项目说明文档

操作系统

——文件系统管理

作 者 姓 名： 杨煜

学 号： 1850217

指 导 教 师： 张惠娟

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 项目背景 1](#_Toc43808628)

[1.1 项目简介 1](#_Toc43808629)

[1.2 项目目的 1](#_Toc43808630)

[1.3 项目要求 1](#_Toc43808631)

[2 需求分析 2](#_Toc43808632)

[2.1 格式化 2](#_Toc43808633)

[2.2 创建子目录 2](#_Toc43808634)

[2.3 删除子目录 2](#_Toc43808635)

[2.4 显示目录 2](#_Toc43808636)

[2.5 更改当前目录 2](#_Toc43808637)

[2.6 创建文件 2](#_Toc43808638)

[2.7 打开文件 2](#_Toc43808639)

[2.8 关闭文件 2](#_Toc43808640)

[2.9 删除文件 2](#_Toc43808641)

[2.10 保存 2](#_Toc43808642)

[2.11 导入 2](#_Toc43808643)

[3 数据结构 3](#_Toc43808644)

[3.1 树形目录结构 3](#_Toc43808645)

[3.2 位示图 3](#_Toc43808646)

[3.3 显式链接分配 3](#_Toc43808647)

[4 设计实现 3](#_Toc43808648)

[4.1 FAT类 3](#_Toc43808649)

[4.2 FCB类 3](#_Toc43808650)

[4.3 创建文本文件 4](#_Toc43808651)

[4.4 创建目录 6](#_Toc43808652)

[4.5 删除文本文件 9](#_Toc43808653)

[4.6 删除文件目录 11](#_Toc43808654)

[4.7 格式化 13](#_Toc43808655)

[4.8 打开文本文件 14](#_Toc43808656)

[4.9 保存与导入 17](#_Toc43808657)

[5附录 19](#_Toc43808658)

[5.1 文件说明 19](#_Toc43808659)

[5.2 开发环境 20](#_Toc43808660)

# 1 项目背景

## 1.1 项目简介

文件系统是操作系统用于明确存储设备或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：文件系统的接口，对对象操纵和管理的软件集合，对象及属性。从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

## 1.2 项目目的

1、理解文件存储空间的管理；

2、掌握文件的物理结构、目录结构和文件操作；

3、实现简单文件系统管理；

4、加深文件系统实现过程的理解

## 1.3 项目要求

基本要求

1、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;

2、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：

1. 文件存储空间管理可采取显式链接（如FAT）或者其他方法；
2. 空闲空间管理可采用位图或者其他方法；
3. 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。

# 2 需求分析

## 2.1 格式化

清空文件系统中的所有的文件目录和文本文件。

## 2.2 创建子目录

在当前目录下新建立一个目录，要求与当前目录下的文件目录不重名。

## 2.3 删除子目录

在当前目录下删除一个目录，当该目录下有其他的目录和文件时，一并删除。

## 2.4 显示目录

显示当前目录的路径和目录下的内容。

## 2.5 更改当前目录

可以选择退回到上级目录，更改当前目录。

## 2.6 创建文件

在当前目录下新建立一个文件，要求与当前目录下的文件不重名。

## 2.7 打开文件

选中文件打开，可以看到其存放的文本内容，同时可以对其进行修改等操作。

## 2.8 关闭文件

关闭文件后，输入的相关内容被存放到文件系统中。

## 2.9 删除文件

删除当前目录下选中的文件。

## 2.10 保存

将文件系统中的内容保存到磁盘上。

## 2.11 导入

从磁盘上导入相应的内容。

# 3 数据结构

## 3.1 树形目录结构

树形目录结构是文件目录，文件等内容构成树。可以方便地对文件进行分类，层次结构清晰，也能有效的进行文件的管理和保护。访问文件时，需要逐级访问。

## 3.2 位示图

利用数组中的一位来表示磁盘中块的使用情况，磁盘上所有的盘块都有一位来与之对应。

## 3.3 显式链接分配

把用于连接文件各个物理块的指针，从原有的隐式链接的每个物理块中的末尾提取出来，显式地放入在内存中地一张表。当文件要查找当前块地下一块地址时，只需找到链接表中相应位置即可找到相应地指针。

# 4 设计实现

## 4.1 FAT类

FAT类将位示图和显式链接结合到一起，在该类中开辟一个空间大小未2048的数组，每一位都置为-2。

class FAT(object):

def \_\_init\_\_(self):

self.bitMap = [-2 for i in range(1024 \* 2)]#位示图结合显示连接

## 4.2 FCB类

FCB类为文件控制块，保存有文件的文件名，文件的父亲节点的路径，文件第一块在位示图中的位置，文件的大小，文件的类型以及如果是目录，存放目录下的文件。

class FCB(object):#文件控制表

def \_\_init\_\_(self, name, path, address, size, type):

self.name = name#文件名

self.path = path#父节点的路径

self.address = address #第一块内容在位示图中的位置

self.size = size # 文本文件内容大小

self.type = type # 文件类型

self.child = {} #文件的子节点的路径

## 4.3 创建文本文件

首先获取路径，判断该路径是否是目录，随后弹出弹窗输入新建立的文本文件名。判断文本文件名是否符合要求，不符合则退出。取出当前选中的目录树的节点，将它的子节点全部取出，判断新建立的文件名是否有重名。如果有，则退出该函数。随后，弹出弹窗要求输入文本文件的内容。判断位示图上是否有足够的空间。随后在FCB列表中加入新添加的文本文件，文件内容填入了存放内容的列表中。当前目录添加新建的文本文件。

def createText(self):#创建文本文件

path = self.getPath()#获取路径

if path==False or self.FCBlist[path].type == 'file':#判断是否是文件

return False

name, isSuccess = QtWidgets.QInputDialog.getText(self, "name", "请输入名称", QtWidgets.QLineEdit.Normal, '')#对话框输入

if name == '':#判断输入名字是否正确

return False

currentFolder = self.treeWidget.currentItem()

childNode = currentFolder.takeChildren()

if self.isRepeat(childNode,"file",path,name,currentFolder):

return False

text, isSuccess = QtWidgets.QInputDialog.getMultiLineText(self, "file", '请输入文本内容', '')#对话框输入内容

address = self.fillBlock(len(text))#填入位示图

if address == -1:#判断是否有空间

currentFolder.addChildren(childNode)

return False

currentFolder.addChildren(childNode)

self.addNode(name+'.txt',currentFolder)#加入树节点

insertFCB = FCB(name + '.txt', path, address, len(text), 'file')#修改插入FCB

currentPath = path + '\\' + name + '.txt'

self.FCBlist[currentPath] = insertFCB

self.storageContent[currentPath] = text

self.FCBlist[path].child[currentPath] = insertFCB

self.treeWidget.expandAll()





## 4.4 创建目录

首先获取路径，判断该路径是否是目录，随后弹出弹窗输入新建立的文件目录名。判断文本目录名是否符合要求，不符合则退出。取出当前选中的目录树的节点，将它的子节点全部取出，判断新建立的文件目录名和原有的是否有重名。如果有，则退出该函数。随后在FCB列表中加入新添加的文件目录。当前目录添加新建的文件目录作为子节点。

def createFolder(self):#创建目录

path = self.getPath()#获取路径

if path==False or self.FCBlist[path].type == 'file':#判断是否是文件目录

return False

name, isSuccess = QtWidgets.QInputDialog.getText(self, "name", "请输入名称", QtWidgets.QLineEdit.Normal, '')#输入文件名称

currentFolder = self.treeWidget.currentItem()

childNode = currentFolder.takeChildren()

if self.isRepeat(childNode,"folder",path,name,currentFolder):

return False

currentFolder.addChildren(childNode)

if name == '':

return False

self.addNode(name,currentFolder)#加入树节点

path = self.getPath()

currentFCB = FCB(name, path, '', '', 'folder')#修改插入FCB

fullPath = path + '\\' + name

self.FCBlist[fullPath] = currentFCB

self.FCBlist[path].child[fullPath] = currentFCB

self.treeWidget.expandAll()





## 4.5 删除文本文件

首先获取路径，判断该路径是否是目录还是文件。对于文件，找到当前所要删除的文件物理块的起始地址。随后通过递推的方式将当前所要删除的物理块的位示图回复成空闲状态。随后FCB列表中不在存放当前节点，当前节点的父节点移除该子节点。

def deleteText(self, textNode, path):#删除文本文件

currentFCB = self.FCBlist[path]

fileHead = currentFCB.address

next = self.myFAT.bitMap[fileHead]

self.myFAT.bitMap[fileHead] = -2

while next != -1 and next >= 0:#位示图填充初始状态

prev = next

next = self.myFAT.bitMap[next]

self.myFAT.bitMap[prev] = -2

self.deleteFCB(path)#删除节点

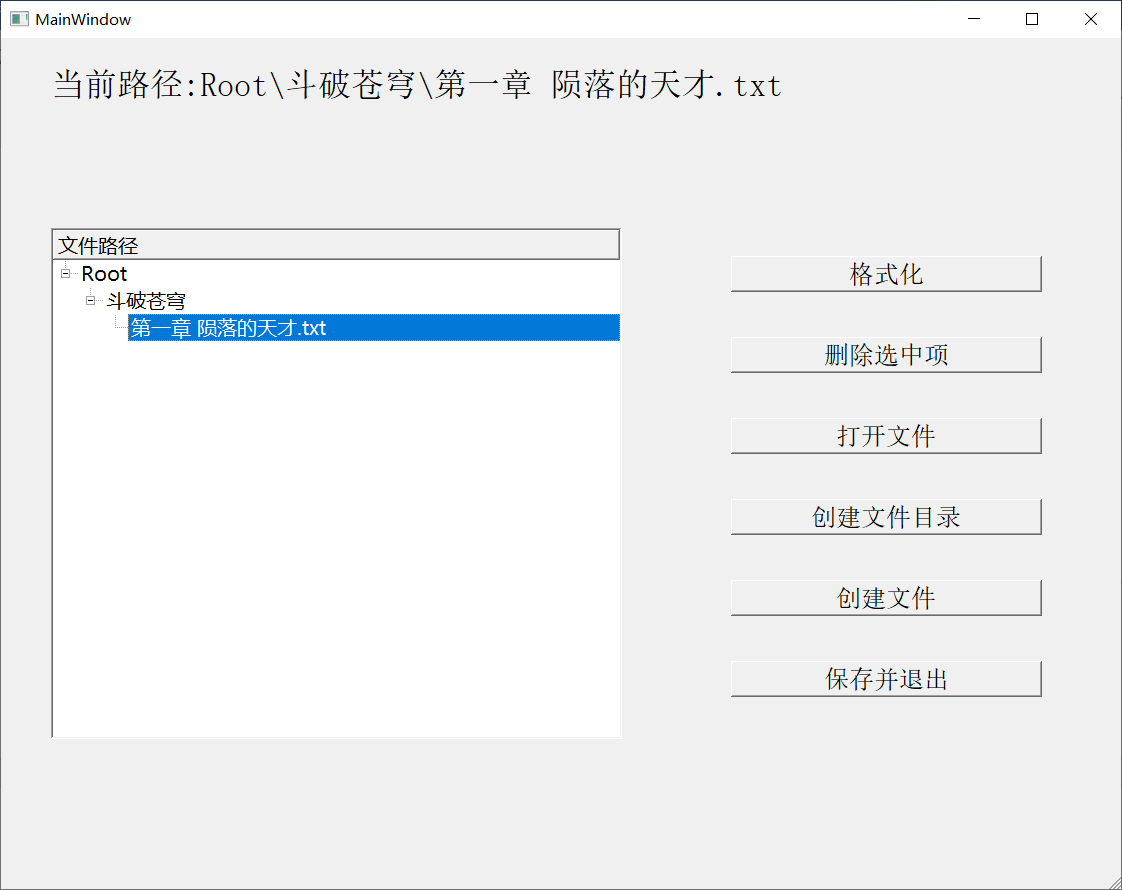
self.storageContent.pop(path)

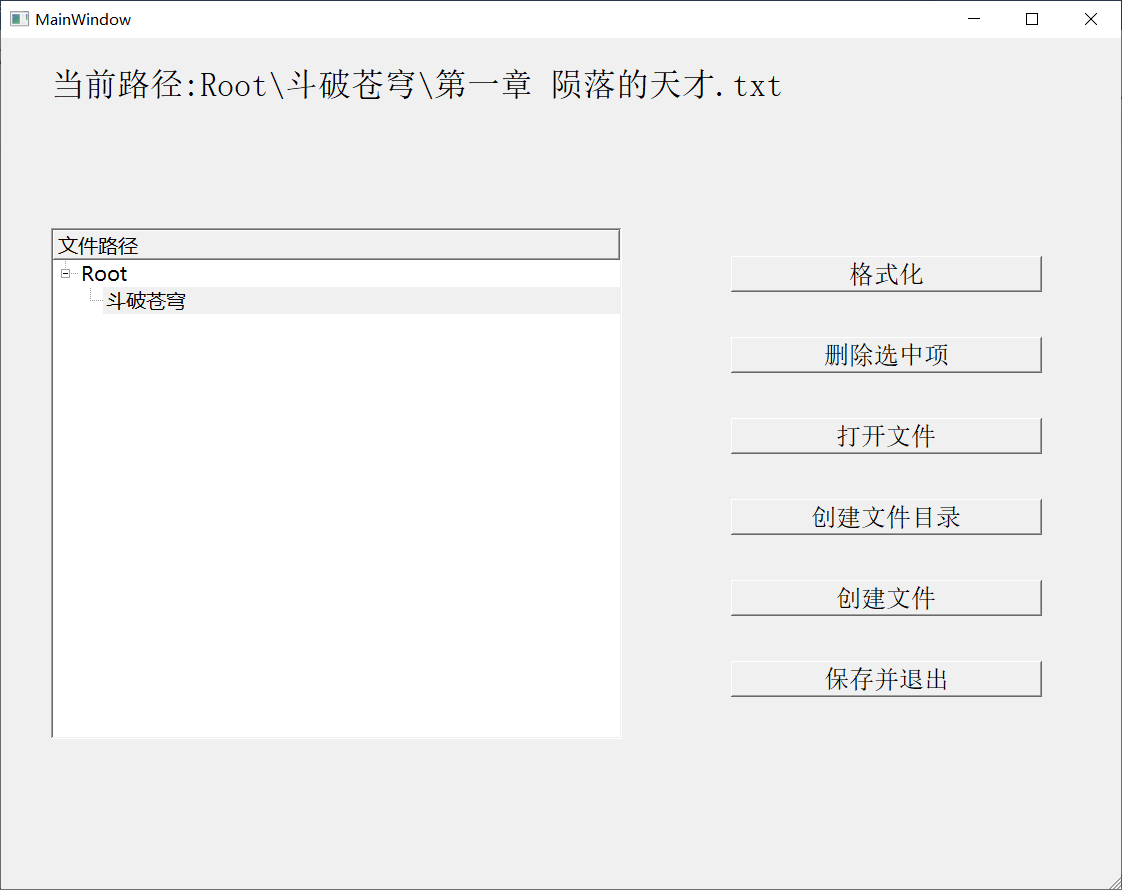
parentNode = textNode.parent()

if parentNode:

parentNode.removeChild(textNode)

self.treeWidget.expandAll()





## 4.6 删除文件目录

首先获取路径，判断该路径是否是目录还是文件。对于目录，如果它没有相应的子节点，则直接删除该节点。如果有子节点且是文本文件的话，则调用相应的删除文本文件的函数。如果是文件目录的话，则递归调用删除自身的函数。

def deleteFolder(self, folderNode, path):#删除文件目录

childNode = folderNode.takeChildren()

if len(childNode) == 0:

if path in self.FCBlist:

self.deleteFCB(path)

parentNode = folderNode.parent()

if parentNode:

parentNode.removeChild(folderNode)

return

for child in childNode:#对于每个子节点

name = child.text(0)

childPath = path + '\\' + name

if self.FCBlist[childPath].type == 'file':

self.deleteText(child, childPath)

else:

self.deleteFolder(child, childPath)

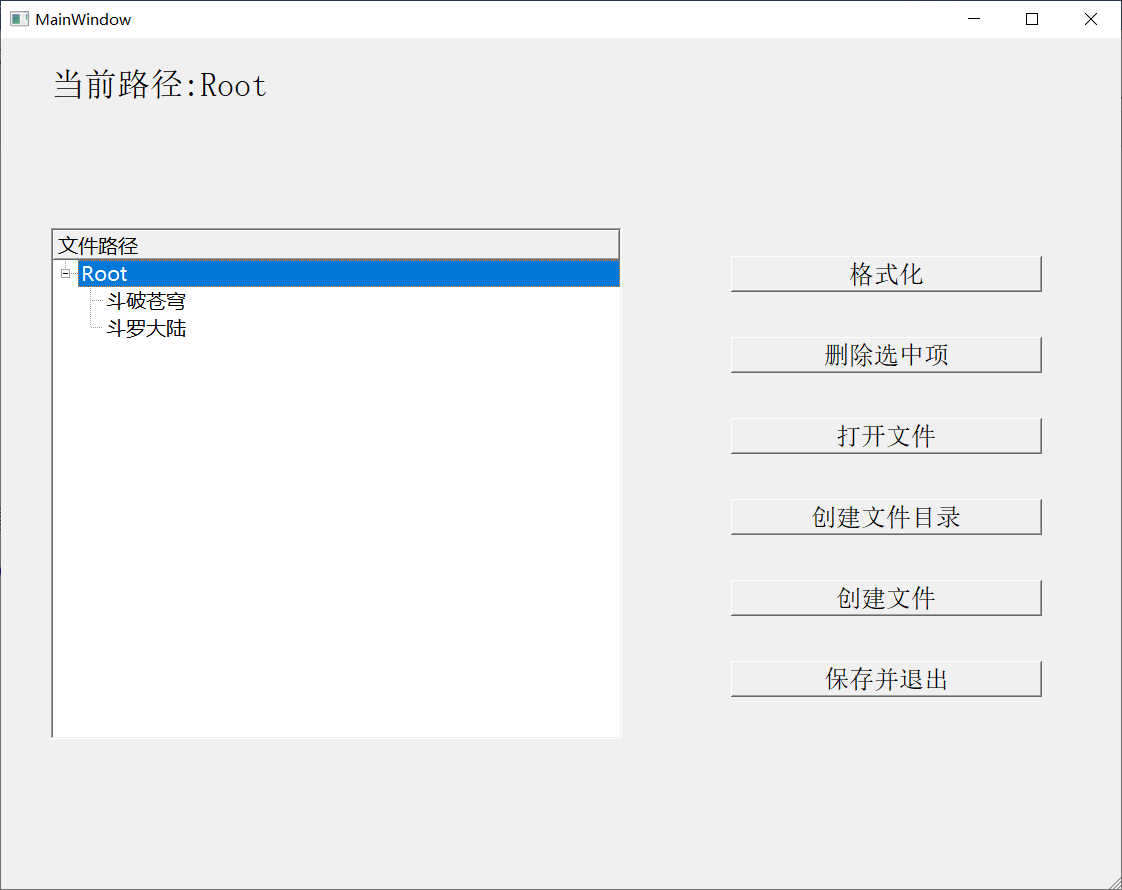
if path!='Root':#除根节点外的操作

parentNode = folderNode.parent()

if parentNode:

parentNode.removeChild(folderNode)

self.deleteFCB(path)





## 4.7 格式化

直接调用删除目录的函数，以根节点为起始点，删除目录。

def formatStorage(self):#格式化

self.deleteFolder(self.item\_0,'Root')



## 4.8 打开文本文件

首先获取路径，判断该路径是否是目录，随后弹出弹窗显示当前文本的内容。用户可以修改其中的内容，当新的文本被读到时，将其进行处理。随后对其和原有的文件物理块的数量进行比较。如果大于，则需要在原有的物理块之后添加新的块。通过遍历位示图，找到当前文件的最后一块，将新产生的块的头部地址链接到原有的尾部地址。

如果小于，则需要删除原有物理块中的多出来的部分，将其还复为空闲状态。通过遍历的方式找到链接尾部，但是过程中保存寻找到的链接，随后按空余的大小对块进行删除。

def openFile(self): # 读写文件内容

path = self.getPath()

if path==False or self.FCBlist[path].type == 'folder':

return False

currentText = self.storageContent[path]

inputText, isSuccess = QtWidgets.QInputDialog.getMultiLineText(self, "File", '请输入文本内容', currentText)#获取文本

if isSuccess:

self.storageContent[path] = inputText

newNum = math.ceil(len(inputText)/512)

oldNum = math.ceil(self.FCBlist[path].size/512)

currentFCB = self.FCBlist[path]

if newNum > oldNum:#如果新块大于旧块

shortage = newNum - oldNum

fileHead = currentFCB.address

next = self.myFAT.bitMap[fileHead]

pre = fileHead

while next != -1 and next >= 0:#找到链接尾部

pre = next

next = self.myFAT.bitMap[next]

address = self.fillBlock(len(inputText)+(self.FCBlist[path].size % 512)-512-self.FCBlist[path].size)#添加块

self.myFAT.bitMap[pre] = address

self.FCBlist[path].size = len(inputText)

elif newNum < oldNum:#如果新块小于旧块

abound = oldNum - newNum

fileHead = currentFCB.address

next = self.myFAT.bitMap[fileHead]

pre = fileHead

par = {next: pre}

while next != -1 and next >= 0:#找到尾部

pre = next

next = self.myFAT.bitMap[next]

par[next] = pre

eliminateBit = par[next]

for i in range(abound):#回溯复原块

self.myFAT.bitMap[eliminateBit] = -2

if eliminateBit in par:

eliminateBit = par[eliminateBit]

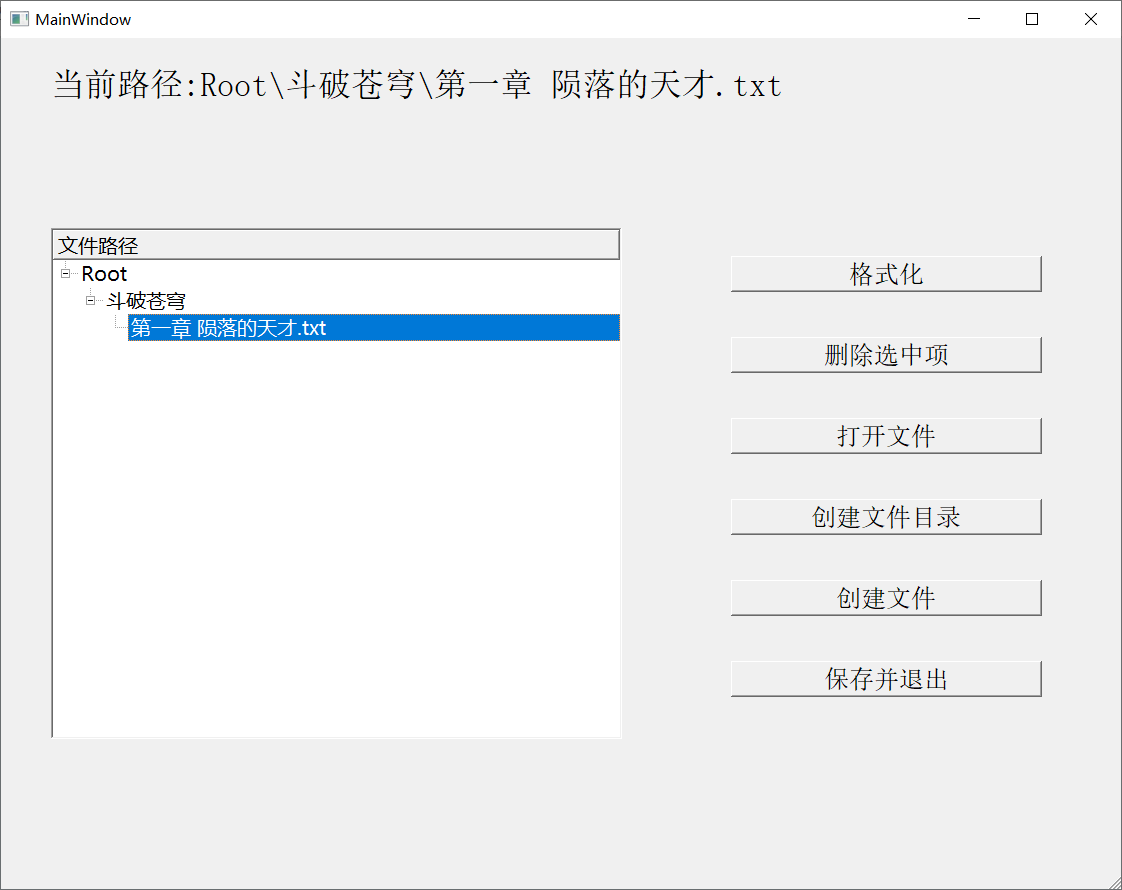
if newNum == 0:

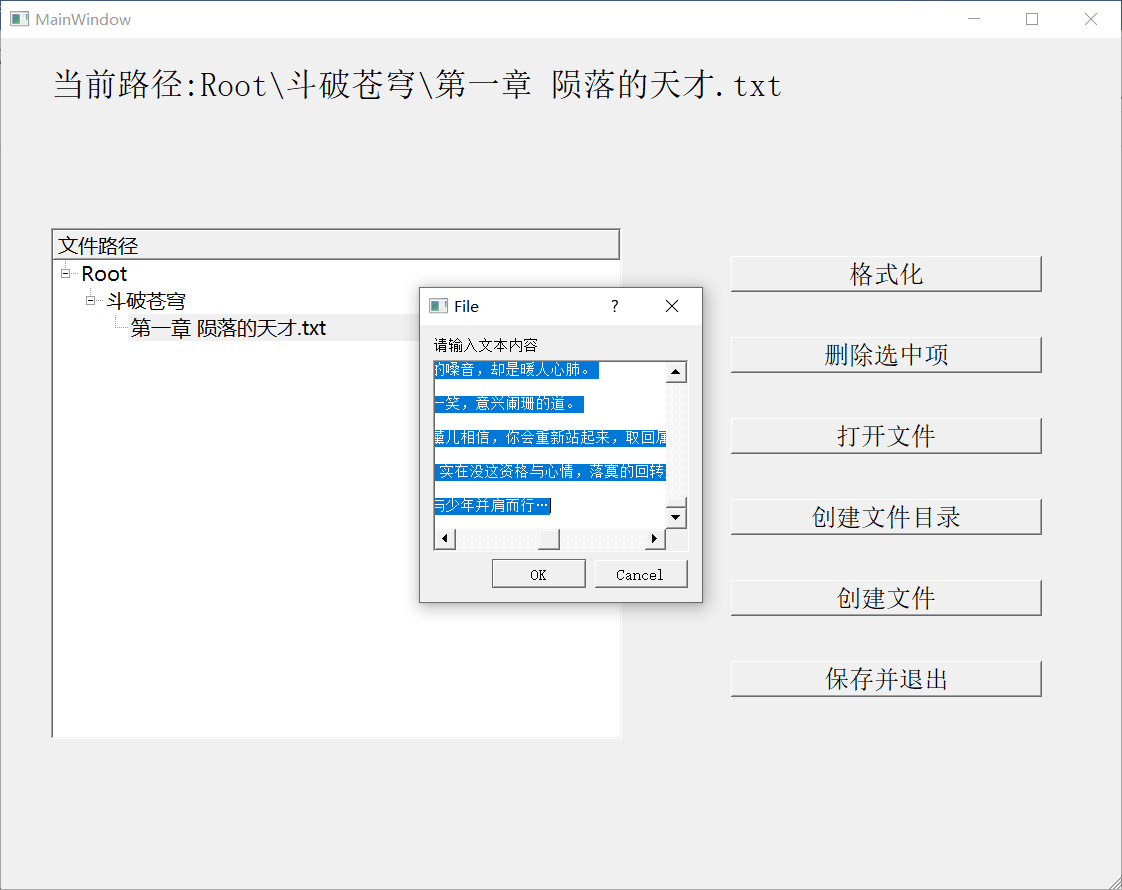
self.myFAT.bitMap[eliminateBit] = -2

else:

self.myFAT.bitMap[eliminateBit] = -1

self.FCBlist[path].size = len(inputText)





## 4.9 保存与导入

将FCB和FAT以及文本内容保存到二进制文件中。导入时，从二进制文件中导入被保存的内容。递归初始化建立原有的目录树结构。

def storeFile(self):#存入二进制文件

fileFCB=open('fileFCB.bin','wb')

pickle.dump(self.FCBlist,fileFCB)

fileFCB.close()

fileFAT=open('fileFAT.bin','wb')

pickle.dump(self.myFAT.bitMap,fileFAT)

fileFAT.close()

fileCNT=open('fileCNT.bin','wb')

pickle.dump(self.storageContent,fileCNT)

fileCNT.close()

def importStorage(self):#从二进制文件中读出

fileFCB=open('fileFCB.bin','rb')

self.FCBlist=pickle.load(fileFCB)

fileFCB.close()

fileFAT=open('fileFAT.bin','rb')

self.myFAT.bitMap=pickle.load(fileFAT)

fileFAT.close()

fileCNT=open('fileCNT.bin','rb')

self.storageContent=pickle.load(fileCNT)

fileCNT.close()

self.createTree('Root',self.item\_0)

def createTree(self,path,parentNode):#递归的构造目录树

for i in self.FCBlist[path].child:

item=self.addNode(self.FCBlist[i].name,parentNode)

self.createTree(i,item)



# 5附录

## 5.1 文件说明

documentSystem.py为文件系统的核心代码

mainUI.py为文件系统的界面代码

documentSystem.exe为生成的可执行文件

fileFCB.bin为存放FCB的二进制文件

fileFAT.bin为存放FAT的二进制文件

fileCNT.bin为存放文本内容的二进制文件

由于python环境问题，exe文件运行并不稳定（使用pyinstaller生成），所以使用时建议执行documentSystem.py，所需环境为：PyQt5

## 5.2 开发环境

操作系统：windows10 1909 64bit

语言：python 3.7

相关python包：PyQt5, PyQt-tools, PyUIC，pickle