华 南 农 业 大 学 信 息（软 件） 学 院

《操作系统分析与设计实习》成绩单

开设时间：2020 学年第一学期

小组成员、组内分工、工作量比例、各成员个人成绩									
学号	2018xxxxxx	姓名	xxx	分工		工作量比例		成绩	
学号	2018xxxxxx	姓名	xxx	分工		工作量比例		成绩	
学号	2018xxxxxx	姓名	xxx	分工		工作量比例		成绩	
学号	2018xxxxxx	姓名	xxx	分工		工作量比例		成绩	
实验题目	模拟磁盘文件系统实现								
自我评价									
教师评语									
					教师签名	孙微微			

一、需求分析

设计一个简单的文件系统，用文件模拟磁盘，用数组模拟缓冲区，要求：

（1） 支持多级目录结构，支持文件的绝对读路径；

（2） 文件的逻辑结构采用流式结构，物理结构采用链接结构中的显式链接方式；

（3） 采用文件分配表 FAT；

（4） 实现的命令包括建立目录、列目录、删除空目录、建立文件、删除文件、显示文件内容、打开文件、读文件、写文件、关闭文件、改变文件属性。可以采用命令行界面执行这些命令，也可以采用“右击快捷菜单选择”方式执行命令。

（5） 最后编写主函数对所作工作进行测试。

二、概要设计

1. 采用一个文件来模拟电脑磁盘。
2. 本程序共个 14 模块，分别如下：

（1）主程序模块 main()

（2）界面初始化模块 mainview()

（3）模拟文件模块 file()

（4）模拟文件夹模块 folder()

（5）模拟磁盘块模块 block()

（6）文件分配表模块 fat()

（7）已打开文件模块　　fileopened()

（8）文件创建模块　　create_file()

（9）文件夹创建模块　　create_folder()

（10）文件(夹)删除模块　　delete()

（11）文件读写模块　　open_file()

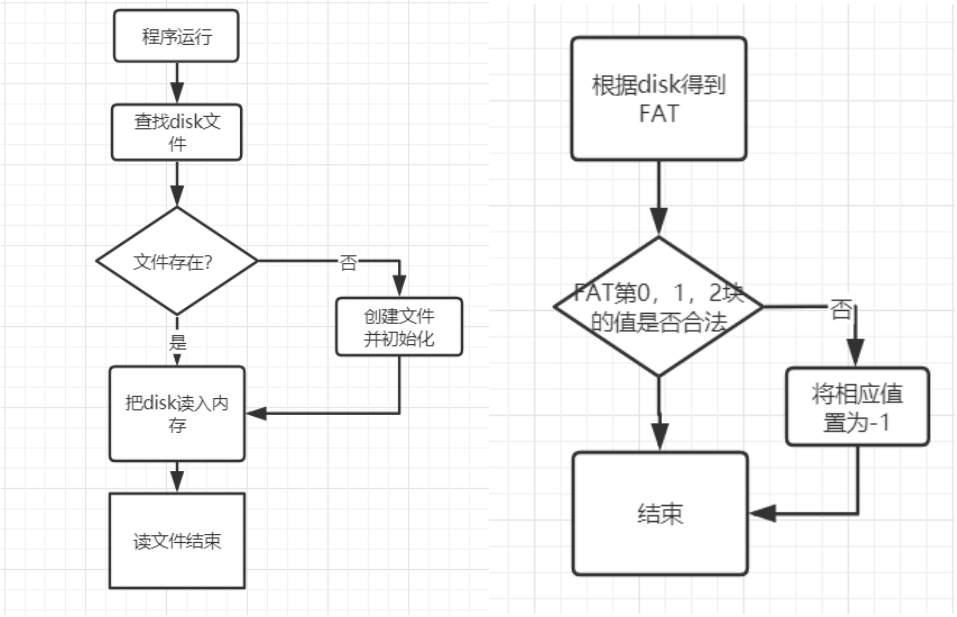
（12）磁盘块分配模块　　allocblock()

（13）重命名模块　　rename()

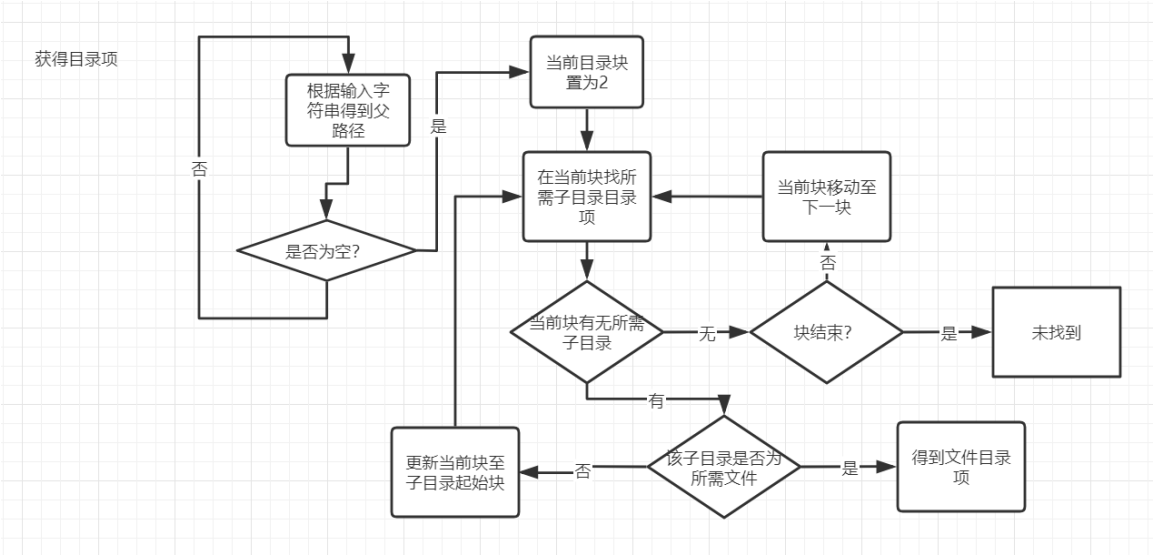
（14）属性模块　　property()

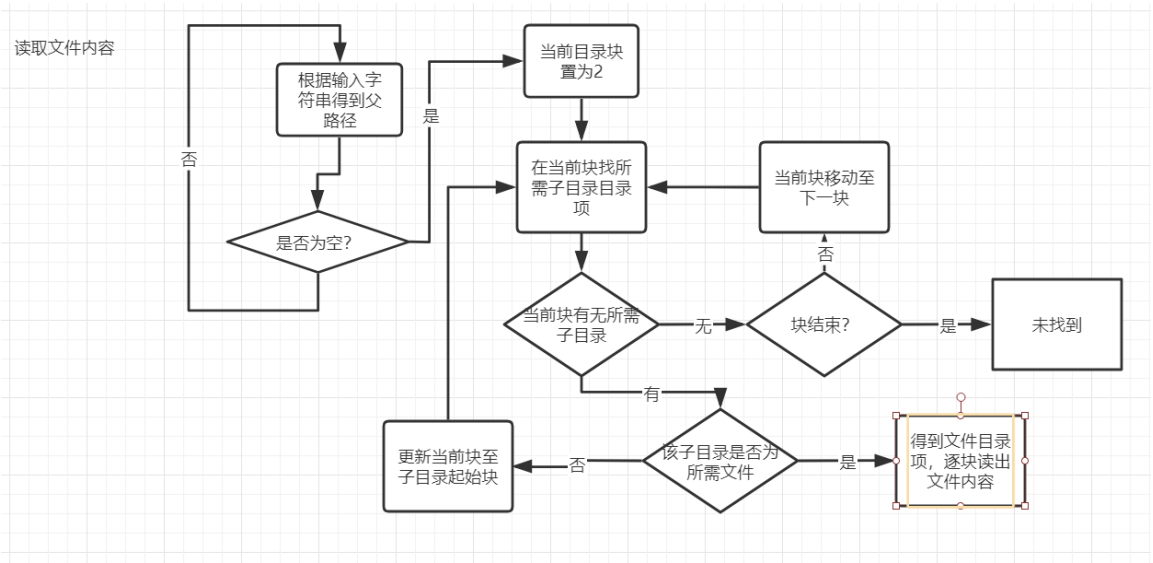
（15）界面显示模块　　desktop()

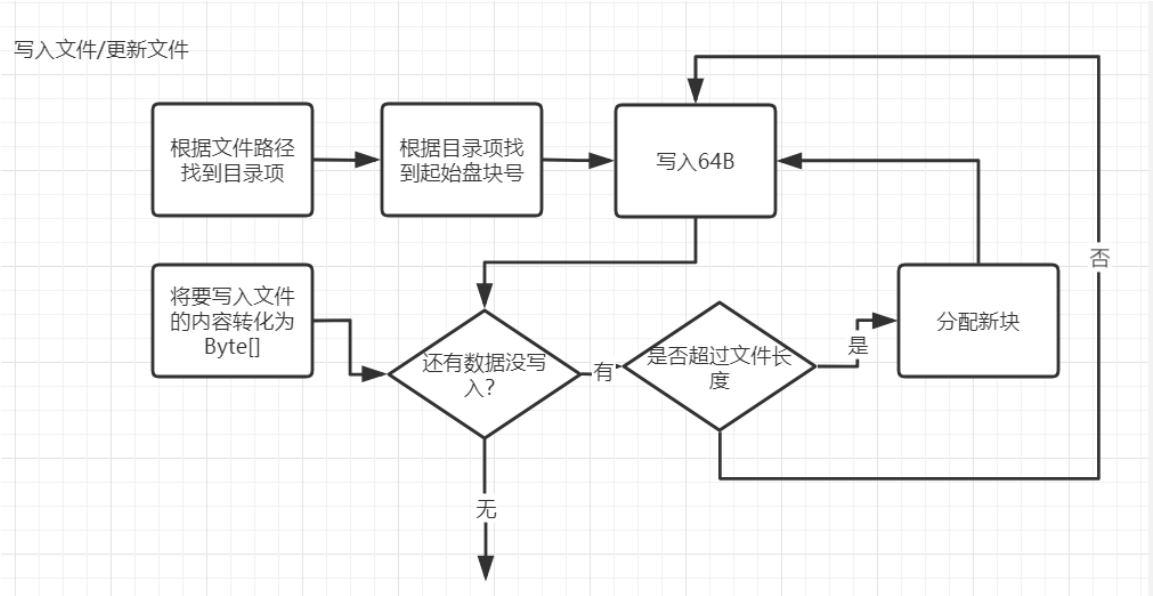
3. 主程序流程图



4. 各程序模块之间的层次关系







三、详细设计

1. 存储结构

disk：一个文件，分为 128 块，每块 64B，第 0 块和第 1 块存储文件分配表，第 2 块为根目录块用于存储根目录。

FAT：FAT 文件分配表一共有 128 个字节，每一个字节存储对应块的下一块位置。

若值为-1 代表结束，0 代表未被使用，254 代表该块损坏无法使用。由于第 0 和第 1 块与第 2 块分别用于存储 FAT 和根目录，所以将该 FAT 的第 0，1，2 字节值固定为-1，且不得更改。

目录项：目录项大小为 8 个字节，前 3 个字节为目录名，第 3 个字节为扩展名，文件属性存在第 4 字节，接下来的一个字节为起始盘块号，最后两个字节存储文件长度。

读写方法：使用 64B 的 byte 数组 readBuffer 和 writeBuffer,读取或写入时先将文

件指针移动到对应位置，然后以块为单位进行存取。

2. 模拟文件模块 file()

文件类 File{

主要成员变量：{

文件名：String fileName

只读/读写标志：int flag

起始盘块号：int num

输入的内容：String content，文件里面保存的字符串

占用的字节大小：double size

父母文件夹：Folder parent

路径：String path

文件打开标志：boolean open

各变量的 StringProperty：用于界面更新

}

构造方法（文件名，路径，起始块号，父母文件夹）{

赋值初始化{

this.fileName = 文件名；

this.path = 路径；

this.num = 起始块号

this.content = “”

setOpen（false）

.....

}

}

各种变量的 getter、setter.

}

3. 模拟文件夹模块 folder()

文件夹类 Folder{

主要成员变量：{

文件名：String folderName

起始盘块号：int num

占用的字节大小：double size

父母文件夹：Folder parent

路径：String path

孩子结点：ArrayList<Object> children

各变量的 StringProperty：用于界面更新

}

构造方法（文件夹名，路径，起始块号，父母文件夹）{

赋值初始化{

this.folderName = 文件名；

this.path = 路径；

this.num = 起始块号;

this.size = 0;

Children = new ArrayList<>();

.....

}

}

各种变量的 getter、setter.

}

4. 模拟磁盘块模块 block()

磁盘块 Block(){

主要成员变量：{

本块号：int num

下一块的索引：int index，0 表示空闲，-1 表示结束，其它小于 128

表示下一块的块号

本块代表的类：Object object

首块标志：boolean begin，用于标记此块是否是文件（夹）的第一块

各变量的 StringProperty：用于界面更新

}

构造方法（本块号，下一块的索引，本块代表的类）{

赋值初始化{

this.num = 本块号；

this.index = 下一块的索引；

this.object = 本块代表的类;

this.begin = false;

.....

}

}

设置本块代表的类 setObject（Object object）{

this.object = object;

if(object instanceof File){

将 object 文件名绑定在 object 的 StringProperty，监听

}

if（该类是文件夹）{

将 object 文件夹名绑定在 object 的 StringProperty，监听

}

}

更新块 update（下一块的索引，本块代表的类，首块标志）{

返回值：void

调用 setXXX 的方法，给改块成员变量重新赋值。

//可用于文件内容超过 64KB 时的块拓展初始化等等。

}

清空块 clearBlock（）{

返回值：void

将 index 置零；

将 begin 置 false；

将 object 置 null；

//可用于块的回收等等

}

各种变量的 getter、setter.

}

5. 文件分配表模块 fat()

文件分配表 FAT{

主要成员变量{

根结点：Folder root

已打开文件表：arraylist<Object> filesOpened

128 个磁盘块的数组：Block[] blocks

}

构造方法（）{

root = 新建文件夹；

Folde fat = 新建文件夹;

初始化 blocks；

blocks 初始化 0，1 号块，装 fat；

blocks 初始化 2 号块，装 root；

for（int i=3； i<128; i++）{

blocks[i] = new Block(i,0,null);

//块号为 i，index 为 0，object 为空.

}

初始化已打开文件表；

}

addFilesOpened（Block block）{

返回值：void

//文件打开时，需要更新已打开文件表

file =block 里面提取的类；

将 file 加进已打开文件表；

将 file 的已打开标志置 true

}

removeFilesOpened（Block block）{

返回值：void

//文件关闭时，需要更新已打开文件表

file =block 里面提取的类；

将 file 从已打开文件表删除；

将 file 的已打开标志置 false;

}

getEmptyBlock(){

返回值：int

//进行创建、块拓展时，需要获得第一块空闲块

for(int i=3; i<128; i++){

If(blocks[i].isEmpty){

return i;

}

}

}

saveDate(FAT fat){

返回值：void

//在系统关闭时，自动调用该方法，将 fat 类的数据保存到“disk”文件中

save = 新建 ObjectOutputStream(“disk”);

save.writeObject(fat);

输出保存结果；

}

getEmptyBlocksCount(){

返回值：int

//当需要进行块的分配时，先查看是否还有空闲块

int n = 0；

for(int i=2; i<128; i++){

if(blocks[i].isEmpty)

n++;

}

返回 n；

}

reallocBlocks（int num，Block block）{

返回值：boolean

//文件长度变化时重新分配盘块

file = 从 block 里面的文件；

begin = file 的起始块号；

index = 起始块的 index

oldnum = 1；

//让 index 指向文件旧最后的那个块号,oldnum 计已用块数

while（index 不等于-1）{

oldnum++;

if(blocks[begin]的 index 不为-1){

begin = index;

}

index = 该 begin 块的 index；

}

If（num>odlnum）{

//增加磁盘块 n 个

检查 n 有没有超磁盘容量;

得到一个空闲块；

先让最新的那个块的 index 指向一块新的盘块；

for(int i=0;i<n; i++){

得到一个空闲块；

If(i==n-1){

//已经分配到最后一个

调用对应 block 的 update 方法更新块信息;

}else{

调用对应 block 的 update 方法更新块信息;

得到一个空闲块；

这一块的 index 指向新得到的空闲块；

}

}

}else if（num<odlnum）{

//需要回收 n 个块

end = 文件起始块；

//让 end 指向最后一个块的块号

while（num>1）{

end = end 块号的块的 index；

num--；

}

Int j =0;

//继续 end 的索引开始清空块

for(int i=end; i！=-1; i=j){

j = i 号块对应的 index；

blocks[i] 清空；

}

将 end 号块的 index 置 0；

}

}

getFolders(String path){

返回值：arraylist<Folder> list

//返回 path 路径下的所有文件夹

for(int i=2; i<128; i++){

If(i 号盘块不为空){

if（i 号盘块的 p 路径==path）{

if（i 号盘块的类是文件夹）{

list 中加入 i 号盘块的文件夹；

}

}

}

}

return list；

}

getFather（String path）{

返回值：folder

//返回指定路径指向的文件夹，通常原来得到父母结点

检查如果是根路径，直接返回 root；

以“\\”为分割点，从 path 后面开始得到第一个分割点 index；

path2 = path 中 0 到 index-1 的字符串；

fatherName = path 中 index+1 到最后的字符串；

调用 getFolders(path2)得到 list；

for（list 中的每个 folder）{

如果 folder 的名字==fatherName；

返回这个 folder；

}

没找到，返回 null；

}

getBlockList(String path){

返回值：arraylist<Block> list;

//主要是更新界面的当前目录下的文件（夹）图标用

```
//先获得文件夹再获得文件
for(int i=2;i <128; i++){
If(blocks[i]不为空){
If(blocks[i]的类是文件夹）{
if（它的路径==path&&该块是起始块){
list 中加入这个 block；
}
}else if（它是文件）{
if（它的路径==path&&该块是起始块){
list 中加入这个 block；
}
}
}
return list；
}
}
getFolderSize（Folder folder）{
返回值：double size
//递归得到一个文件夹的字节大小
list = folder 的孩子结点列表；
size=0；
for(list 中的每一个结点 obj){
```

If（obj 是文件）{

size+=obj 的字节大小

}else{

size+=getFolderSize（（Folder）obj）

}

}

return size（保留 2 位小数）；

}

各种 getter，setter；

}

6. 文件创建模块 create_file()

新建文件 create_file(){

返回值：int 新文件的起始块号 num

index=1；

do{

//先得到新的文件夹名 newName（“文件”+index)

for(int i=2; i<128; i++){

If(blocks[i]不为空){

If（blocks[i]是文件）{

如果 blocks[i]的名字==newName

不可命这个名；

}

}

}

Index++；

}while（不可命名）

if（磁盘有空闲块）得到空闲块号 num；

else return　255//没空间了

father = 新建文件夹的父母结点；

file　= 新建文件；

file 置为可读可写；

father 的孩子结点中加入 file；

blocks[num]更新信息；

return num；

}

7.　文件夹创建模块 create_folder()

新建文件夹 creat_folder(String path){

返回值：int 新文件夹的起始块号 num

index=1；

do{

//先得到新的文件夹名 newName（“文夹”+index)

for(int i=2; i<128; i++){

If(blocks[i]不为空){

If（blocks[i]是文件夹）{

如果 blocks[i]的名字==newName

不可命这个名；

}

}

}

Index++；

}while（不可命名）

if（磁盘有空闲块）得到空闲块号 num；

else return　255//没空间了

father = 新建文件夹的父母结点；

folder　= 新建文件夹；

father 的孩子结点中加入 folder；

blocks[num]更新信息；

retun num；

}

8.　文件(夹)删除模块 delete()

文件(夹)删除 delete(Block block){

If(block 代表文件并){

if（它已经打开）

return 3；//不能删除已经打开的文件

}

file= block 代表的类；

father = file 的父母结点；

从 father 的孩子中移除这个 file 结点；

father 重新设置获取的新大小 size；

while(father 有 parent){

parent 重新设置获取的新大小 size；

father= parent；

}

for（int i=2；i<128; i++）{

If(blocks[i]代表的类==block 代表的类){

释放清空这个块；

}

}

}else{

得到 block 代表的文件夹的孩子的路径 path；

for 循环{

检测如果 block[i]代表的类的路径==path，

return　2；//文件夹不为空，删除失败；

如果文件夹为空，index=i；

}

获得 i 号块的文件夹 folder；

得到 folder 的 father；

从 father 中删除 floder；

释放清空 i 号块；

return 0；//删除成功

}

9. 文件读写模块 open_file() {

//具体见 mainview 模块

打开文件；

置 Open 为 true；

获取输入框的内容，更新 file 的 content；

根据新的 content 字节大小，按需重新分配块；

}

10. 磁盘块分配模块 allocblock()

磁盘分配 reallocBlocks（int num，Block block）{

返回值：boolean

//文件长度变化时重新分配盘块

file = 从 block 里面的文件；

begin = file 的起始块号；

index = 起始块的 index

oldnum = 1；

//让 index 指向文件旧最后的那个块号,oldnum 计已用块数

while（index 不等于-1）{

oldnum++;

if(blocks[begin]的 index 不为-1){

begin = index;

}

index = 该 begin 块的 index；

}

If（num>odlnum）{

//增加磁盘块 n 个

检查 n 有没有超磁盘容量;

得到一个空闲块；

先让最新的那个块的 index 指向一块新的盘块；

for(int i=0;i<n; i++){

得到一个空闲块；

If(i==n-1){

//已经分配到最后一个

调用对应 block 的 update 方法更新块信息;

}else{

调用对应 block 的 update 方法更新块信息;

得到一个空闲块；

这一块的 index 指向新得到的空闲块；

}

}

}else if（num<odlnum）{

//需要回收 n 个块

end = 文件起始块；

//让 end 指向最后一个块的块号

while（num>1）{

end = end 块号的块的 index；

num--；

}

Int j =0;

//继续 end 的索引开始清空块

for(int i=end; i！=-1; i=j){

j = i 号块对应的 index；

blocks[i] 清空；

}

将 end 号块的 index 置 0；

}

}

11. 重命名模块 rename()

重命名 rename(Block block，String newName){

If(有重名) return；

If（block 代表文件）{

这个文件的名字 =newName；

}else{

这个文件夹的名字 =newName；

rePath（path，newName，block 的类）

}

}

repath（String path，String newName，Folder folder）{

返回值：void

//递归路径

newPath = path+folder 的名字；//这个 folder 的孩子的路径

for（folder 的每个孩子结点 obj）{

如果 obj 是文件：obj 的 path=newPath；

else{

obj 的 path = newPath；

if（obj 有孩子结点）{

repath（newpath，obj 的名字，obj）}

}

}

}

}

12. 属性模块 property()

设置“属性”点击事件 PropertyView（）{

显示文件或文件夹属性界面{

名称：文件名（或文件夹名）

文件类型：FILE（或 FOLDER）

位置：文件路径

大小：占用字节数

属性：只读或读写（仅文件类型可操作）

}

}

}

13. 界面初始化模块 mainview()

包含文件系统各个面板和部件，如目录树，文件显示面板，磁盘使用情况面板，及其内的各种 Item 与事件处理方法。

mainview{

根据 disk 载入 FileItem，

如 disk 不存在，创建 dsik

initFrame 初始化窗口并显示

}

initFrame(Stage stage){

对窗口的标题等进行设置。

initContextMenu();

menuItemSetOnAction();//初始化菜单栏并未菜单栏中每个选项

添加事件

initTopBox(); initTables(); initTreeView();//初始化各个面板

}

initTopview(){

读取当前打开目录。加入显示目录文本框

加入后退，前进按钮。

为各个按钮添加点击事件，当点击按钮时更换目录。

}

initTables(){

在界面右侧初始化表格 tableview，并为表格添加磁盘使用情况。

并使用 setCellValueFacetory 方法绑定每一列的属性类型，以用于表格中数据的自动更新。

打开的文件的列表。表中每一列都与 DiskBlock 与 File 中相应的属性绑定，以此实现数据更新。

}

initTreeview(){初始化目录树函数根据初始化时载入的 FileItem 和

getChildren 方法

为目录树逐个添加节点。

使用 setCellFactory()方法来设置目录内容，遍历文件以及目录来为 treeview 添加结点。

}

intiTreeNode(){

用 TreeItem 的对象来添加新结点;

if（目录树下已经有结点）

{

通过递归调用结点初始化函数 initTreeNode（）来完成目录树下数据更新。

}

}

initContextMenu（）{//初始化鼠标右键菜单栏

添加“新建文件”、“新建文件夹”两个菜单项到窗口右键菜单栏 contextMenu；

添加“打开”、“删除”、“重命名”、“属性”四个菜单项文件右键菜单栏

contextMenu2；

设置“新建文件”点击事件创建新文件 create_file()，删除 flowPane 中的图标，

更新界面 updateShow（）；

设置“新建文件”点击事件创建新文件夹 create_folder()，删除 flowPane 中的
图标，更新界面 updateShow（）；
设置“打开”点击事件 onOpen（）{
if（点击目标为文件）{
打开文件界面 FileView（）；
}else{//点击目标为文件夹
设置路径为该文件夹，删除 flowPane 中的图标，更新界面为该文件夹 updateShow（）；
}
}
设置”删除“点击事件{
删除文件或文件夹 delView（）{
如果文件为打开状态，删除失败；
如果文件夹内有子文件或子文件夹，删除失败；
否则删除该文件或文件夹；
显示删除成功或失败提示框；
}
删除 flowPane 中的图标，更新界面 updateShow（）；
}
设置”重命名“点击事件 RenameView{
输入新名称 newName；

if（newName 超出长度或重名）重新输入；

if（如果为文件）修改文件名字，更新图标；

if（如果为文件夹）[

修改文件夹名字；

if（该文件夹有子文件）{

reLoc（）；修改子文件的路径名

}

}

}

设置“属性”点击事件 PropertyView（）{

显示文件或文件夹属性界面{

名称：文件名（或文件夹名）

文件类型：FILE（或 FOLDER）

位置：文件路径

大小：占用字节数

属性：只读或读写（仅文件类型可操作）

}

}

}

updateShow{//更新文件、文件夹图标

for（遍历当前选中目录的子节点）{

if（是文件夹）icons=new Lable(文件夹图标，文件夹名)；

else　icons=new Lable(文件图标，文件名)；

将图标 icons 添加到 flowPane 面板的 children 中显示图标；

设置图标右键点击显示菜单栏和双击打开文件（或文件夹）；

}

}

14. 界面显示模块 desktop()

desktop 模块是显示一个类似系统桌面的界面，主要包含 Desktop（），ContainPane（），FolderPane（））。

Desktop{//系统桌面

if（本地有存储文件 disk）{

读取本地的 FAT;

}else{

创建新的 FAT=new FAT（）；

}

创建图标界面 ContainPane（）；

设置关机按钮 creareShutDownIcon（）{

创建关机按钮；

点击关机可选择是否退出系统；

}

}

ContainPane（fat）{//图标界面

this.fat = fat;//初始化 fat

initContextMenu{//初始化右键菜单栏

添加“打开”、“删除”、“属性”三个菜单项右键菜单栏；

设置“打开”点击事件 onOpen（）{

if（点击目标为文件）{

打开文件界面 FileView（）；

}else{//点击目标为文件夹

打开文件夹界面 FolderPane();

}

}

设置”删除“点击事件{

删除文件或文件夹 delFile（）{

如果文件为打开状态，删除失败；

如果文件夹内有子文件或子文件夹，删除失败；

否则删除该文件或文件夹；

显示删除成功或失败提示框；

}

}

设置“属性”点击事件 PropertyPane（）{

显示文件或文件夹属性界面{

名称：文件名（或文件夹名）

文件类型：FILE（或 FOLDER）

位置：文件路径

大小：占用字节数

属性：只读或读写（仅文件类型可操作）

}

}

}

创建图标 createFileIcon（）{

创建“关于”、”帮助“、 “文件管理”三个图标 createicon{

Label icon；//用标签组件，上面显示图标，下部分显示图标名称

设置点击事件分别为

AboutPane（），//显示关于小组信息界面

HelpPane（），//显示系统操作指南界面

MainView（）；//显示文件管理界面

获取系统 c 盘路径 C：下的文件和子目录

bList=fat.getBlockList (“C:");

for(遍历 blist）{

if（是文件夹）fIcons=new Lable(文件夹图标，文件夹名）；

else　fIcons=new Lable(文件图标，文件名）；

将图标 fIcons 添加到 FlowPane 面板中；

设置图标右键点击显示菜单栏和双击打开文件（或文件夹）；

}

}

}

}

FolderPane{//文件夹窗口

获取该目录下的文件和子文件夹 bList；

创建文件、子文件夹图标。。。（与 ContainPane 创建图标及鼠标事件一致）

}

15. 主程序模块 main()

四、调试分析

(1) 调试过程中遇到的问题是如何解决的以及对设计与实现的讨论和分析

在调试过程中，会遇到了如打开文件失败(报错文件已打开)，磁盘使用情况图标动态更新不正确等问题。对于打开文件失败(报错文件已打开)，我们小组几人的电脑上出现了不同的情况，　有些人可以打开，有些人可以打开新建的但不能打开上次程序结束前保存的，有些人一个都打不开，对于这个问题，我们讨论觉得是“已打开文件表”的读取错误，原以为是类和在关闭文件时保存信息错误，但是发现并没有，关闭文件时已经把这个文件从已打开文件表移除，后来再次尝试发现这个问题只在小组成员间文件传递才会出现，可能是因为编码不同的问题，所以尝试删除一次磁盘文件，发现问题就解决了。对于磁盘使用情况图标动态更新不正确的问题，

我们起初的更新是创建文件（夹）时将空磁盘块对于磁盘使用情况图标取出来，改颜色再装进去，但是删除文件（夹）和文件输入内容后，没有考虑到磁盘块已经重新回收和分配，而更新只更新了文件夹）的首块，但是如果是在块重新分配和回收的地方循环更改图标颜色，竟然会出现反射异常，觉得没办法后，我们试图给 block 的 int 索引增加一个监听器，通过测试，发现还是反射异常，因此，我们就采用了新的方法，就是在创建、删除和保存输入文件的内容后，重新 128 个初始化磁盘使用情况图标的颜色，每次都整个表刷新，问题得以解决。

(2) 设计过程的经验和体会

xxx：在本次课程设计中，我主要负责系统桌面设计实现，文件图标创建与更新，鼠标点击事件实现。这次课程设计中，虽然分工是界面和逻辑结构连接的部分，但也查阅和学习一些内存、文件方面的知识，也学习了组员的底层代码的实现。也学到了在类创建时运用获取实例的方法让程序各部分都使用同一个实例的方法。但由于组内沟通不及时，在完成自己负责的模块时，没能很好的将各个不同的模块分隔开来，给后续设计工作带来了一些麻烦。

xxx：在本次课程设计中，我主要负责文件管理系统的界面设计、界面属性配置以及界面中目录树和表格内部数据的更新。在这个过程中，我了解了界面设计在代码实现上所需要的逻辑关系，包括容器之间的嵌套、位置调整以及组件之间的分布。除此之外还了解了如何完成目录树以及表格的数据更新。由于没有掌握多线程编程技术，因此只能另辟蹊径，尝试用别的办法，例如递归、监听器绑定等等来完成。这次过程让我体会到编程真的是一项团队工作，每一个人都得前进才是前进，并且一定要沟通频繁，才能联系的更密切，做出来的成品也才能融合的更好。

xxx：在这次课程设计中，我主要负责的是文件操作即创建，删除，重命名，重

新分配块及其相关方法的底层编写。代码编写前，我参考了现有的样例，学习他们的写法思想，一开始编写时，组内的沟通不良好，不是很清楚要写什么才能满足其他组员的使用需求，但是在不断的交流中，终于逐渐地达到要求。在编写代码中，我遇到的最大问题就是重命名后子结点路径的更新和删除文件后父 结点的大小更新，对于路径更新，我一开始采用的是多重循坏找孩子，从上往下更 新,但是通过测试结果发现子文件夹也有子文件夹，子子文件夹也有子文件夹，很 显 然 不 能通过简单的循环来更改，后来查阅资料发现，这就是一个很典型的树的遍 历 问题，直接用递归，在遇到子文件夹时改变参数再次调用原方法，遇到文件时则直接改名而不需递归，测试的结果完全正确，运用这种递归的思想，我在删除文件后父结点的大小更新时，也是不断地向下递归返回得到大小，向上循环不断更新父结点大小，这种方法非常高效而且简单。通过这次实验，我发现了自己的编程能力仍然需要向进阶的发现去努力，在团队的开发中应该加强和组员的沟通，一块块分开做，没有前期的需求分析规划是做不出结果的。

xxx：在这次课程设计中，我主要负责的是整体样式、各页面样式、底层数据与界面的交互设计编写、界面类的编写和程序调试，由于要给所有页面调改样式，所以我需要将每个界面的大概结构都能看明白，并插入样式表或是内部嵌入对它们的样式进行修改，刚开始的时候由于缺乏和组员的沟通，自己走了很多弯路，自己看懂看明白组员的代码实在花耗自己太多时间，于是我后来马上去问了每个组员，他们所写界面的整体结构，于是我很快就把各个界面都插入不同样式，使之从几个杂乱的界面变得简洁明了。在整体样式风格上，出于个人喜欢以及它的应用广泛，我参考了 window10 初始桌面的蓝色简约风格，在各个页面尽量以简洁明了为原则，颜色以蓝为主。这次页面的设计，在没法使用 css 库和 js 还会出现部分 css3 属性不

支持的情况下，大部分用原生 css 对页面进行更改，让我更加了解页面设计中较为底层的基础，这让我以后写的样式会有更多可能性。

(3) 实现过程中出现的主要问题及解决办法

在实现过程中，我们主要遇到的问题是目录树的更新以及 fat 表的更新，我们最初的想法就是在每次文件夹创建删除，块使用情况变，每次重命名等情况时，每次都调用一次全局刷新的方法，但是这种方法实现起来及其困难，因为有非常多种情况要考虑，而且每一次都要执行全局的刷新，有可能会导致界面不流畅，通过和参考样例和查 jdk 文档，我们发现用属性绑定界面变化可以很地解决单一子结点刷新的问题，比如在文件（夹）等类中给需求变量添加属性特征，就可以在目录树时使用 setCellFactory()方法来设置目录内容，遍历文件以及目录来为 treeview 添加结点，而在 fat 图的信息更新中，可以用方法 setCellValueFacetory 绑定每一列的属性类型，以用于表格中数据的自动更新。这样有效解决了程序底层数据和界面更新的交互问题。

五、用户使用说明

1、在桌面中，打开关于界面可查看小组信息；打开帮助可查看系统操作指南；

2、鼠标左键双击文件夹可进入该文件目录；双击文件可查看文件内容进行编辑；

鼠标右键单击文件或文件可选择打开、删除、属性，打开文件（文件夹），删除文件（文件夹），查看文件（文件夹）属性。

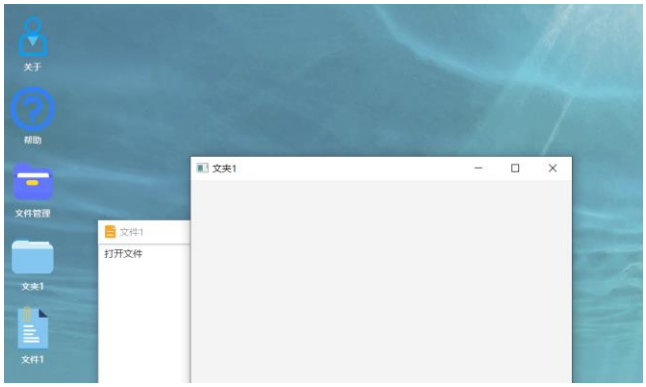
3、双击文件管理可进入文件系统管理界面，左边选择目录树可查看相对应的目录下的文夹与文件夹；右边为文件分配表信息；中下为磁盘使用情况；中上为选中目录中的文件与文件夹信息，左键双击可打开文件（文件夹），右键单击可选择进行打开、删除、重命名、属性等操作，右键单击空白可新建文件或文件夹。

六、测试与运行结果

桌面

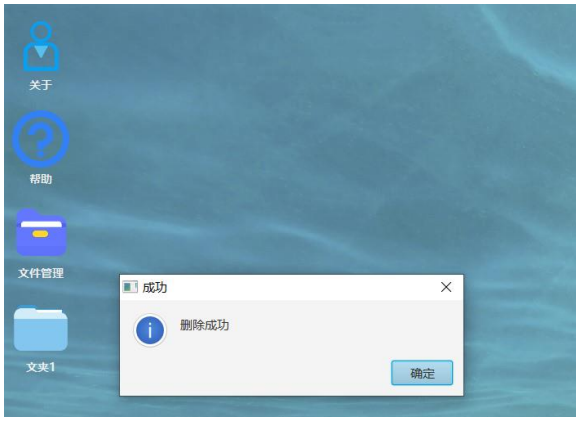
右键菜单栏功能测试

打开文件、文件夹

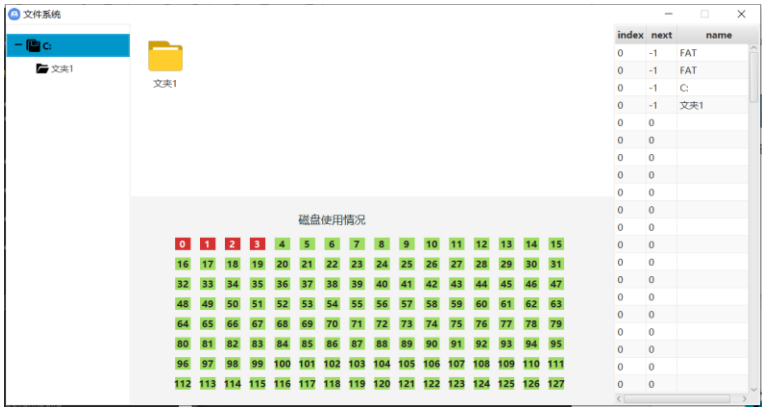


文件、文件夹属性

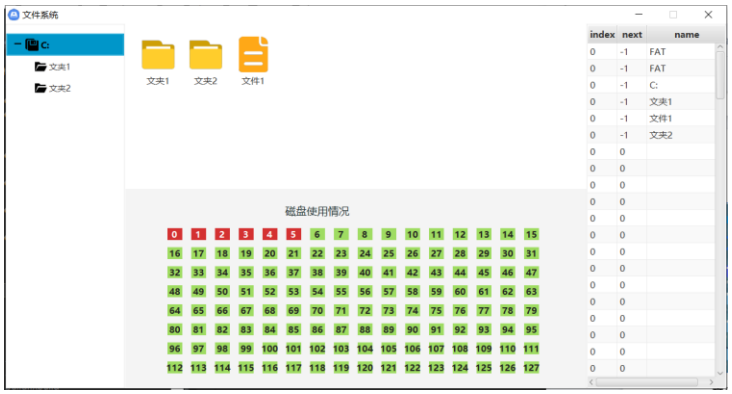
删除文件



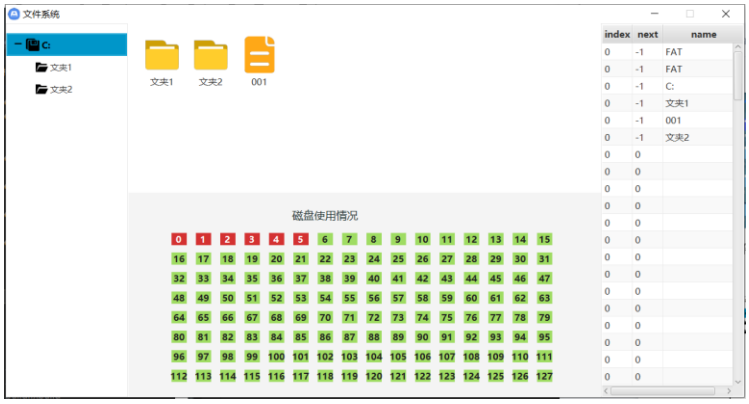
文件管理



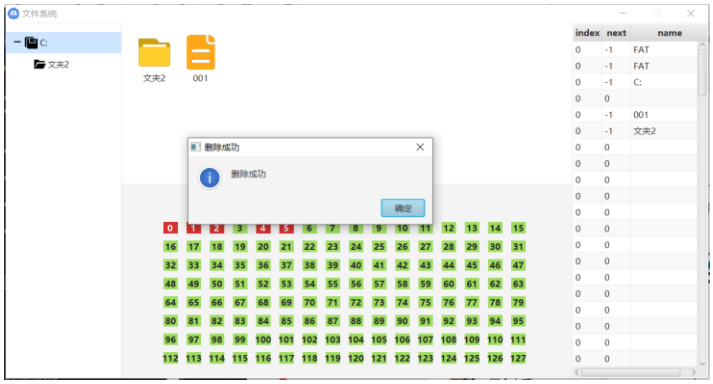
新建文件 1、文夹 2，文件分配表、磁盘使用情况与预期符合



文件 1 重命名为 001



删除文夹 1,文件分配表相应项清空，磁盘使用情况更新


文件系统
C:
文夹2
文夹2
001
删除成功
删除成功
确定
index next name
0 -1 FAT
0 -1 FAT
0 -1 C:
0 0
0 -1 001
0 -1 文夹2