }  if (file.isDirectory( )) {  File[] files = file.listFiles( );  for (File f : files) {  forDelFile(f);  }  }  return file.delete( );  } |

### 4.3乐观锁机制

#### 4.3.1乐观锁思想

  每次获取数据的时候，都不会担心数据被修改，所以每次获取数据的时候都不会进行加锁，但是在更新数据的时候需要判断该数据是否被别人修改过。如果数据被其他线程修改，则不进行数据更新，如果数据没有被其他线程修改，则进行数据更新。由于数据没有进行加锁，期间该数据可以被其他线程进行读写操作。一般使用version方式和CAS操作方式。

#### 4.3.2 乐观锁的运用

系统采用CAS机制实现乐观锁。CAS（Compare-and-Swap），即比较并替换，是一种实现并发算法时常用到的技术。

在写入文件的时候比较磁盘中指定文件的最后修改时间，当最后修改时间与当前修改文件的不一致（修改之前的）则提示不保存，同时更新到缓存中，如果一致，写入磁盘。

|  |
| --- |
| /\*\*  \* 保存文本文件  \* @param path  \* @param updateTime  \* @param content  \* @return  \*/  public boolean saveFile( String path, long updateTime, String content) {  File f = new File(path);  boolean flag = false;  //文件存在+最后修改时间相等（不等=被改过-->不能修改）  if (f.exists( ) && updateTime == f.lastModified( )) {  System.out.println("aaaaaaaaaaa");  //保存文件的逻辑  FileIOUtil.write(path, content);  LinkedList<String> flist = new LinkedList<>( );  int length = FileTree.fileInfo.getAbsolutePath( ).replace('\\', '/').split("/").length;  String[] slist = path.split("/");  //修改成功需要更新文件树的最后时间  for (int index = length; index < slist.length; index++  ) {  flist.add(slist[index]);  }  FileInfo fileInfo = FileTree.fileInfo.findChildFiles(flist);  fileInfo.setLastModified(f.lastModified( ));  flag = true;  }  return flag;  } |

采用CAS机制可以大大提高服务的并发量，使得公共资源的共享变得高效，也提升了多用户在使用文件系统的时候的体验。

### Markdown的文本编辑格式

系统支持特定文本文件的查看与修改，比如.txt .md等

#### Editor.md

Editor.md是一款国人开发的基于HTML+CSS的Web端Markdown书写格式插件，

支持“标准”Markdown / CommonMark和Github风格的语法，也可变身为代码编辑器；

支持实时预览、图片（跨域）上传、预格式文本/代码/表格插入、代码折叠、搜索替换、只读模式、自定义样式主题和多语言语法高亮等功能；

支持ToC（Table of Contents）、Emoji表情、Task lists、@链接等Markdown扩展语法；

支持TeX科学公式（基于KaTeX）、流程图 Flowchart 和 时序图 Sequence Diagram;

支持识别和解析HTML标签，并且支持自定义过滤标签解析，具有可靠的安全性和几乎无限的扩展性；

支持 AMD / CMD 模块化加载（支持 Require.js & Sea.js），并且支持自定义扩展插件；

兼容主流的浏览器（IE8+）和Zepto.js，且支持iPad等平板设备；

支持自定义主题样式；

#### Editor.md的嵌入使用

在本系统中，文件查看页面和文件编辑页面基于Editor.md的JS文件，调用期中的function，实现了将Markdown的查看与编辑嵌入HTML中，并且MD格式本质上是文本文件，因此也可以查看各类文本文件，代码等。

|  |
| --- |
| 下为调用Editor的JS |
| var testEditor;  $(function () {  testEditor = editormd("test-editormd", {  width: "100%",  height: 750,  autoHeight: true,  path: 'lib/',  theme: "dark",  previewTheme: "dark",  editorTheme: "pastel-on-dark",  codeFold: true,  //syncScrolling : false,  saveHTMLToTextarea: true, // 保存 HTML 到 Textarea  searchReplace: true,  //watch : false, // 关闭实时预览  htmlDecode: "style,script,iframe|on\*", //开启 HTML 标签解析  //toolbar : false, //关闭工具栏  //previewCodeHighlight : false, // 关闭预览 HTML 的代码块高亮，默认开启  emoji: true,  taskList: true,  tocm: true, // Using [TOCM]  tex: true, // 开启科学公式TeX语言支持，默认关闭  flowChart: true, // 开启流程图支持，默认关闭  sequenceDiagram: true, // 开启时序/序列图支持，默认关闭,  imageUpload: true,  imageFormats: ["jpg", "jpeg", "gif", "png", "bmp", "webp"],  imageUploadURL: "./php/upload.php",  onload: function () {  console.log('onload', this);  }  });  }); |

## 运行结果

### 运行环境

开发环境：win10 专业版 2.1GHz 8G内存

开发语言：Java 8

开发工具：IDEA

开发框架：基于Springboot的微服务

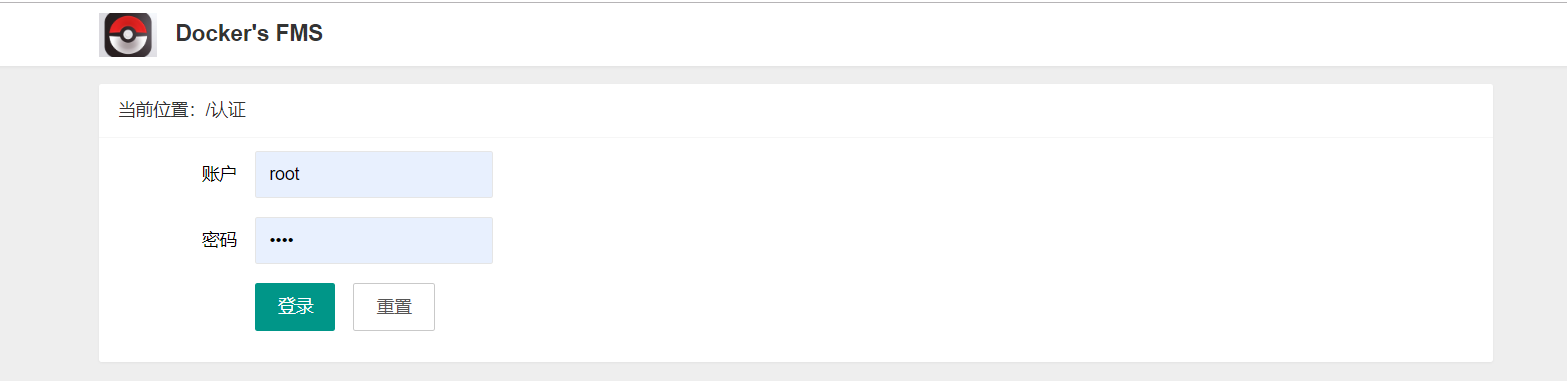
测试环境：阿里云Centos7.3 Xeon 2.2Ghz 2G内存

### 服务模式

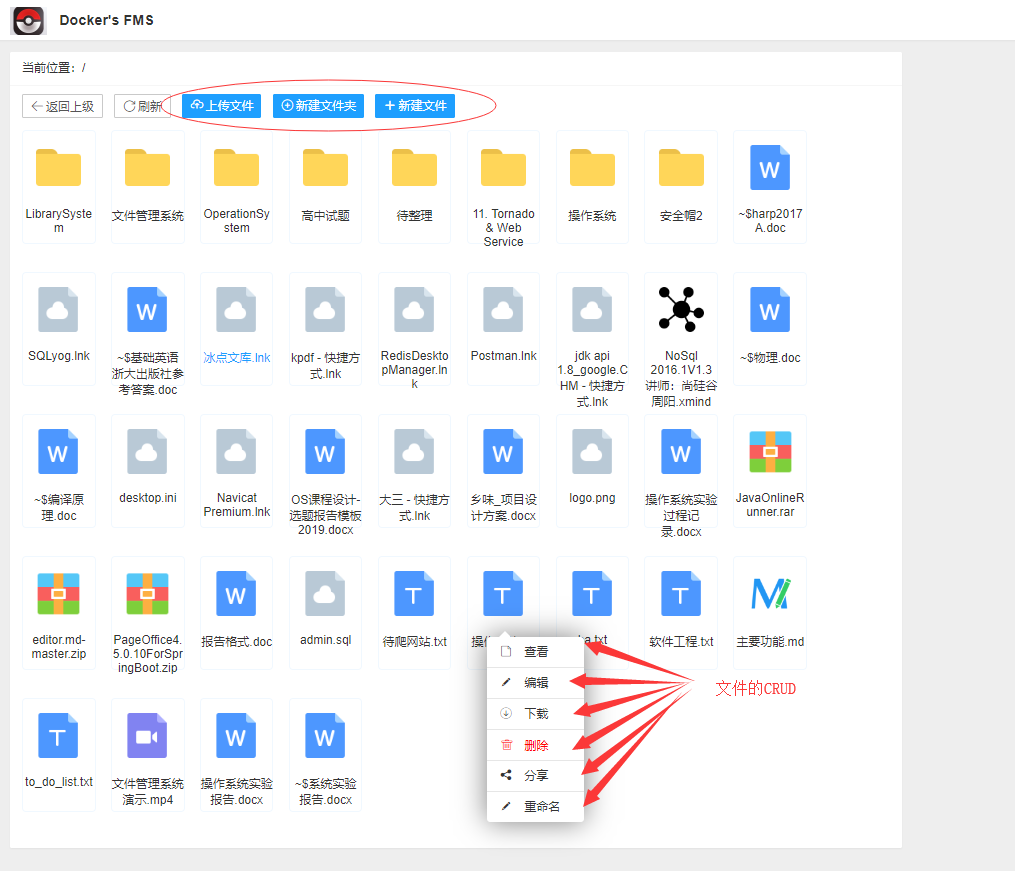
本系统采用B/S 架构，以Springboot的Jar包（内置tomcat）发布服务，后台数据库采用mysql5.7

### 5.3 运行结果

##### 5.3.1 登陆界面



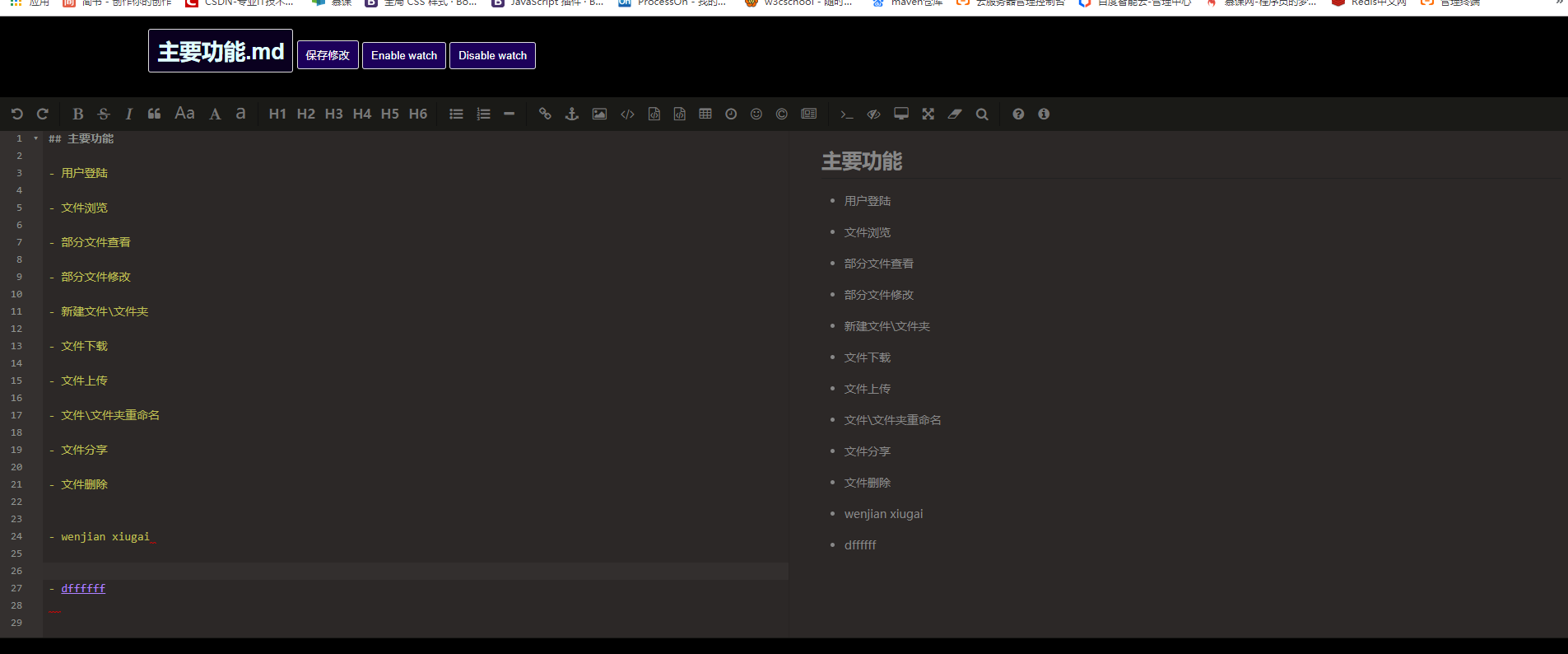
##### 5.3.2 主界面



##### 5.3.3 文件查看界面

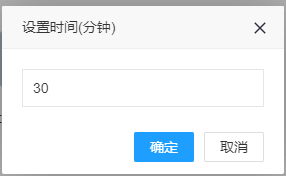


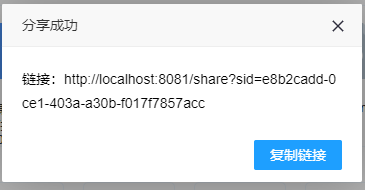
##### 5.3.4 文件修改界面



##### 5.3.5文件分享界面









### 5.4实验结果分析

##### 5.4.1 文件的浏览

浏览功能的实现主要是使用文件树的结构，每个节点除了自己的信息之外，还需要包含子节点的指针，项目启动会初始化一遍文件树，随后使用定时线程每三分钟更新一遍，每次对文件有更改操作的时候就需要更新文件树（部分），在浏览的时候通过URL提取文件相对路径，根据路径递归调用查询方法查找子节点，找到该节点的files队列，根据队列信息，调用文件类型查找对于图标，前端渲染并显示。

##### 5.4.2 文件的查看与编辑

引用Editor.md相关文件在HTML中嵌入Markdown编辑模块，可以将文本文件以较美观的形式展现。

非文本文件的查看则用response输出二进制流，交由浏览器渲染，因此类似pdf、png、MP4是可以在浏览器中浏览的。

##### 5.4.2 文件的分享

文件的分享是在内存中新开一片区域记录分享文件的信息，分享时间等，分享结束后由定时线程每三秒回收一次。

## 调试和改进

使用devtools热部署，使得项目不必每次都stop and run，在前端调试的过程中可能会遇到各种各样的问题，比如数据取不到，浏览器对CSS样式不兼容等问题的，在翻阅资料的过程中逐一解决。

## 7. 心得和结论

### 7.1 结论和体会

本次实验前前后后花费了一周多的时间来完成，其中前期设计并不算十分完美，在开发的过程中不断返工完善设计，才有了最终的效果。就个人来说，前端永远是痛点，永远停留在会写但写不漂亮的阶段，即使是文件图标也找的五花八门，对比之下后端的实现反而比较容易，一方面业务逻辑比较简单，另一方面也写了不少这种Web了，也或多或少积累了些经验。对于课设的要求，我使用了诸如多线程、CAS乐观锁（安全+效率）、缓存模拟等方式表达了出来，也算是有那味了。

### 7.2 进一步改进方向

本项目未来的改进方向其实在一开始就很明确——文件的在线浏览、编辑+服务器的进程、资源监控+权限管理，后两者比较简单，前者需要集成各种插件诸如PageOffice用于在线浏览doc等，在2.0版本会考虑更多文件格式的预览，同时也会加上权限管理与资源监控。