

# 文件管理项目——文件系统模拟

---

## 文件管理项目——文件系统模拟

- 项目需求

  - 基本任务

  - 功能描述

  - 项目目的

- 开发环境

- 操作说明

- 系统分析

  - 磁盘

  - 索引表示法

  - 位示图

- 系统设计

  - 类设计

  - 界面设计

- 系统实现

  - 磁盘

    - 创建文件

    - 读取文件内容

    - 更新文件内容

    - 文件重命名

    - 删除文件

  - 目录

    - 创建目录（创建文件夹）

    - 显示目录

    - 返回上一级目录

    - 判断重名

    - 格式化

  - 持久化

    - 记录目录信息

    - 记录位示图信息

    - 记录磁盘信息

    - 根据日志信息加载系统

- 功能展示

  - 新建文件/文件夹

  - 编辑文件

  - 删除文件/文件夹

  - 文件重命名

  - 进入下一层目录

  - 返回上一层目录

  - 格式化

学号：1850250

姓名：赵浠明

班级：操作系统1班

教师：张慧娟

---

## 项目需求

---

### 基本任务

#### 基本要求

- 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
- 退出这个文件系统时，需要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

#### 具体要求

- 文件存储空间管理可采取显式链接（如FAT）或者其他方法
- 空闲空间管理可采用位图或者其他方法
- 文件目录采用多级目录结构，目录项目中应包含：文件名、物理地址、长度等信息

### 功能描述

- 文件
  - 创建文件
  - 打开文件
  - 关闭文件
  - 读写文件
  - 删除文件
  - 重命名文件
- 磁盘
  - 创建子目录
  - 删除子目录
  - 显示目录
  - 更改当前目录
  - 格式化
- 持久化

- 退出系统时保存相关信息
- 进入系统时加载已有信息

## 项目目的

- 理解文件存储空间的管理
- 掌握文件的物理结构、目录结构和文件操作
- 实现简单文件系统管理
- 加深文件系统实现过程的理解

## 开发环境

---

- 操作系统平台: Windows 10
- 开发语言: Java SE
  - JDK版本: jdk1.8.0\_152
- 开发软件: Eclipse IDE 2020-03 (4.15.0)

## 操作说明

---

!!! 注意: 请确保.exe文件与info文件夹处于同一目录下, 同时请勿修改、删除info文件夹中的内容, 否则会影响程序的运行及可持久化

### 1. 创建文件/文件夹

按下主界面中相应的按钮, 在弹出界面中输入文件名称即可

### 2. 返回上一层目录

按下主界面中“返回上层目录”按钮即可, 若已在最上层目录则按钮无效

### 3. 格式化

按下主界面中“格式化”按钮即可, 操作后系统内所有信息将被清空, 请慎重选择

### 4. 删除文件/文件夹

**右键单击**需删除的文件、文件夹, 选择“删除”

### 5. 重命名文件/文件夹

**右键单击**需删除的文件、文件夹, 选择“重命名”, 并在弹出界面中输入文件名称

### 6. 进入下一层目录

**左键单击**需进入的文件夹

### 7. 读写文件

**左键单击**需读写的文件, 将弹出文件的输入界面, 可查看文件具体内容并进行修改

# 系统分析

---

## 磁盘

本文件模拟系统中，磁盘分为大小相同的块进行存储，默认虚拟磁盘的总空间为**2000B**，一个磁盘块的大小为**8B**。

## 索引表示法

本文件模拟系统中，磁盘存储空间管理将使用**索引表示**的方法。创建文件时，将分配磁盘的一个块存储文件的**索引表**；文件内容可离散存储于磁盘不同的块中，索引表的各项依次指向文件具体内容所在位置。

## 位示图

本文件模拟系统中，磁盘空闲空间管理将使用**位示图**的方法，空闲的磁盘块在位示图中使用 `false` 表示，已使用的磁盘块在位示图中使用 `true` 表示。

# 系统设计

---

## 类设计

### 磁盘类 MyDisk

在内存中开辟一个空间作为系统的存储空间，模拟磁盘上的文件管理

```
public class MyDisk {  
  
    /** 磁盘可视化 */  
    MainFrame frame;  
  
    /** 磁盘块的大小，设置为8个字节 */  
    private int blockSize = 8;  
  
    /** 磁盘大小，即磁盘中的物理块的数目 */  
    public int size;  
  
    /** 磁盘空闲块的数目 */  
    public int free;  
  
    /** 位示图，反应磁盘空间的使用情况 */  
    public boolean[] bitMap;  
  
    /** 磁盘内容，用一个String数组来模拟磁盘的块 */  
}
```

```
public String[] memory;

/** 文件目录 */
public MyCatalog catalog;

/** 根据参数中需要的磁盘大小，在内存中开辟相应空间 */
public MyDisk(int sz);

/** 根据文件信息初始化 */
public void initial();

/** 创建文件或文件夹 */
public boolean createFile(String name, int type);

/** 检查当前目录是否有相同名字的文件或文件夹 */
private boolean isLegal(String name, int type);

/** 设置文件或文件夹的名称 */
public boolean setFileName(FCB file, String name);

/** 更新文件 */
public boolean updateFile(FCB file, String content);

/** 计算文件所需磁盘块数 */
private int blockNum(int byteNum);

/** 顺序查找第一个空闲块的地址 */
private int findFreeBlock();

/** 释放一个磁盘块 */
private void releaseBlock(int index);

/** 删除文件 */
public void deleteFile(FCB file);

/** 释放文件空间 */
public void release(FCB file);

/** 更新页面上显示的目录 */
public void refreshCatalog();

/** 格式化 */
public void clearAll();

/** 更新当前空闲空间 */
```

```
public void setFreeBlockNum();

/** 设置可视化窗口 */
public void setFrame(MainFrame frame);
}
```

## 目录类 MyCatalog

存放文件模拟系统的目录信息

```
public class MyCatalog {
    /** 目录根节点 */
    Node root;

    /** 当前目录 */
    Node cur;

    /** 磁盘 */
    MyDisk disk;

    public MyCatalog();

    /** 添加节点 */
    public void add(FCB fcb);

    /** 删除节点 */
    public void delete(FCB file);

    /** 递归删除子目录 */
    private void erase(Node p, int depth);

    /** 根据子女节点的编号设置当前节点 */
    public void setCurrentCatalog(int index);

    /** 在当前目录下查找文件 */
    public FCB findFile(String name);

    /** 返回上层目录 */
    public boolean returnToUpper();

    /** 获取当前目录路径 */
    public String getCurrentPath();

    /** 清空目录 */
    public void clear();
}
```

```

    /** 将目录写入文件 */
    public boolean writeToFile();

    /** 递归写入 */
    private void write(PrintWriter pw, Node p);

    /** 从文件读取目录 */
    public void readFromFile();

    /** 递归读取 */
    private Node read(BufferedReader br, Node parent);
}

```

## 文件类 FCB

存储一个文件的具体信息，实现文件信息的封装

```

public class FCB {

    /** 文件名 */
    public String name;

    /** 文件类型 */
    int type;

    /** 两种文件类型：文本文件、文件夹 */
    public static final int TXT = 1;

    public static final int FOLDER = 2;

    /** 文件索引表地址 */
    int start;

    /** 文件当前长度 */
    int curLength = 0;

    /** 文件最大允许长度 */
    int maxLength;

    /** 文件创建时间 */
    Date dateCreated;

    /** 文件修改时间 */
    Date dateUpdated;
}

```

```

    /** 文件路径 */
    String path;

    /** 父文件 */
    FCB parent;

    /** 文件读写权限 */
    public boolean protection = false;

    /** 文件所在磁盘 */
    MyDisk disk;

    FCB();

    /** 获取文件文本内容 */
    public String getContent();

    /** 设置文件路径 */
    void setPath(String path);
}

```

## 主界面 MainFrame

继承自java.swing.JFrame，实现系统的主界面显示及相关交互功能

```

public class MainFrame extends JFrame {

    /** 虚拟磁盘 */
    MyDisk disk;

    private JPanel contentPane;

    MainFrame self;

    /** 文件表格 */
    public DefaultTableModel fileTableModel;

    private JTable fileTable;

    /** 路径标签 */
    private JLabel pathJl;

    /** 剩余空间标签 */
    public JLabel freeJl;
}

```



```

        public MainFrame(String title);

        /** 设置窗口显示当前路径 */
        public void setPath(String path)
    }

```

## 创建文件界面 CreateDialog

继承自java.swing.JDialog, 实现创建文件（文件夹）的交互

```

public class CreateDialog extends JDialog {

    private final JPanel contentPanel = new JPanel();

    private CreateDialog self;

    MainFrame parent;

    private JTextField nameText;

    private final JButton enterJb;

    public CreateDialog(int type, MainFrame parent);
}

```

## 文件输入界面 InputDialog

继承自java.swing.JDialog, 可查看文件内容并进行修改

```

public class InputDialog extends JDialog {

    private final JPanel contentPanel = new JPanel();

    private InputDialog self;

    /** 父界面 */
    MainFrame parent;

    /** 对应输入文本信息的文件 */
    FCB file;

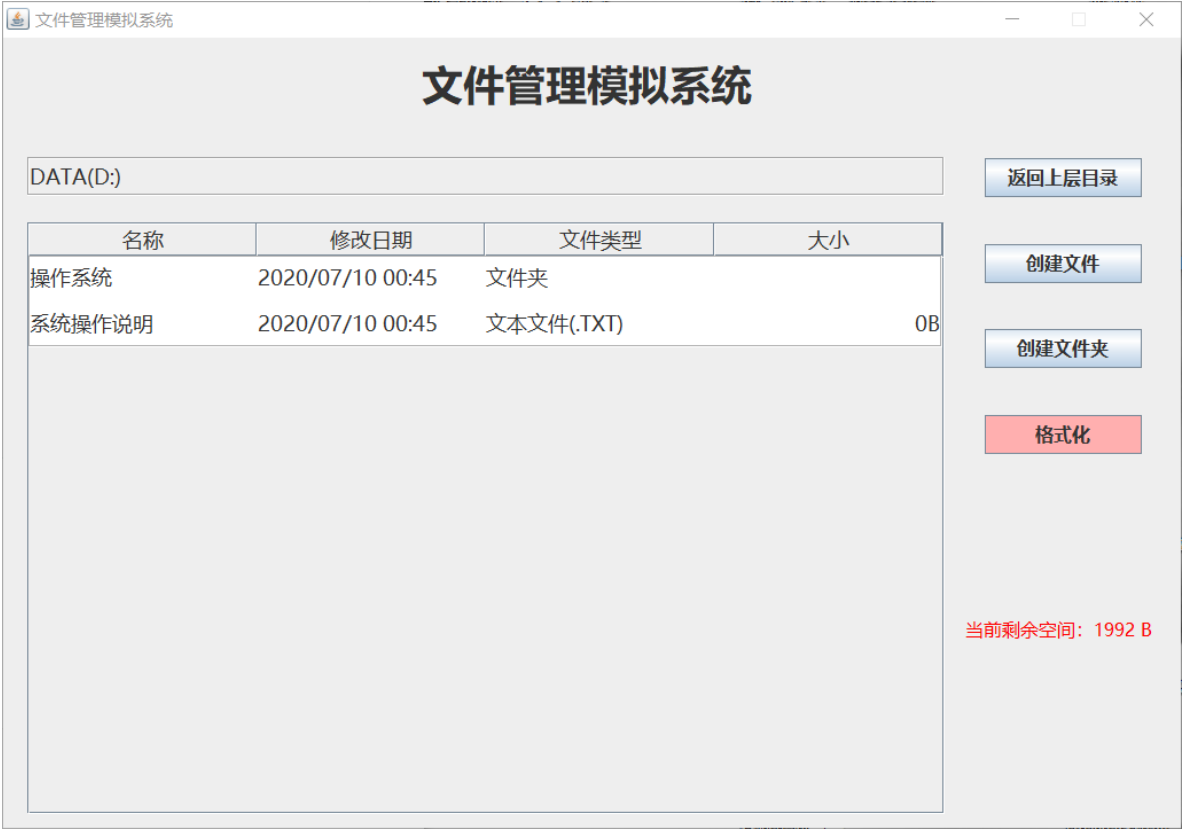
    public InputDialog(FCB file, MainFrame parent);
}

```

# 界面设计

## 主界面

- 标题
- 当前路径显示
- 当前目录下文件显示（名称，修改日期，文件类型，大小）
- 按钮
  - 返回上层目录
  - 创建文件
  - 创建文件夹
  - 格式化
- 当前剩余空间显示



## 文件创建界面

创建新文件

请输入新文件的名称:

确认

## 文件输入界面

文件管理模拟系统

DATA(D:)

名称	修改日期	文件类型	大小
操作系统	2020/07/10 00:45	文件夹	
系统操作说明	2020/07/10 00:45	系统操作说明	

test

保存

返回上层目录

创建文件

创建文件夹

格式化

当前剩余空间: 1992 B

## 文件重命名界面

重命名

请输入更改后的名称:

确认

## 系统实现

# 磁盘

## 创建文件

- 点击“创建文件”按钮，弹出创建文件界面
- 输入文件的名称，并点击“确认”
- 调用 MyDisk 的 `createFile()` 函数创建新文件
  - 检查磁盘是否有空闲空间
  - 检查是否有重名文件
  - 在目录中添加相应的项

```
public boolean createFile(String name, int type) {
    /* 磁盘是否有空闲空间 */
    if (free == 0) {
        return false;
    }

    /* 检查有无重名文件 */
    String temp = name;
    int count = 1;
    while (!isLegal(temp, type)) {
        temp = name;
        temp += "(" + count + ")";
        count++;
    }
    name = temp;

    /* 在目录中添加相应项，为文件分配一个块 */
    if (type == FCB.TXT)
        catelog.add(new FCB(name, type, this, findFreeBlock(),
        catelog.cur.fcb));
    else
        catelog.add(new FCB(name, type, this, -1, catelog.cur.fcb));
    setFreeBlockNum();
    return true;
}
```

- 更新界面的目录显示及剩余空闲空间显示

## 读取文件内容

- 单击需读取的文件
- 调用该文件的 `getContent()` 函数，获取文件具体信息

- 根据文件属性 `start`，获取文件索引表的位置

```
char[] ch = disk.memory[start].toCharArray();
```

- 根据索引表顺序获得文件的具体内容

```
for (int i = 0; i < curLength; i++) {  
    String cur = disk.memory[(int) ch[i]];  
    content += cur;  
}
```

- 去掉内容中尾部的空格

```
int i = content.length() - 1;  
if (i < 0) {  
    return content;  
}  
while (content.charAt(i) == ' ') {  
    i--;  
}  
content = content.substring(0, i + 1);
```

- 将文件内容显示在输入界面的文本框中

```
textArea.setText(file.getContent());
```

## 更新文件内容

- 单击需更新的文件，读取文件具体信息
- 输入文件内容，并按下“保存”键
- 调用文件的 `updateFile()` 函数，更新文件信息
  - 检查空间是否足够
  - 释放原先的磁盘块

```
/* 释放磁盘块 */  
char[] index = memory[file.start].toCharArray();  
for (int i = 0; i < file.curLength; i++) {  
    releaseBlock((int) index[i]);  
}
```

- 将信息写入磁盘中，同时更新索引表信息

```
int i = 0;
```

```

file.curLength = 0;
index = new char[blockSize];
char[] temp = new char[blockSize];

while (block > 0) {
    int j = findFreeBlock();

    /* 更新索引 */
    index[file.curLength++] = (char) j;
    /* 将内容写入块内 */
    Arrays.fill(temp, ' ');
    System.arraycopy(ch, i, temp, 0, ch.length - i >= blockSize
? blockSize : ch.length - i);
    memory[j] = String.valueOf(temp);
    i += blockSize;
    block--;
}
memory[file.start] = String.valueOf(index);

```

- 更新文件修改时间

```
file.dateUpdated = new Date(System.currentTimeMillis());
```

- 更新界面的目录显示及剩余空闲空间显示

## 文件重命名

- 右键单击需重命名的文件
- 弹出重命名窗口，输入文件名称
- 调用磁盘的 `setFileName()` 函数，更改文件名称

```

/** 设置文件或文件夹的名称 */
public boolean setFileName(FCB file, String name) {
    if(!isLegal(name, file.type)) {
        return false;
    }
    file.name = name;
    refreshCatalog();
    return true;
}

```

- 更新界面的目录显示及剩余空闲空间显示

## 删除文件

- 右键单击需删除的文件（如果删除的是文件夹，则同时删除文件夹中的所有内容）
- 调用磁盘的 `deleteFile()` 函数删除文件
  - 在目录中删除对应目录项

```
/** 删除节点 */
public void delete(FCB file) {
    Node p1 = cur.leftChild;
    Node p2 = p1.nextSibling;

    /* 修改前驱结点的指针 */
    if (p1.fcb == file) {
        cur.leftChild = p2;
        erase(p1);
    } else {
        while (p2 != null) {
            if (p2.fcb == file) {
                p1.nextSibling = p2.nextSibling;
                erase(p2);
                break;
            }
            p1 = p2;
            p2 = p1.nextSibling;
        }
    }
}

/** 递归删除子目录 */
private void erase(Node p, int depth) {

    if (p.leftChild != null) {
        /* 当前结点是文件夹 */
        erase(p.leftChild, depth + 1);
        if (p.nextSibling != null && depth > 0)
            erase(p.nextSibling, depth + 1);
    } else {
        if (p.nextSibling != null && depth > 0)
            erase(p.nextSibling, depth + 1);
    }

    /* 释放空间 */
    if (p.fcb.type == FCB.TXT) {
        disk.release(p.fcb);
    }
}
```

```
p = null;
}
```

- 释放内存空间

```
public void release(FCB file) {
    char[] index = memory[file.start].toCharArray();
    for (int i = 0; i < file.curLength; i++) {
        releaseBlock(index[i]);
    }
    releaseBlock(file.start);
}

/** 释放一个磁盘块 */
private void releaseBlock(int index) {
    bitMap[index] = false;
    free++;
}
```

- 更新界面的目录显示及剩余空闲空间显示

## 目录

### 创建目录（创建文件夹）

- 点击“创建文件夹”按钮，弹出创建文件界面
- 输入文件夹的名称，并点击“确认”
- 调用 MyDisk 的 createFile() 函数创建新文件夹
  - 检查磁盘是否有空闲空间
  - 检查是否有重名文件夹
  - 在目录中添加相应的项

代码部分和创建文件相同

### 显示目录

- 显示当前目录路径

```
/* 更新路径显示 */
frame.setPath(cateLog.getCurrentPath());
```

- 显示当前目录下的所有文件信息（名称、修改时间、文件类型、大小）



```

/* 更新目录显示 */
frame.fileTableModel.setRowCount(0);
Node p = catelog.cur.leftChild;
while (p != null) {
    Object[] temp = { p.fcb.name, myFmt.format(p.fcb.dateUpdated),
p.fcb.type == FCB.TXT ? "文本文件(.TXT)" : "文件夹", p.fcb.type ==
FCB.TXT ? p.fcb.curLength * 8 + "B" : "" };
    frame.fileTableModel.addRow(temp);
    p = p.nextSibling;
}

```

## 返回上一级目录

- 左键单击“返回上级目录”按钮
- 调用目录的 `returnToUpper()` 函数，返回上一级目录

```

/** 返回上层目录 */
public boolean returnToUpper() {
    if (cur.parent == null) {
        System.out.println("已是最上层目录");
        return false;
    } else {
        cur = cur.parent;
        return true;
    }
}

```

- 更新界面的目录显示

## 判断重名

- 对应同一目录下的文件，文件与文件夹的重名需要分开判断（即可以存在重名的一个文件和一个文件夹）

```

private boolean isLegal(String name, int type) {
    try {
        Node p = catelog.cur.leftChild;
        while (p != null) {
            if (p.fcb.name.equals(name) && p.fcb.type == type) {
                return false;
            }
            p = p.nextSibling;
        }
    } catch (NullPointerException e) {
        e.printStackTrace();
    }
}

```

```
}  
    return true;  
}
```

## 格式化

- 左键单击“格式化”按钮
- 调用磁盘的 `clearAll()` 函数，清空系统内容
  - 重置位示图及磁盘空间

```
free = size;  
bitMap = new boolean[size];  
for (boolean b : bitMap) {  
    b = false;  
}  
memory = new String[size];  
Arrays.fill(memory, "");
```

- 清空目录

```
root = new Node(new FCB(), null);  
root.fcb.setPath("DATA(D:)");  
cur = root;
```

- 更新界面的目录显示及剩余空闲空间显示

## 持久化

### 记录目录信息

- 将目录写入 `catelogInfo.txt` 文件中

```
/** 将目录写入文件 */  
public boolean writeToFile() throws IOException {  
    OutputStream os = new FileOutputStream("info/catelogInfo.txt");  
    PrintWriter pw = new PrintWriter(os);  
  
    write(pw, root);  
    pw.close();  
    os.close();  
    return true;  
}  
  
/** 递归写入 */
```

```

private void write(PrintWriter pw, Node p) {
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm");

    FCB fcb = p.fcb;
    pw.println(fcb.name);
    pw.println(fcb.type);
    pw.println(fcb.start);
    pw.println(fcb.dateCreated == null ? null :
dateFormat.format(fcb.dateCreated));
    pw.println(fcb.dateUpdated == null ? null :
dateFormat.format(fcb.dateUpdated));
    pw.println(fcb.curLength);
    pw.println(p.leftChild == null ? 0 : 1);
    pw.println(p.nextSibling == null ? 0 : 1);

    if (p.leftChild != null) {
        write(pw, p.leftChild);
    }
    if (p.nextSibling != null) {
        write(pw, p.nextSibling);
    }
}
}

```

## 记录位示图信息

- 将位示图写入bitmapInfo.txt文件中

```

/* 保存位示图信息 */
OutputStream bitos = new FileOutputStream("info/bitmapInfo.txt");
PrintWriter bitpw = new PrintWriter(bitos);
for (int i = 0; i < disk.size; i++) {
    bitpw.println(disk.bitMap[i]);
}
bitpw.close();
bitos.close();

```

## 记录磁盘信息

- 将磁盘内容写入memoryInfo.txt文件中

```

/* 保存内存信息 */
OutputStream memoryos = new
FileOutputStream("info/memoryInfo.txt");
PrintWriter memorypw = new PrintWriter(memoryos);
for (int i = 0; i < disk.size; i++) {
    memorypw.println(disk.memory[i]);
}
memorypw.close();
memoryos.close();

```

## 根据日志信息加载系统

- 开始运行程序时，调用磁盘的 `initial()` 函数，根据 `bitmapInfo.txt`, `memoryInfo.txt`, `catelogInfo.txt` 中的信息进行系统的初始化
  - 读取位示图

```

BufferedReader bitbr = new BufferedReader(new
FileReader("info/bitmapInfo.txt"));
String line = null;
int i = 0;
while((line = bitbr.readLine()) != null) {
    bitMap[i] = ("true".equals(line));
    if(bitMap[i] == true)
        free--;
    i++;
}
bitbr.close();

```

- 读取内存信息

```

BufferedReader memorybr = new BufferedReader(new
FileReader("info/memoryInfo.txt"));
line = null;
i = 0;
while((line = memorybr.readLine()) != null) {
    memory[i] = line;
    i++;
}
memorybr.close();

```

- 读取目录信息

```

public void readFromFile() throws IOException, ParseException
{

```

```

        BufferedReader bitbr = new BufferedReader(new
FileReader("info/cateLogInfo.txt"));
        root = read(bitbr, null);
        cur = root;
        bitbr.close();
    }

    /* 递归读取 */
    private Node read(BufferedReader br, Node parent) throws
IOException, ParseException {
        Node p = null;
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-
MM-dd HH:mm");

        String name = br.readLine();
        int type = "1".equals(br.readLine()) ? FCB.TXT :
FCB.FOLDER;
        int start = Integer.parseInt(br.readLine());
        String temp = br.readLine();
        Date dateCreated = "null".equals(temp) ? null :
dateFormat.parse(temp);
        temp = br.readLine();
        Date dateUpdated = "null".equals(temp) ? null :
dateFormat.parse(temp);
        int curLen = Integer.parseInt(br.readLine());
        int left = Integer.parseInt(br.readLine()), right =
Integer.parseInt(br.readLine());

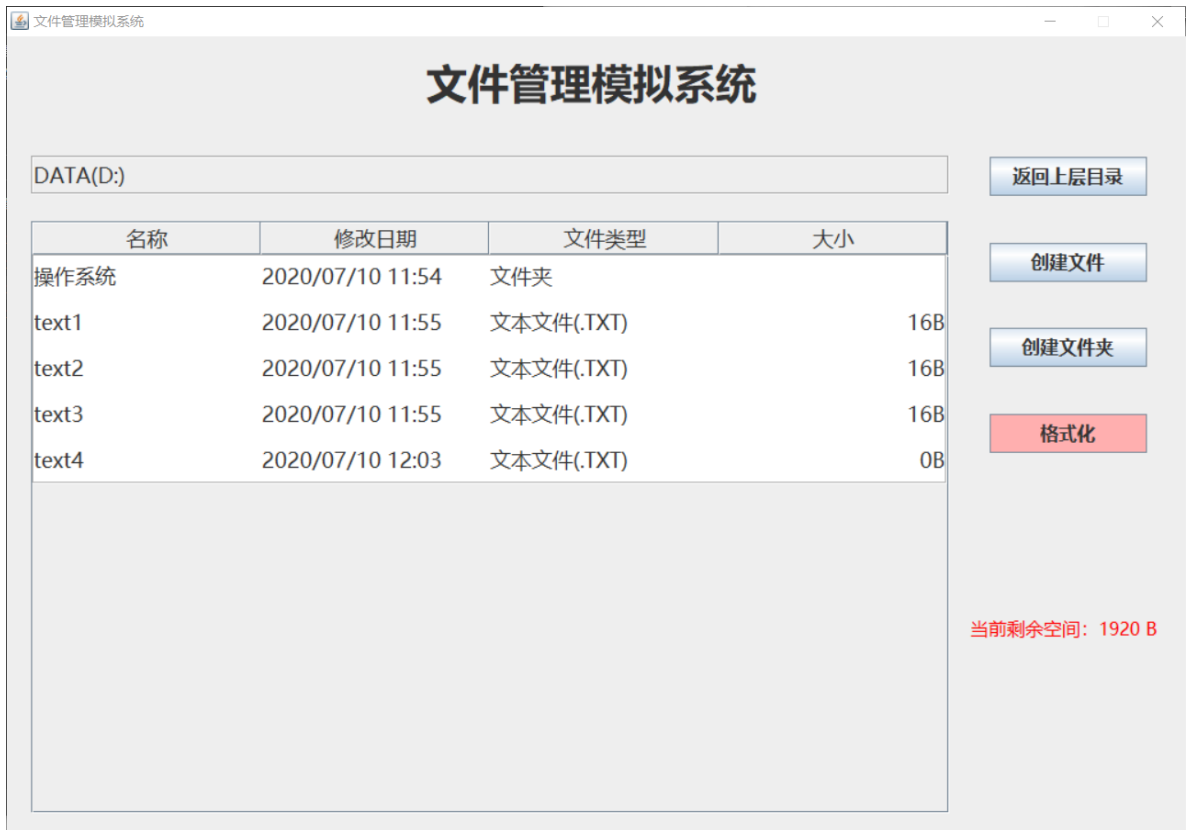
        if (parent == null) {
            p = new Node(new FCB(), null);
            p.fcb.setPath("DATA(D:)");
        } else {
            p = new Node(new FCB(name, type, disk, start, parent.fcb,
dateCreated, dateUpdated, curLen), parent);
        }
        if (left == 1) {
            p.leftChild = read(br, p);
        }
        if (right == 1) {
            p.nextSibling = read(br, parent);
        }
        return p;
    }
}

```

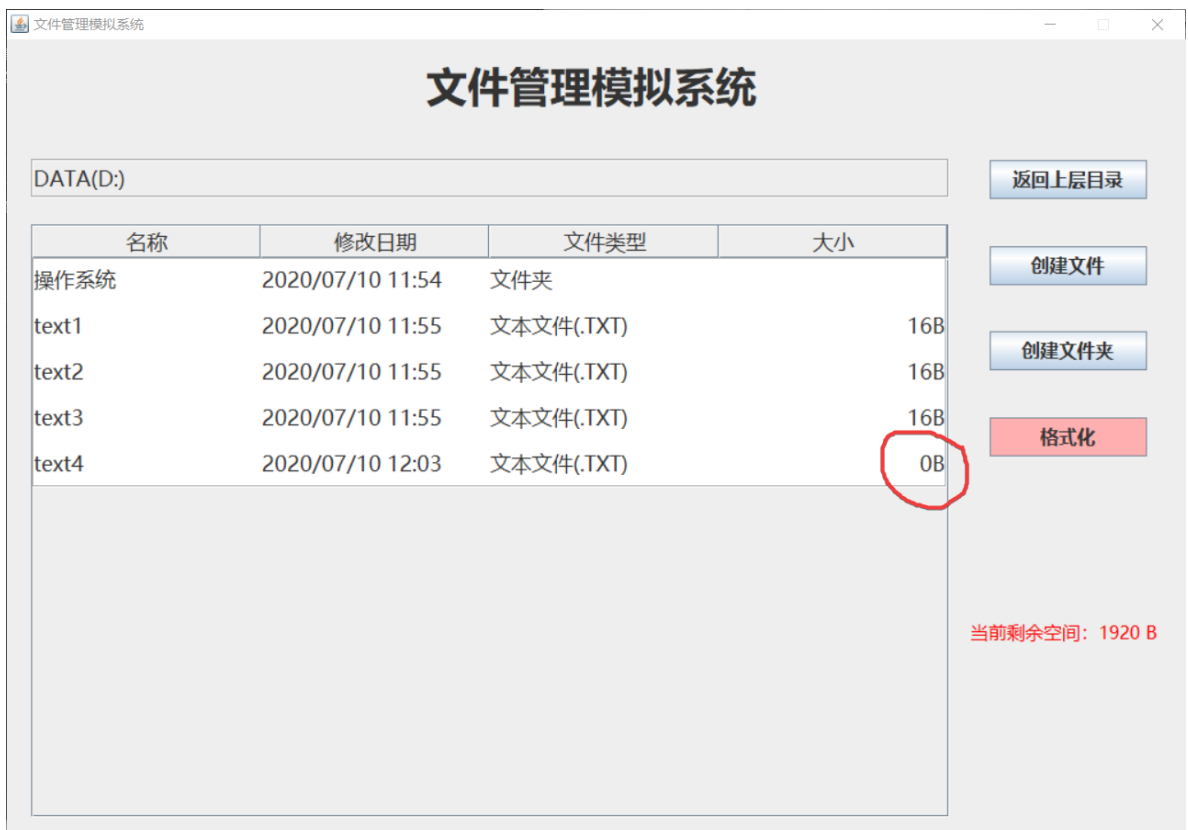
# 功能展示

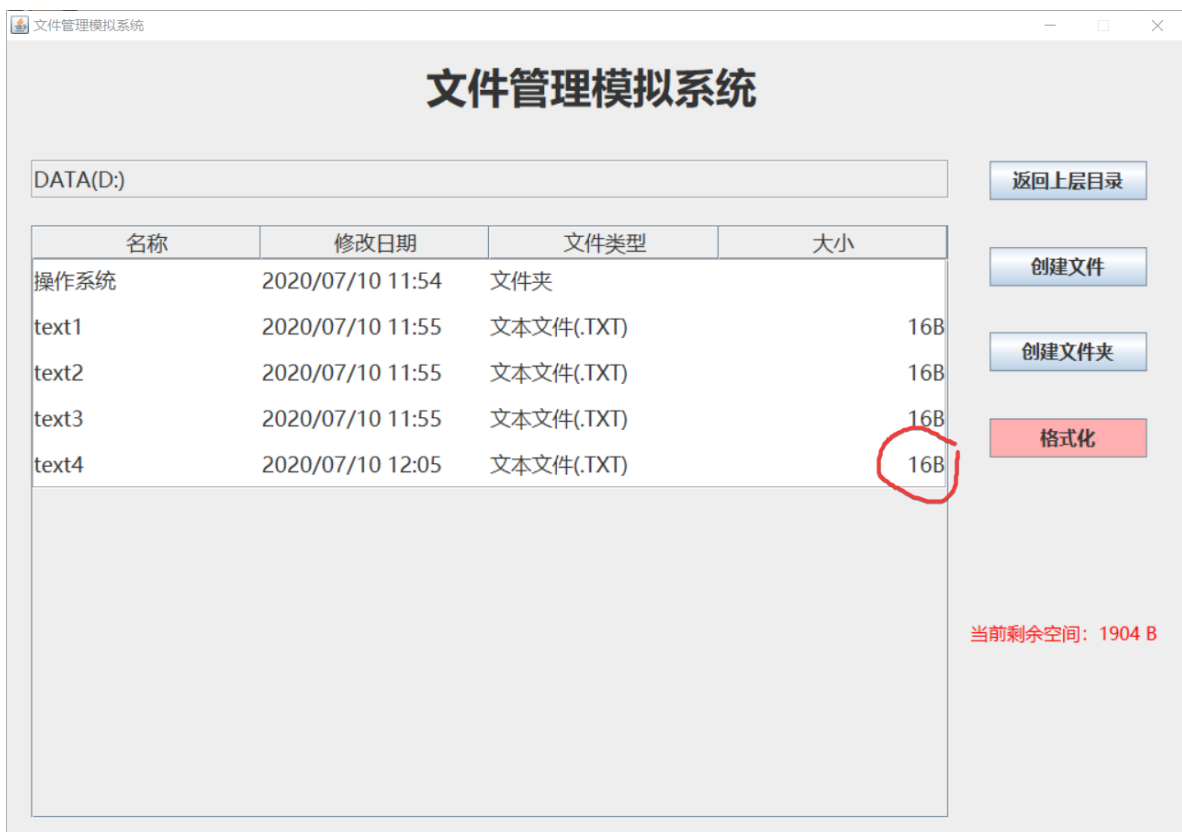
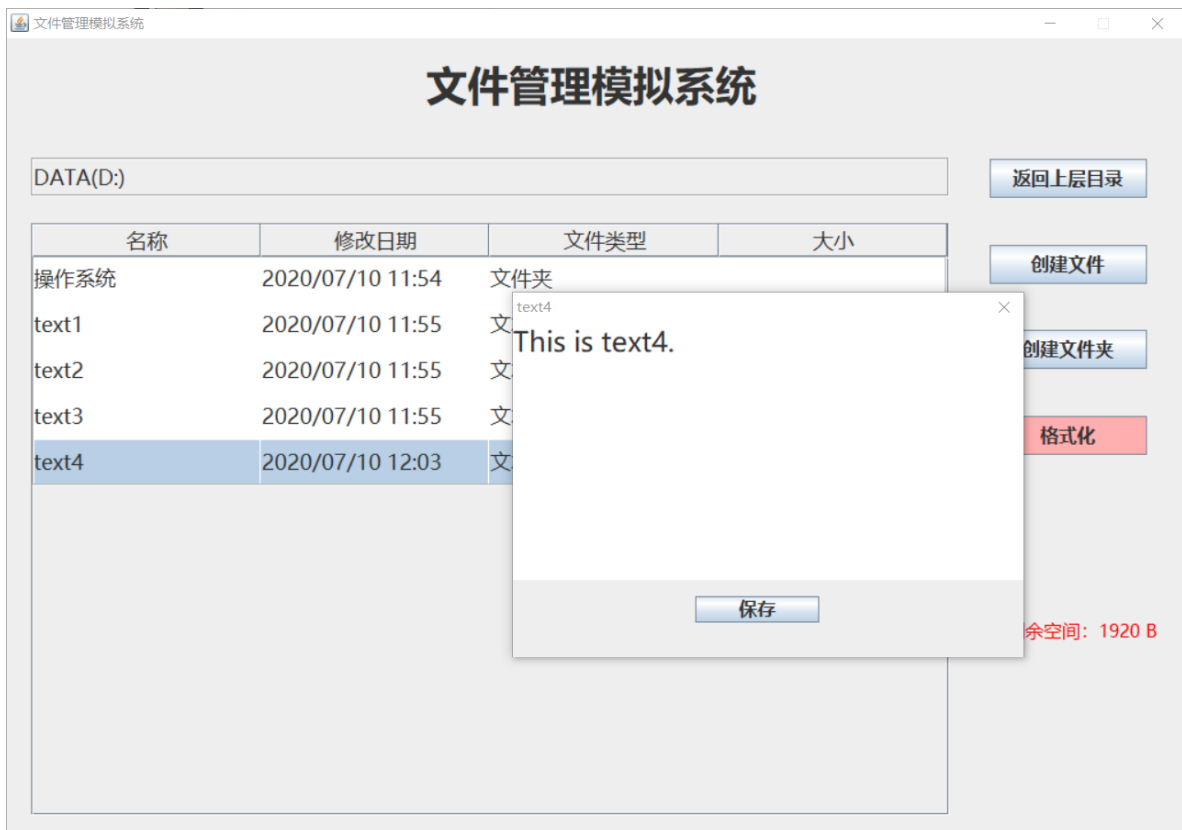
## 新建文件/文件夹





## 编辑文件





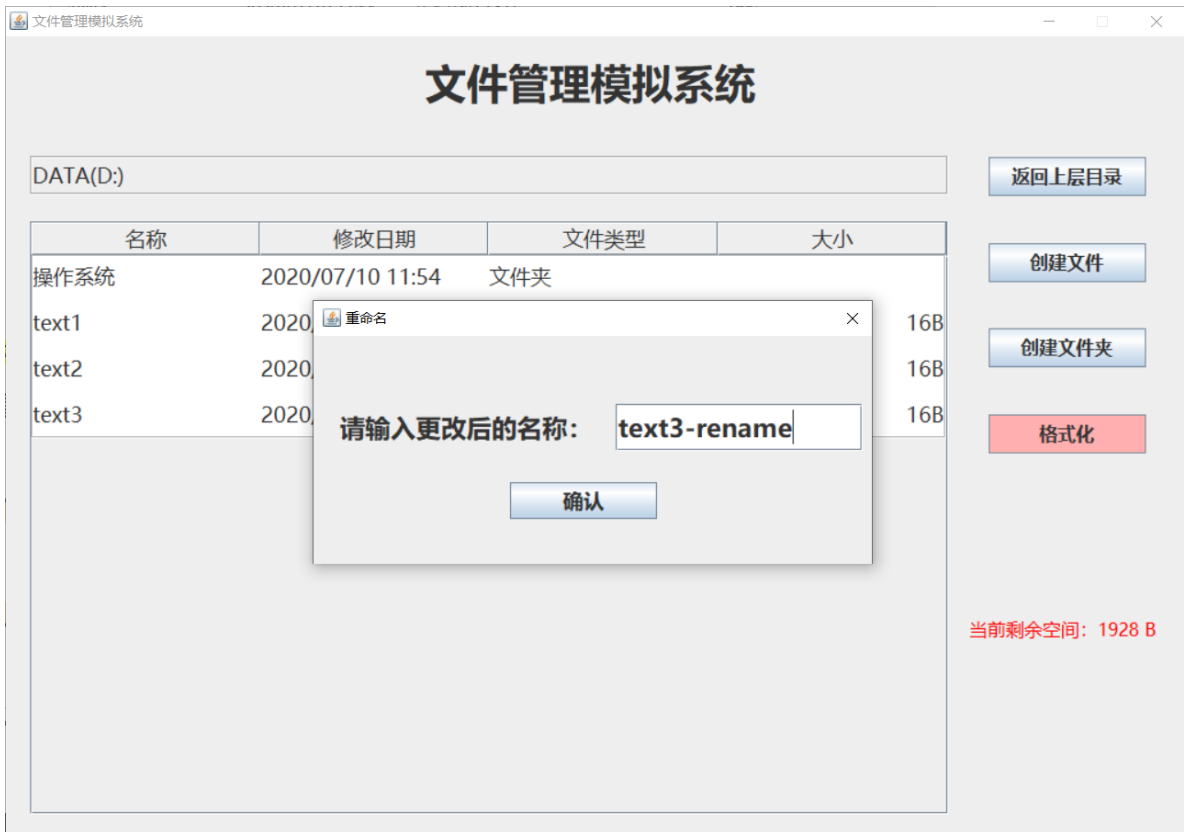
## 删除文件/文件夹

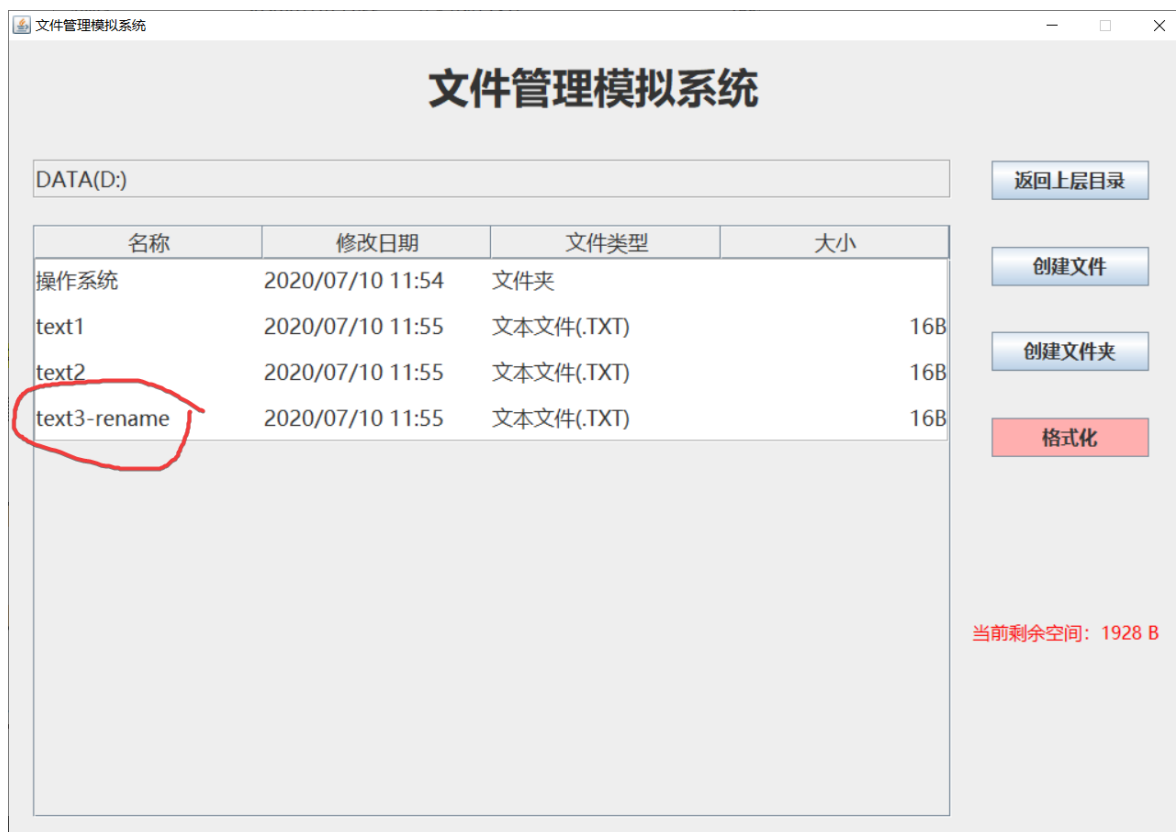




删除的文件大小为16B，其索引表还占了8B，因此删除文件后剩余空间多了24B

## 文件重命名

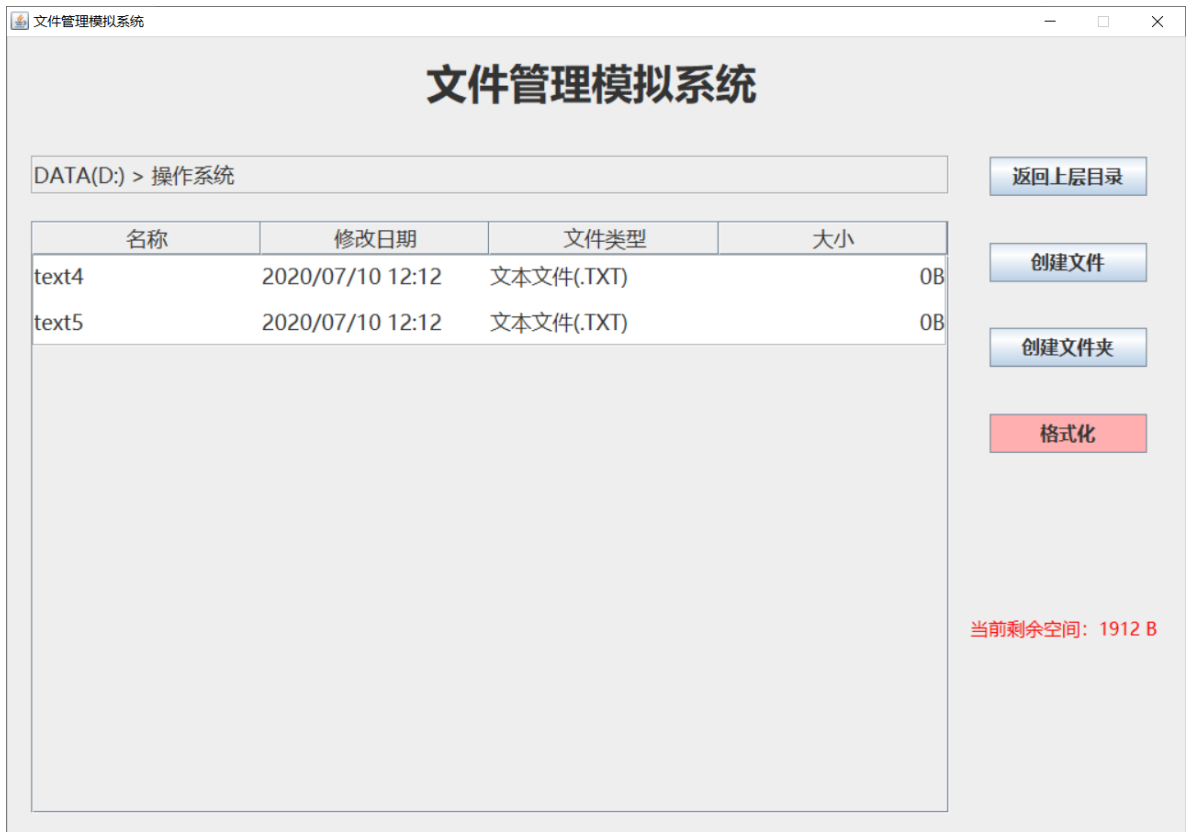




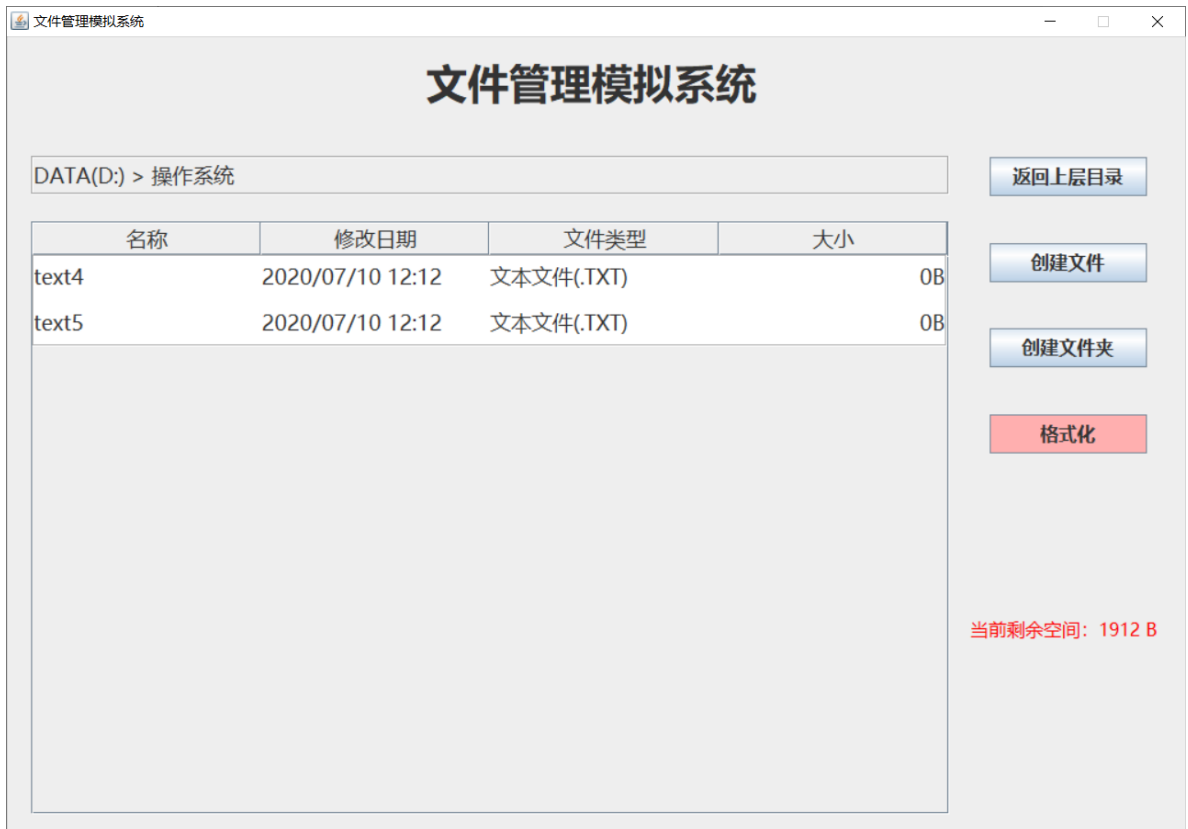
## 进入下一层目录



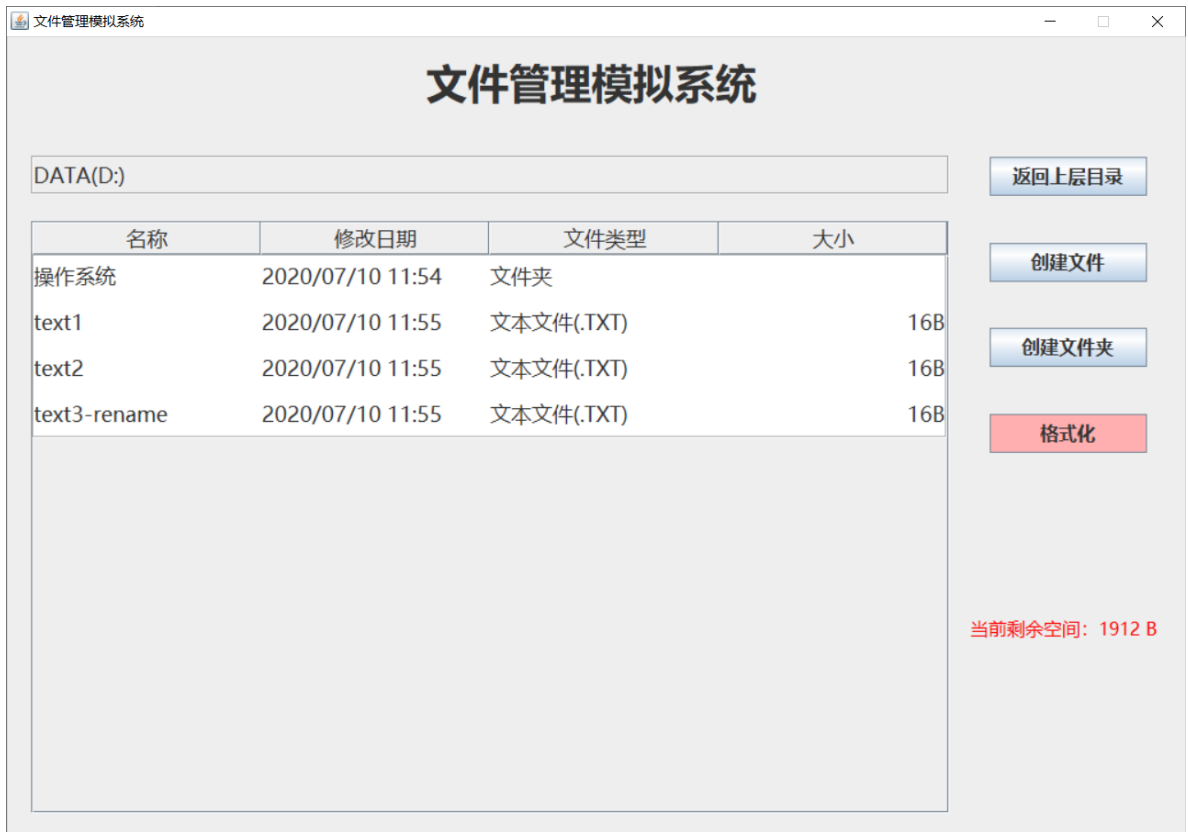
左击“操作系统”文件夹->



## 返回上一层目录



点击“返回上层目录”按钮->

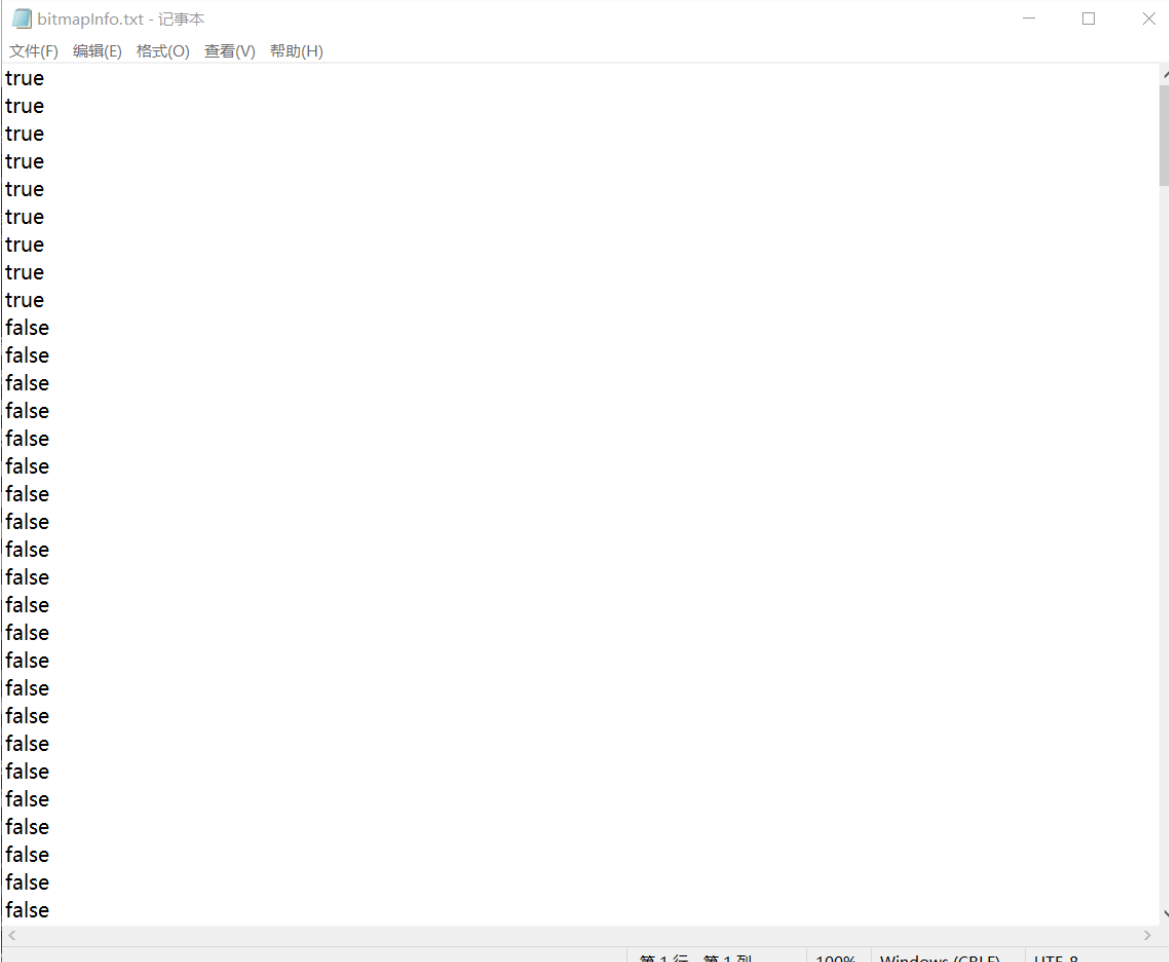


## 格式化

点击“格式化”按钮->



## 退出时记录系统信息



bitmapInfo.txt - 记事本


文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```

true
true
true
true
true
true
true
true
true
false
false
false
false
false
false
false
false
false
false
false

```

< 第 1 行, 第 1 列 100% Windows (CRLF) UTF-8 >

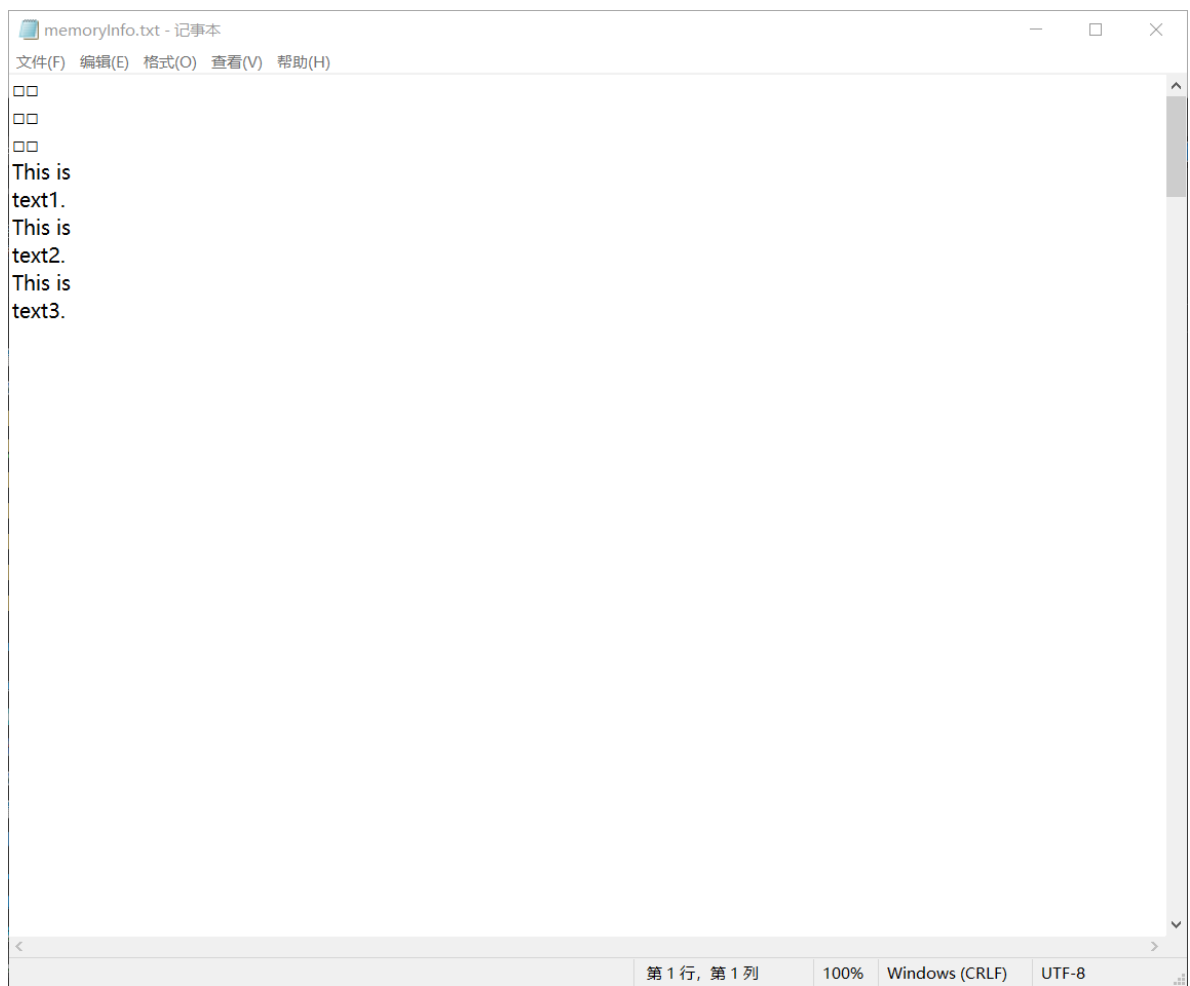


catelogInfo.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
null
0
0
null
null
0
1
0
操作系统
2
-1
2020-07-10 12:24
2020-07-10 12:24
0
0
1
text1
1
0
2020-07-10 12:24
2020-07-10 12:24
2
0
1
text2
1
1
2020-07-10 12:24
2020-07-10 12:24
2
0
```

第 1 行, 第 1 列 100% Windows (CRLF) ANSI



由于memoryInfo中的索引表信息是以一个字节表示一个索引项，再转化为字符串形式，因此显示在txt文件中为乱码，但不影响重新打开系统的恢复