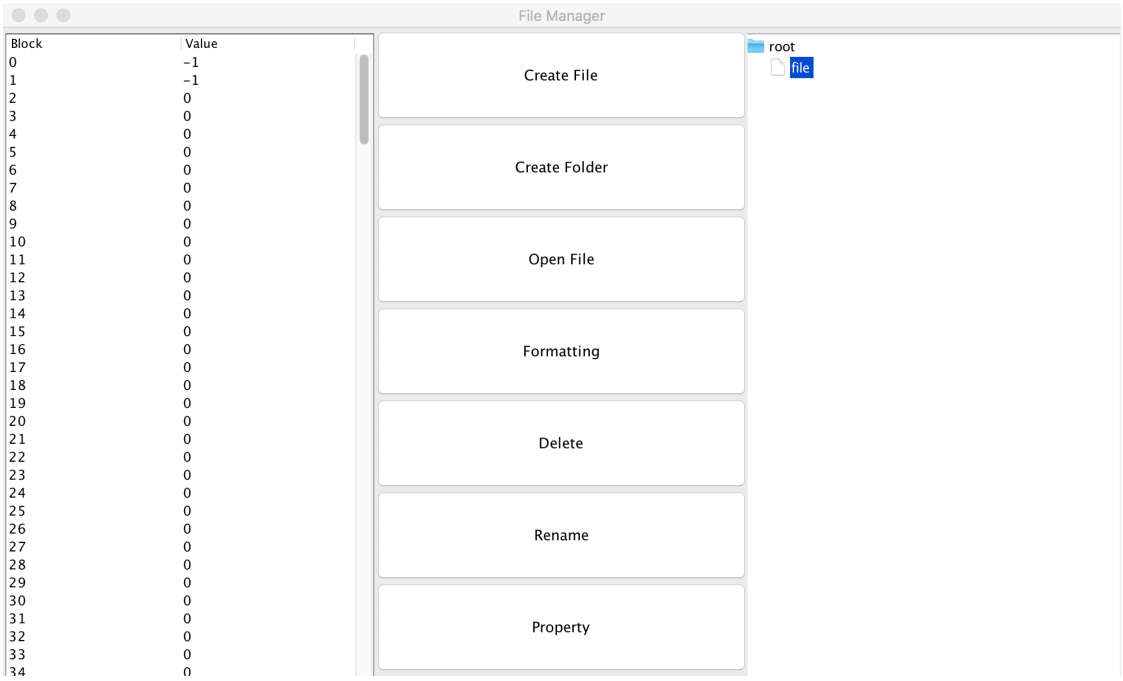
操作系统项目三——文件管理项目文档
1552681 陈淇格

1.开发环境
IDE: Intellij IDEA
语言：java

2.项目简介
在内存中开辟一个空间作为文件存储器，在其上实现一个简单的文件系统。
在系统中，可以创建文件、文件夹，重命名文件、文件夹，写文件内容，查看文件、文件夹属性，删除文件、文件夹，格式化文件、文件夹。
退出这个文件系统时，该文件系统的内容保存到磁盘上，以便下次可以将其回复到内存中来。

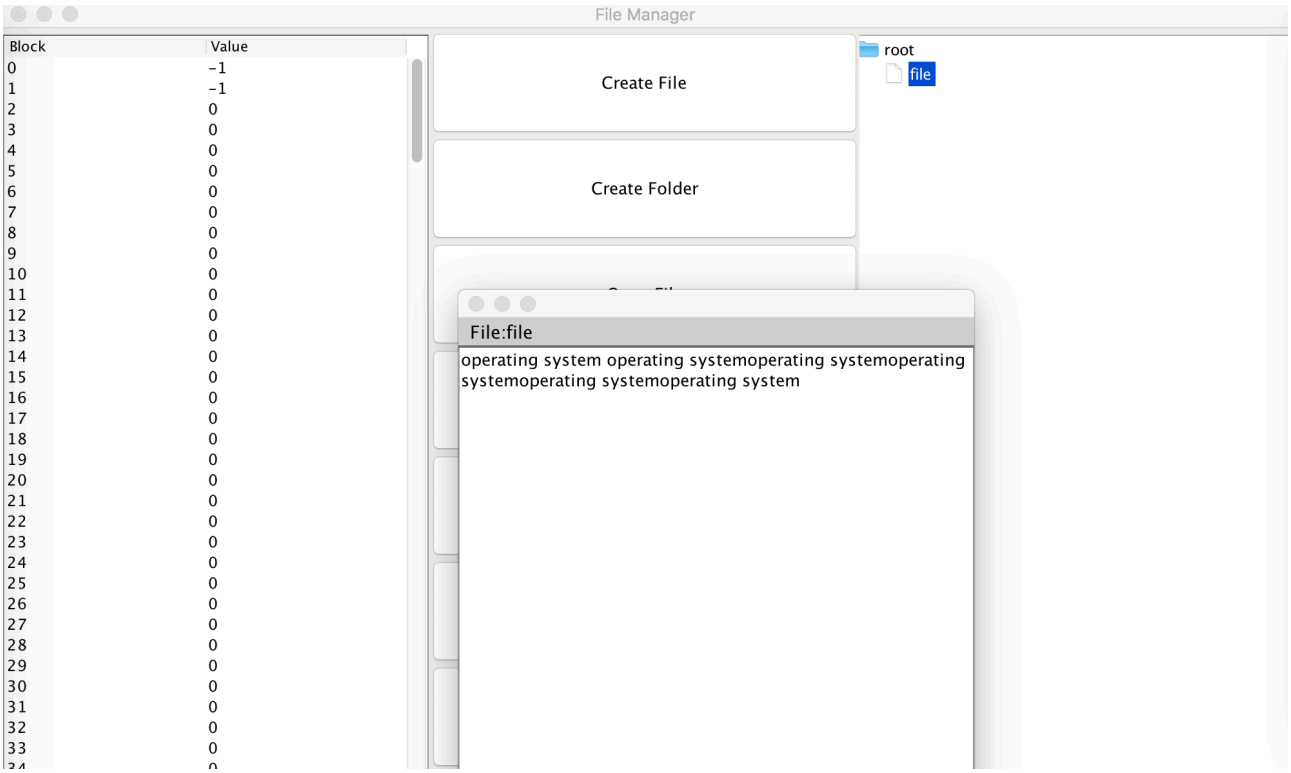
3.操作方法
打开File-Manager.jar。
（1）创建文件



在最右侧的树形目录中选中想创建文件的父文件夹，点击Create File 按钮，输入文件名称，文件被创建。界面最左侧的面板出现FAT的显示，Value为-1表示

文件末尾，为0表示块空，否则，value表示下一块的地址。FAT的第0块永远为-1。

（2）打开文件



在树形目录中选择要打开的文件，点击Open File按钮。可以在文本框中输入文件内容，并在按钮栏中点击“File”菜单下的“save”按钮进行保存。如果不保存直接关闭文本框，则文本不会被保存。文件大小改变，FAT的内容也随之改变。注意不能对文件夹进行打开文件的操作。

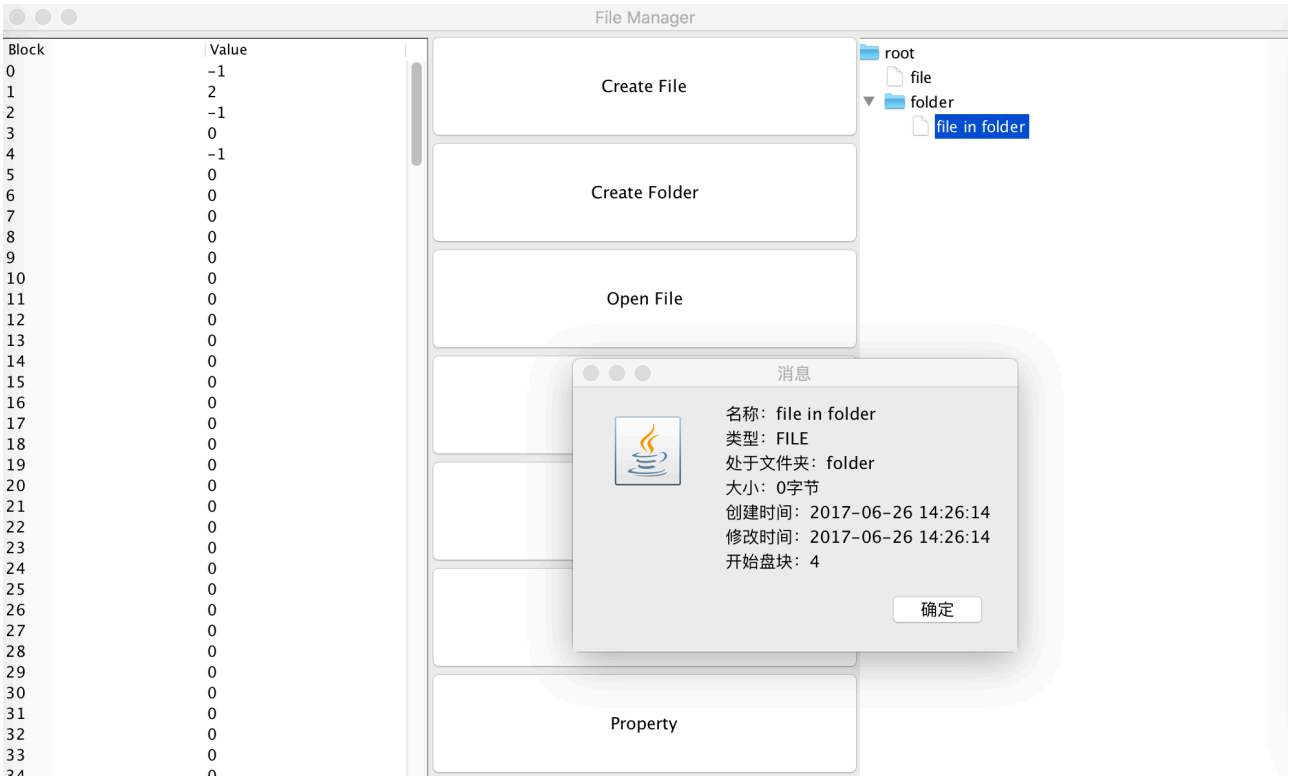
（3）重命名文件/文件夹

在树形目录中选择要重命名的文件或文件夹，点击“Rename”按钮，输入新的名字。注意同一文件夹下文件和文件不能重名，文件夹和文件夹不能重名。

（4)创建文件夹

在树形目录中选中想创建文件夹的父文件夹，点击Create File 按钮，输入文件夹名称，文件夹被创建。

（5）查看文件/文件夹属性



在树形目录中选择要查看属性的文件或文件夹，点击Property按钮,系统显示文件或文件夹的属性，包括名称、类型（是文件还是文件夹）、父文件夹名称、文件大小、创建时间、修改时间、以及文件开始存放的盘块。

（6）格式化文件/文件夹

在树形目录中选择要格式化的文件或文件夹，点击Formatting按钮。如果选择格式化文件，则文件大小会改变为0，文件中的文本也会清空，文件的修改时间被改变，文件所在盘块内容为-1，表示该块有文件，但是没有占满该块。

如果格式化文件夹，则文件夹下所有的文件都会被格式化。

（7）删除文件/文件夹

在树形目录中选择要删除的文件或文件夹，点击Delete按钮，FAT表中存放文件的块显示为0，树形目录中文件被删除。

如果删除文件夹，则文件夹下所有的文件和文件夹都会被删除。

注意根目录root不能被删除。

4.设计思路

文件存储空间管理采用显式链接的FAT方式，用一个大小为256的int型数组模拟FAT表，数组的每一位都表示下一块所在的地址。

将空闲空间管理与FAT 表相结合，用int型数组构造空闲空间栈，为文件分配新的空间时，从栈中取栈顶的块。当删除文件时，释放文件所占的所有FAT块，将他们的值修改为0（也就是空块），并将这些块的序号压栈。栈顶top的值代表着当前的空块数。

文件图形化目录采用树形目录结构，用java swing的JTree树形控件来模拟文件目录结构的图形化表示。

文件逻辑目录采用链表，为每个文件和文件夹构造文件控制块FCB，放入目录链表中。

5.核心算法

（1）FAT与空闲空间管理

FAT的链接结构用数组表示。比如int[a]=b,则第a块的下一块为b。当int[a]=0时，表示第a块为空。int[a]=-1时，表示第a块为文件的末尾。

```
//获取下一块序号
public int getNextBlock(int currentBlockNum) { return fatBlockList[currentBlockNum]; }

//设置块与块间链接
public void setNextBlock(int currentBlockNum,int nextBlockNum) { fatBlockList[currentBlockNum]=nextBlockNum; }
```

将文件存入FAT时，先获取空闲块的栈顶指针top的值，也就是空闲块的数量。计算文件所需要的块数量。这里设定的磁盘是一个256×64的byte型数组。首先将文件的内容转化为byte数组，并计算其大小（文件所占字节数），再用这个大小除以64并向上取整，获取文件所需要的块数量。

如果空闲块数量小于文件所需要的块数量，那么系统提示空间已满，无法创建文件。

如果空闲块数量大于等于文件所需要的块数量n，则不断将n个块从空闲空间栈中出栈，并设置他们之间的链接。文件被存入FAT。

```
public void saveToFAT(FCB fcb)
{
    if(this.top<fcb.getBlockNeeded())//如果空闲块不够
    {
        JOptionPane.showMessageDialog( parentComponent: null, message: "空间已满！");
    }

    else//有空闲块
    {
        fcb.setFileStartBlock(getFreeBlock());//从空闲栈中获取新块

        if(fcb.getFileSize()<=Constants.BYTE_PER_BLOCK)//如果文件大小小于块大小
        {
            setEndBlock(fcb.getFileStartBlock());
        }

        else if(fcb.getFileSize()>Constants.BYTE_PER_BLOCK)//如果文件大小大于块大小
        {
            int blockSum=fcb.getBlockNeeded();
            int startBlock=fcb.getFileStartBlock();

            for(int i=0;i<blockSum-1;i++)
            {
                setNextBlock(startBlock,getFreeBlock());
                startBlock=getNextBlock(startBlock);
            }
            setEndBlock(startBlock);
            top--;
        }
    }

}
```

删除文件时，获取文件的开始块，并通过FAT的链接获取文件所在的所有块，将它们压入空闲块栈中。删除文件夹时，则递归删除文件夹中所有内容，释放文件夹中所有文件所在的FAT块。

```
//删除某个文件所占全部块
public void deleteFromFAT(int startBlockNum)
{
    int index=startBlockNum;
    int num=0;
    if(isEndBlock(index))
    {
        freeBlock(index);
    }

    else
    {
        while(!isEndBlock(index))
        {
            num=index;
            index=getNextBlock(index);
            freeBlock(num);
        }

        freeBlock(index);
    }

}
```

注意文件夹不占空间，不在FAT中表示。
为了方便操作，我在文件被修改时，先从FAT中删除文件，再将修改过的新文件所占空间加入FAT。

（2）FCB结构

文件夹和文件的结构都可以用FCB表示，它们之间用FCB_TYPE的枚举类型来区分。文件与父文件夹间通过folderFCB和childrenList进行链接。

```
class FCB implements Serializable
{
    private String fileName;//文件名
    private String innerText;//文件内容
    private FCB_TYPE fcbType;//是文件还是文件夹

    private FCB folderFCB;//文件夹的FCB
    public LinkedList<FCB> childrenList;//如果是文件夹，则存放文件夹内的文件或文件夹

    private int fileSize;//文件大小(占多少个字节)
    private int fileStartBlock;//开始盘块
    private int blockSum;//所占盘块数

    private String createTime;//创建时间
    private String modifyTime;//修改时间
```

（3）树形目录结构映射

用java swing的JTree控件构造树形目录的图形化界面。将swing的DefaultMutableTreeNode对象与FCB对象的映射放入map中，以便用户在选取图形化目录的某个树节点时，系统可以获取树节点对应的FCB，从而进行系统内部的操作。

```
//目录面板
class catalogPanel extends JPanel implements Serializable
{
    private fileSystem system;
    private JScrollPane catalogPane;
    private DefaultMutableTreeNode root = new DefaultMutableTreeNode( userObject: "root");//根目录作为根结点
    private DefaultTreeModel model=new DefaultTreeModel(root);
    private JTree catalogTree=new JTree(model);

    private Map<DefaultMutableTreeNode,FCB> nodeFCBMap;//建立树结点到FCB的映射

    public FCB selectedFCB;
    public DefaultMutableTreeNode selectedNode;
```

（4）逻辑目录结构

```
class fileSystem implements Serializable
{
    private LinkedList<FCB> fileCatalog=null;//目录
    private FCB fileRoot;//根目录
    private FAT fatList;//fat表
```

逻辑目录结构为存放FCB的一个链表，它的声明位于fileSystem类中。文件被删除时，链表中的文件对应的FCB也会被删除。文件被创建时，文件的FCB加入链表。

6.项目优势与劣势

采用链接的存储方式，没有外部碎片，空闲空间栈中的任何块都可以满足请求。创建文件时不需要说明文件大小，只要有空闲块，文件就可以增大，无需合并空间。

这次项目本来打算用java的Serializable接口，在打开窗口时从磁盘中读取文件系统对象，关闭窗口时向磁盘中写入文件系统对象，来实现文件系统的本地存储，但由于时间匆忙，有一个bug没有改掉，因此本地存储未能成功实现，用注释的方式放在了代码中。