# FileManagement文件系统管理设计方案

## 1 项目简介

### 1.1 基本任务

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。

### 1.2 具体要求

* 文件存储空间管理
* 本项目采取了隐式链接方式。目录项中含有文件第一块的指针和最后一块的指针。每个文件对应一个磁盘块的链表；磁盘块分布在磁盘的任何地方，除最后一个盘块外，每个盘块都含有指向文件下一个盘块的指针，这些指针对用户是透明的。
* 空闲空间管理
* 本项目使用位图方法，用一串二进制位反映磁盘空间中的分配使用情况, 每个物理块对应一位, 分配物理块为1，否则为0。
* 文件目录
* 本项目采用了树形目录结构，文件目录节点中应包含：文件名、物理地址、长度（以FCB形式存储）等信息;文件夹目录节点中应包含：所有子节点的信息。

### 1.3实现功能

#### 基本操作：

* 实现了格式化、创建子目录、删除子目录、显示目录、更改当前目录；
* 实现了创建文件、 打开文件、关闭文件、 写文件、 读文件、 删除文件；

#### 功能操作：

* **界面展示详实**
* 界面展示了多级目录结构，上部则展示了当前所在路径；
* **文件的管理**
* 本项目通过隐式链接的方法对文件存储空间进行管理，而文件目录采用多级目录结构，每一项包括了文件名、是文件或文件夹、物理地址、长度信息。
* **使用简易**
* 通过在文件夹下新建文件夹或新建文件、对文件进行编辑、删除文件或文件夹、格式化5个集成功能、实现了所有基本操作要求。

## 2 项目环境

### 2.1 开发环境

* Python 3.8，基于PyQt5
* windows 11

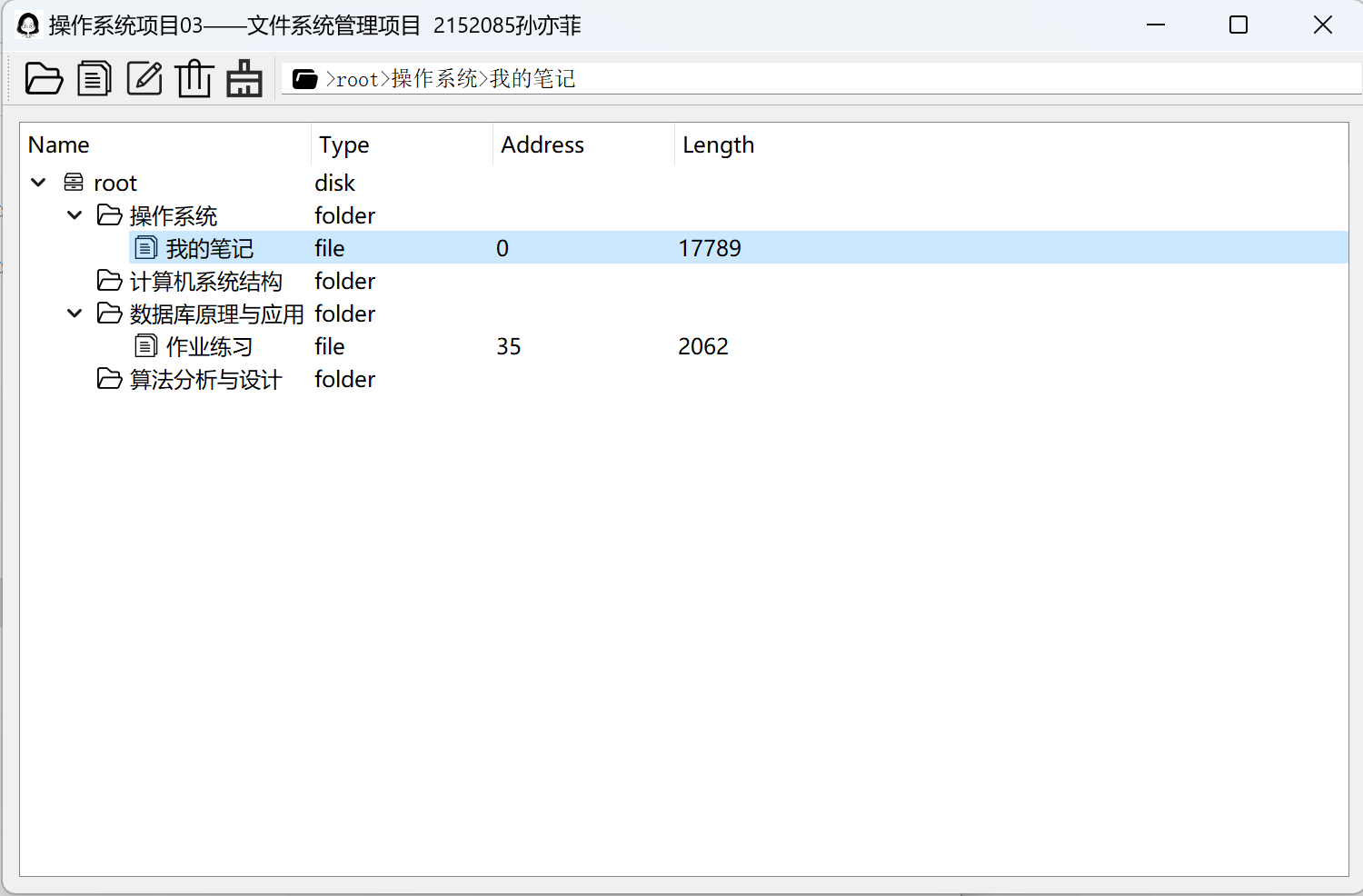
### 2.2 项目结构

├─exe  
│ │ ...  
│ │ FileManagement.exe  
│ │  
│ ├─ico  
│ │  
│ └─PyQt5  
└─src  
 │ FileManagement.py  
 │  
 └─ico

### 2.3 运行

* 直接运行:点击exe/FileManagement.exe
* 运行源码：进入src文件夹，命令行执行:python FileManagement.py

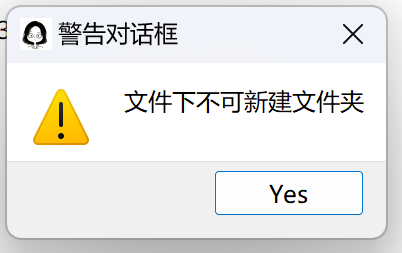
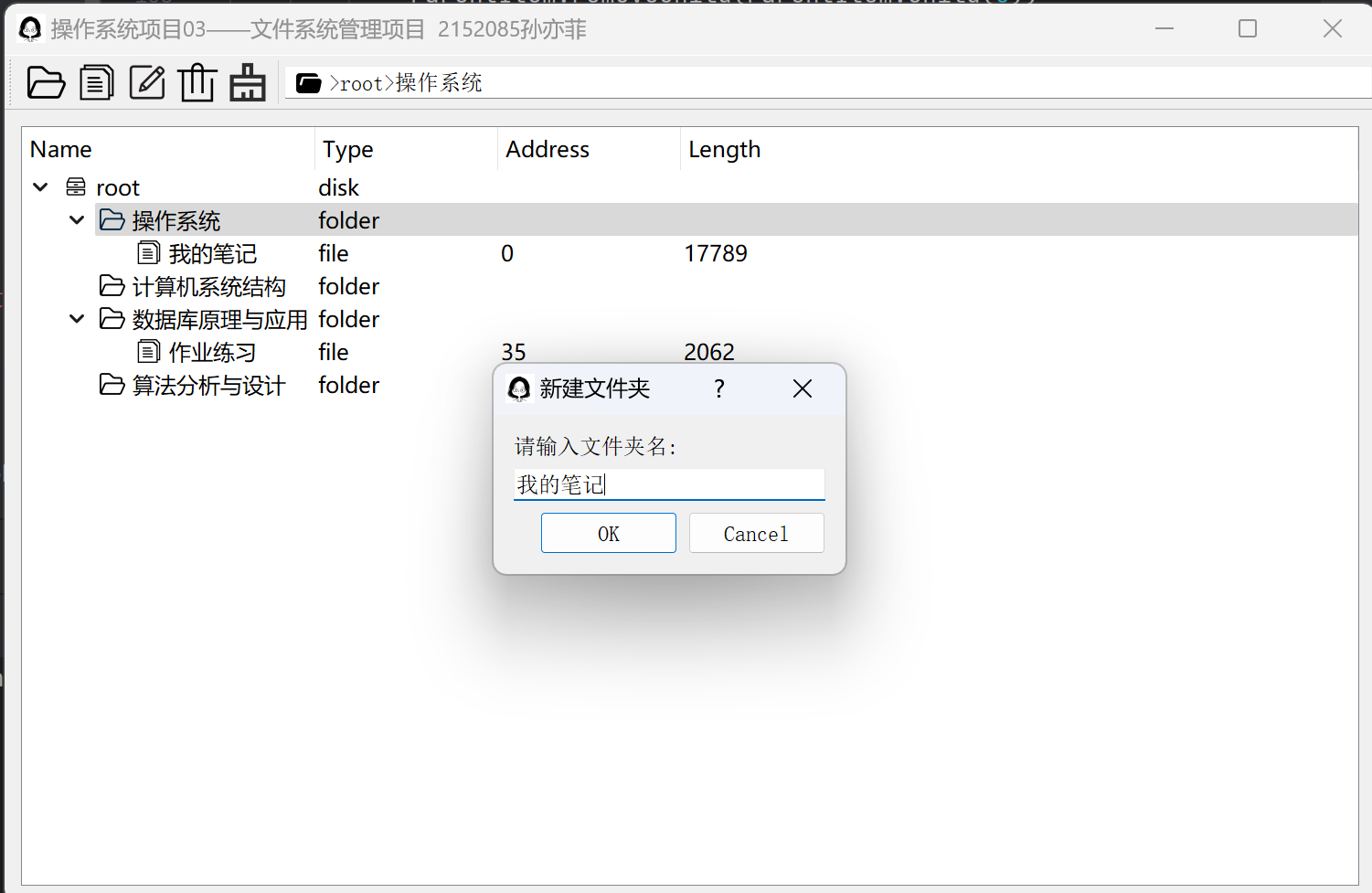
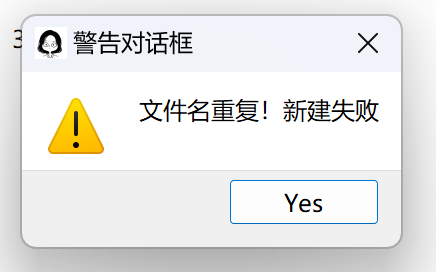
## 3 界面及功能介绍



### 3.1 当前目录文件夹下新建文件夹

* 该图标指在当前文件夹下新建一个文件夹

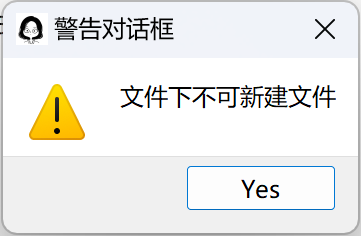
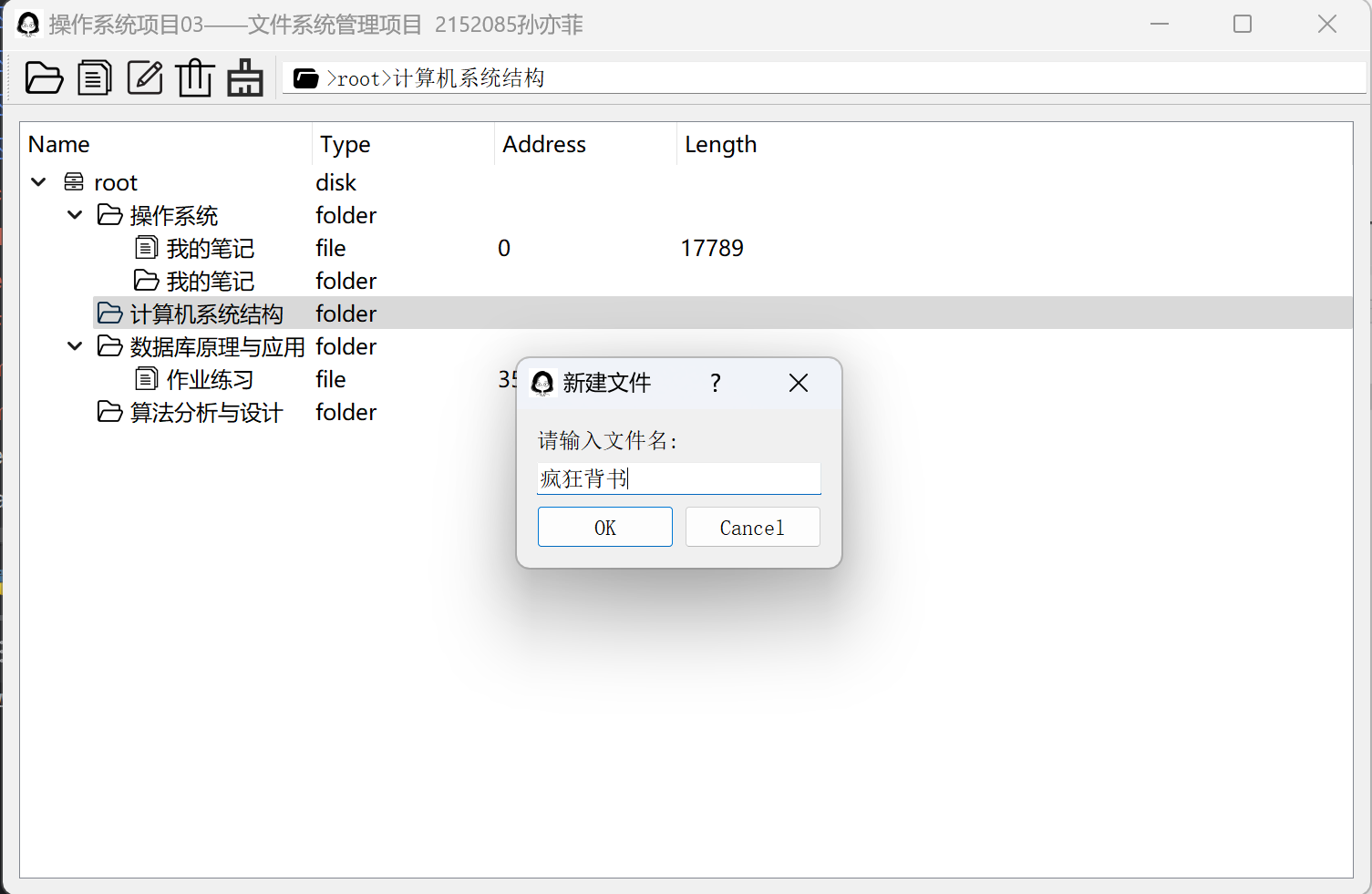
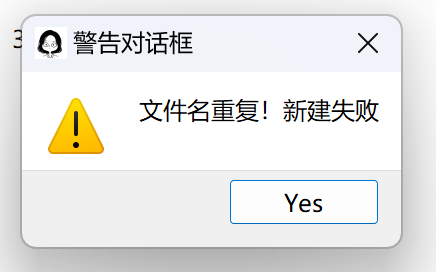


* 当前路径所指为文件，则不可新建文件夹
* 
* 弹窗输入新建文件夹名称
* 
* 检测是否有重复文件名
* 

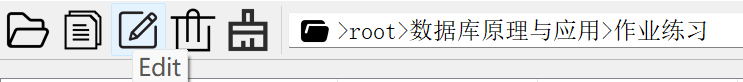
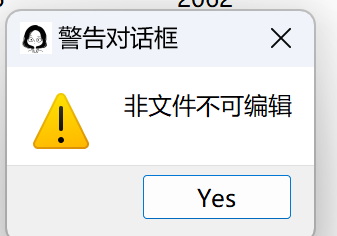
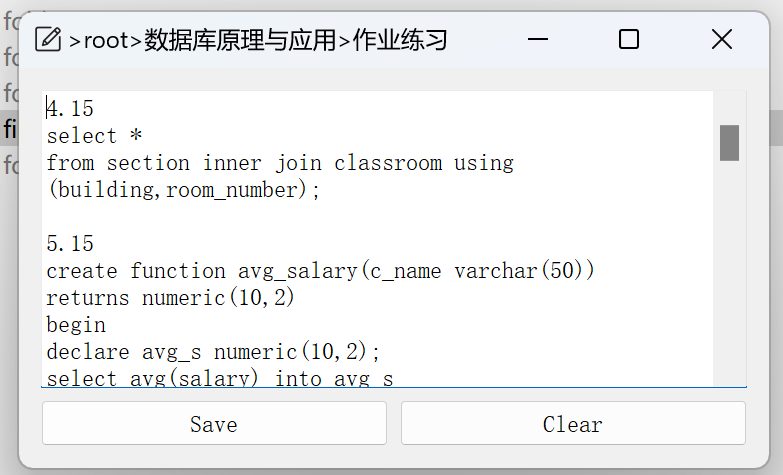
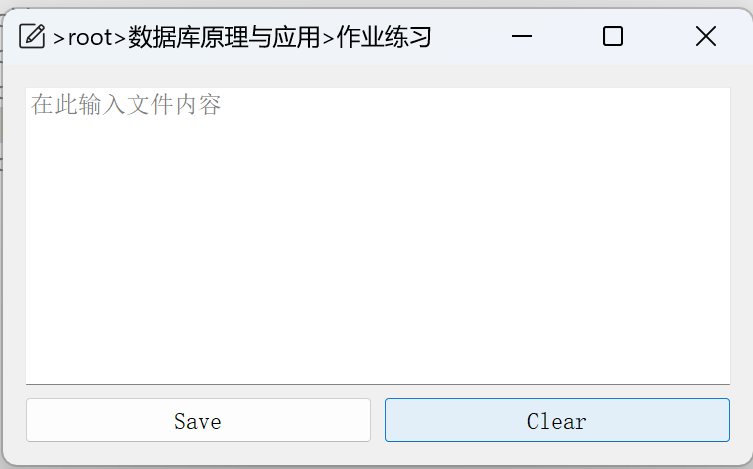
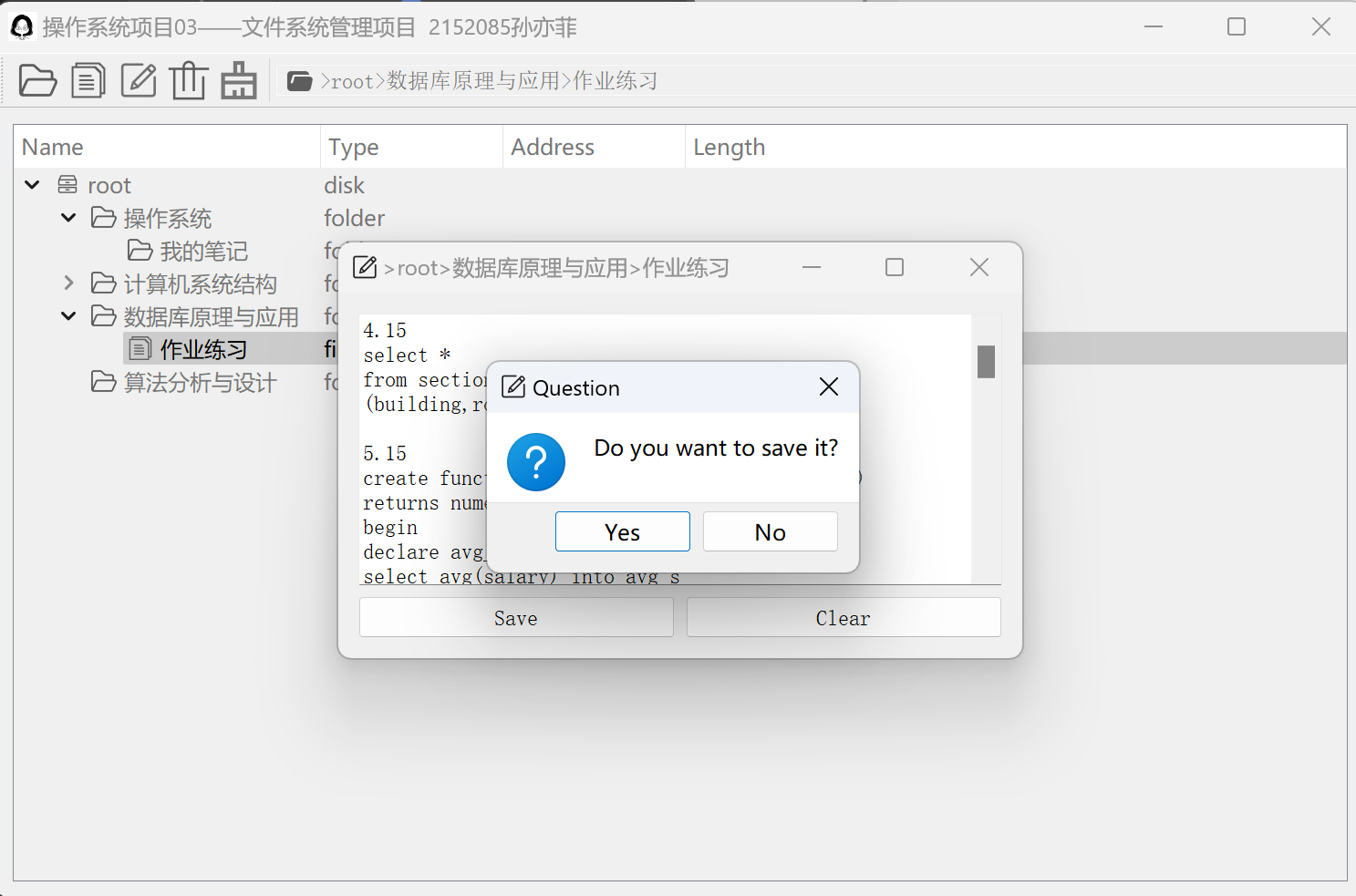
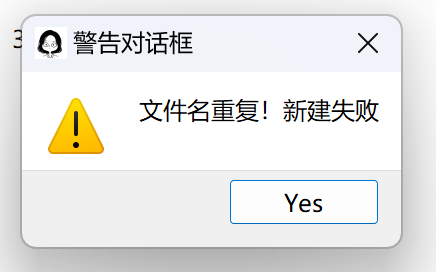
### 3.2 当前目录文件下新建文件

* 该图标指在当前文件夹下新建一个文件



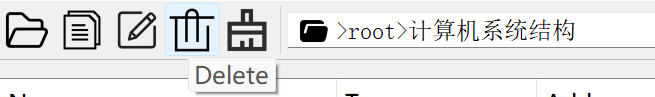
* 当前路径所指为文件，则不可新建文件
* 
* 弹窗输入新建文件名称
* 
* 检测是否有重复文件名
* 

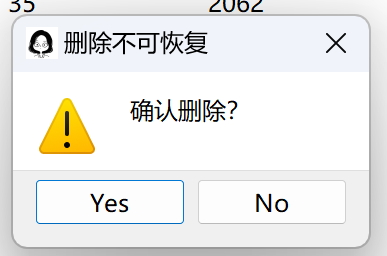
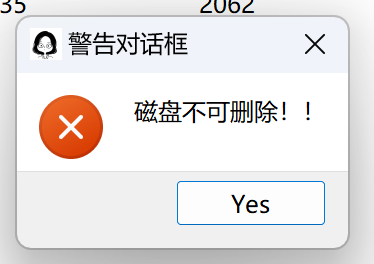
### 3.3 编辑文件

* 该图标指编辑当前文件
* 当前路径所指为文件夹，则不可编辑
* 
* 弹窗阅读文件已有内容，并可进行编辑
  + 输入内容
* 
  + 清空当前内容
  + 
  + 选择是否保存当前编辑
  + 
* 检测是否有重复文件名
* 

### 3.4 删除文件

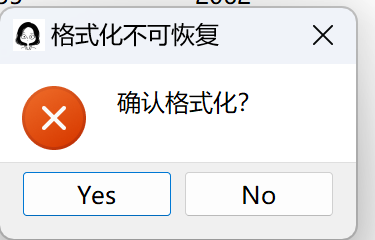
* 该图标指编辑当前文件/文件夹



* 弹窗确定删除操作
* 
* 根目录磁盘不可删除
* 

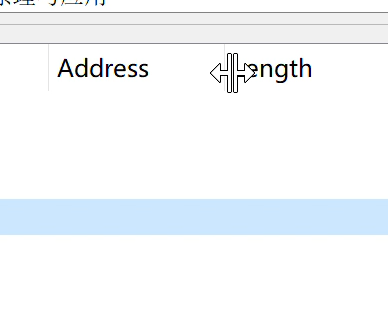
### 3.5 格式化

* 该图标指格式化磁盘
* 弹窗确认格式化操作



### 3.6 其他操作

* 在目录间可以用鼠标拖动调整宽度（尤其用于显示较长文件名）



## 4 实现算法

### 4.1 写入文件：隐式链接及空闲块表更新

本项目实现了一个FileSystemManagement类，其中具有FreeList为空闲块表，Catalog则是隐式链接中的目录设置，每个目录项记录文件名、起始地址、终止地址。

class FileSystemManagement:  
 ##空闲表  
 def \_\_init\_\_(self,blockNum):  
 self.FreeList=[]  
 self.blockNum=blockNum  
 for i in range(blockNum):  
 self.FreeList.append(0)  
 self.Catalog=dict()

在写数据的方式中，也同时完成了对隐式链接及空闲块表的更新。每找到一个新的块可以写数据，则需要让新的块的空闲块表位置1，同时上一个块具有的指针指向新的块。最后一个块指向-1表示终止。

def findBlank(self):  
 for i in range(self.blockNum):  
 if self.FreeList[i]==0:  
 return i  
 return -1  
  
 #写入文件的数据、名称  
 def write(self,data,disk,name):  
 last=-1  
  
 if(self.Catalog.get(name)!=None):  
 last=self.Catalog.get(name)[1]  
  
 while data!="":  
 cur=self.findBlank()  
 if(cur==-1):  
 raise Exception(print('磁盘空间不足!'))  
  
 else:  
 if(last == -1):  
 self.Catalog[name]=[cur,-1]  
 else:  
 disk[last].setNext(cur)#指针指下一个  
 data=disk[cur].write(data)#写入  
 self.FreeList[cur]=1#空闲表  
  
 last=cur#上一个  
  
 self.Catalog[name][1]=last  
 disk[last].setNext(-1)

### 4.2 读入文件：隐式链接的应用

利用了隐式链接，从起始地址开始，顺着指针向下读取数据，获取了最终文件的全部内容。

def read(self,name,disk):  
 data=""  
 if self.Catalog.get(name)==None:  
 self.Catalog[name]=[-1,-1]  
  
 cur=self.Catalog[name][0]  
  
 while(cur!=-1):  
 data+=disk[cur].read()  
 cur=disk[cur].getNext()  
 return data

### 4.3 递归删除目录项

在删除目录项时，需要递归其所有子目录项，递归的方式较容易解决该问题。

def deleteCatalogNode(self,ParentNode:CatalogNode):  
 self.catalog.remove(ParentNode)  
  
 if ParentNode.type!="file":  
 count=len(ParentNode.children)  
  
 for i in range(count):  
 self.deleteCatalogNode(ParentNode.children[0])  
 ParentNode.children.remove(ParentNode.children[0])  
 else:  
  
 ParentNode.data.delete(self.manager,self.disk)

## 5 项目总结

### 5.1 项目亮点

* 实现了文件管理基本功能。
* 界面美观，使用方便，不同类型文件和文件夹图标各异。
* 提示清晰，能够模拟成熟操作系统中的容错提醒功能及当前路径显示。

### 5.2 项目改进方向

与真正成熟的操作系统中文件管理系统相比，还有很多可以实现的功能。如多种目录跳转方式，支持文件名查找文件，能够复制文件粘贴文件，能够显示文件的更多信息如修改时间，以及显示文件夹中视图等待。

同时，还有很多其他策略能够对文件系统进行管理，如FAT表。这些都是未来考虑改进的方向。