文件管理系统

1953348 叶栩冰 操作系统第三次作业

您好：

这是我的第三次操作系统课程作业文件管理系统，其中可执行文件文件夹中Run.exe为可执行文件，双击即可运行，您不可删除同根目录下的EM/IMAGE等配置文件，执行Run.exe弹出模拟文件管理窗口，请您阅读窗口右上方的操作方法，您可以在右侧选中想要修改的文件，右击鼠标来进行操作，详见可执行文件与项目报告。

项目需求

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，以便下次可以将其恢复到内存中来。

功能描述

- 文件存储空间管理可采取显式链接（如FAT）或者其他方法。（即自选一种方法）
- 空闲空间管理可采用位图或者其他方法。如果采用了位图，可将位图和FAT表合二为一。
- 文件目录采用多级目录结构。至于是否采用索引节点结构，自选。目录项目中应包含：文件名、物理地址、长度等信息。同学可在这里增加一些其他信息。
- 文件系统提供的操作：
    - 格式化
    - 创建子目录
    - 删除子目录
    - 显示目录
    - 更改当前目录
    - 创建文件
    - 打开文件
    - 关闭文件
    - 写文件
    - 读文件
    - 删除文件

项目目的

- 熟悉文件存储空间的管理；
- 熟悉文件的物理结构、目录结构和文件操作；
- 熟悉文件系统管理实现；
- 加深对文件系统内部功能和实现过程的理解

开发环境

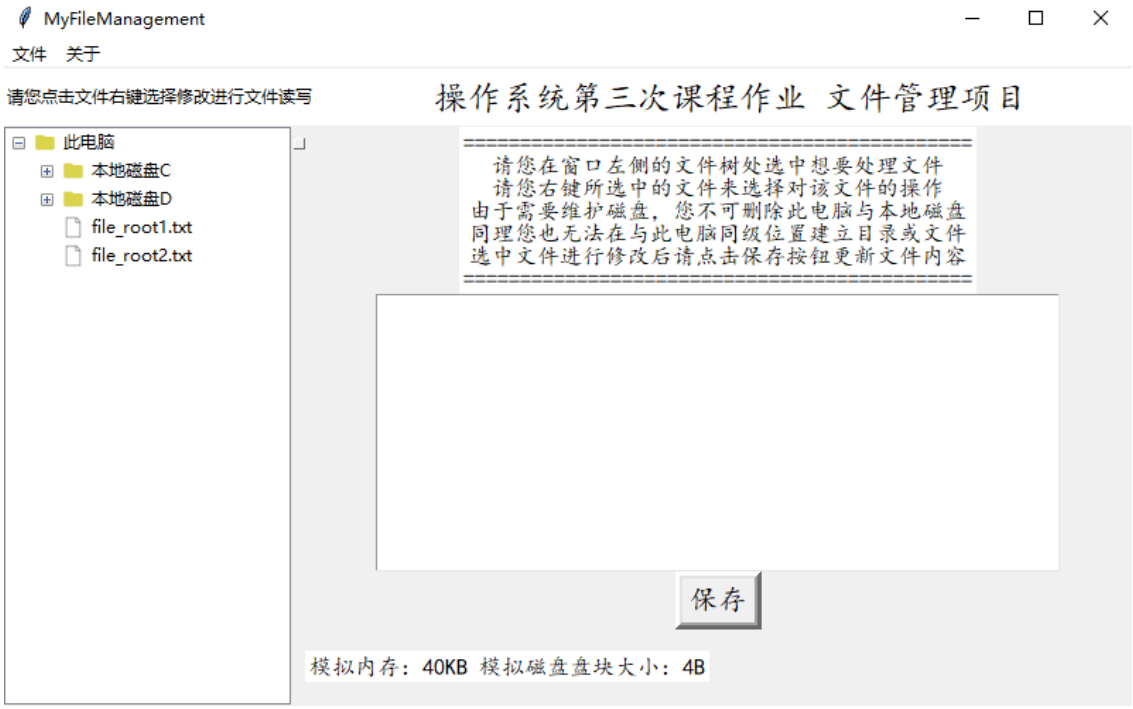
- **开发环境:** Windows 10
- **开发软件:**

PyCharm + Py38 **

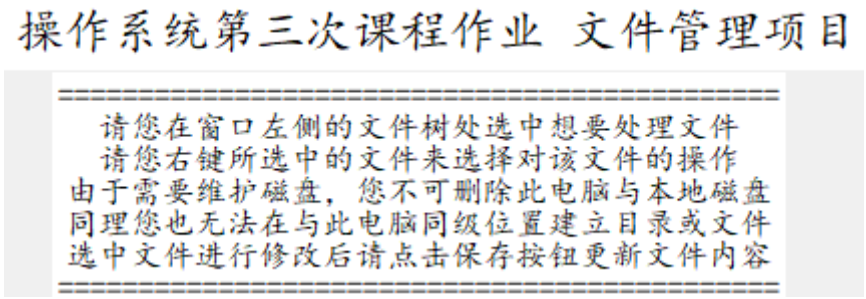
- **开发语言:** Python
- **开发框架:**Tkinter

操作说明

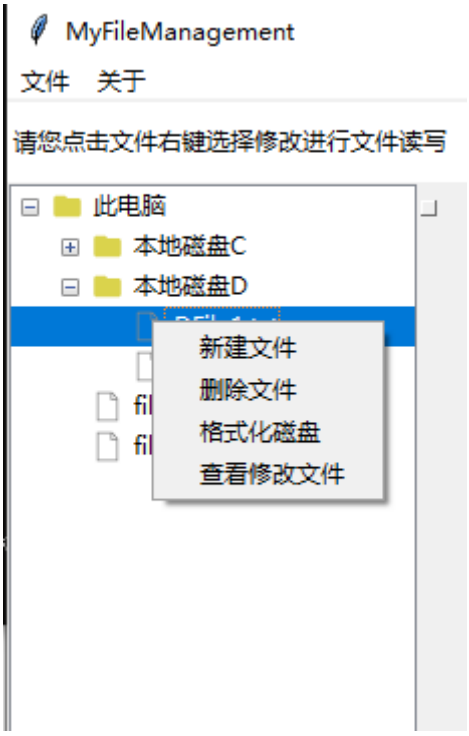
- 双击目录下`Run.exe`可执行文件进入文件系统界面



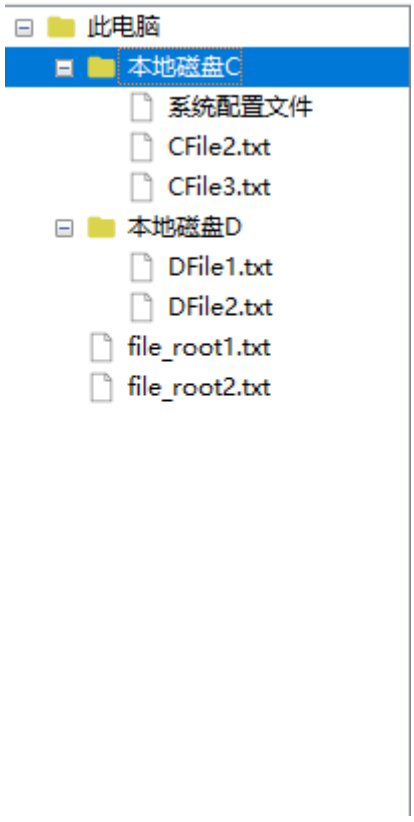
- 请详细阅读右上方的操作说明



- 选中想要进行操作的文件，单击**鼠标右键**, 进行相关操作

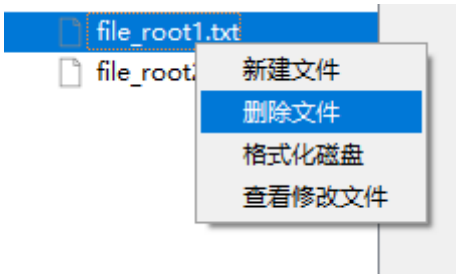


- 您可以在右侧预览所有文件

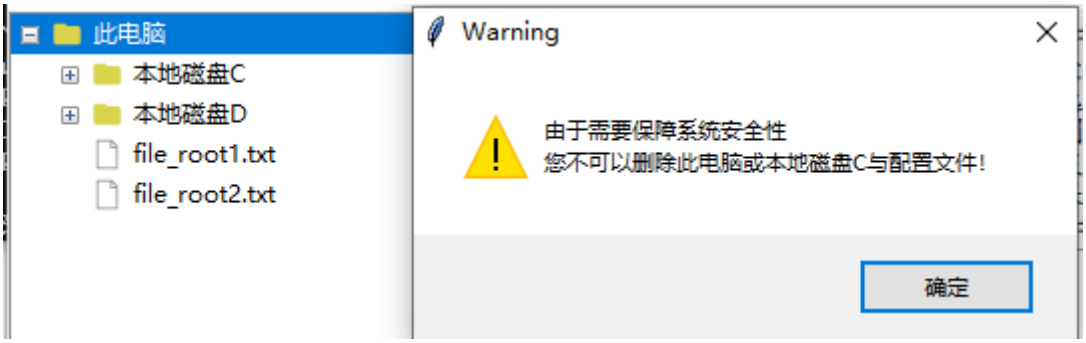


- 点击新建文件，在选中文件同级文件夹下新建文件

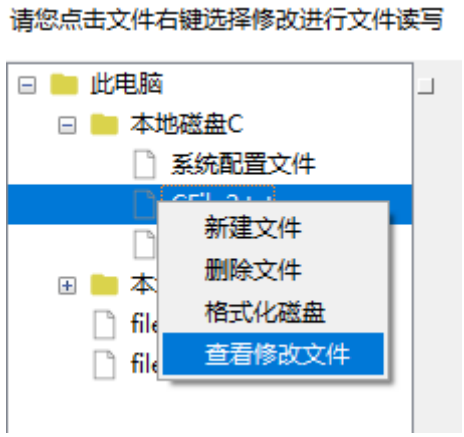
- 在文件夹/文件上点击鼠标右键可选择删除

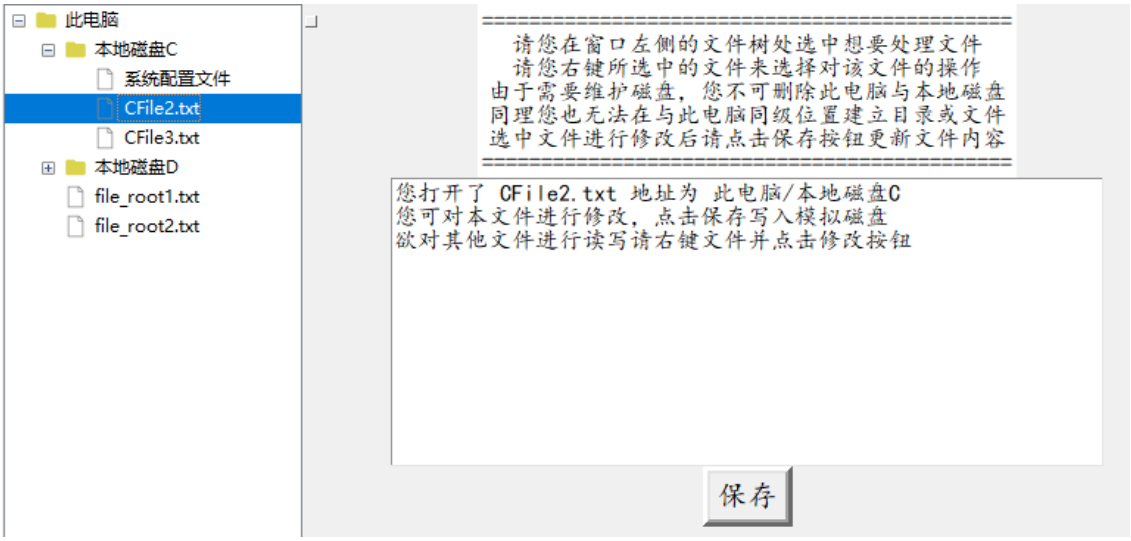


- 您不可删除此电脑、C盘及配置文件

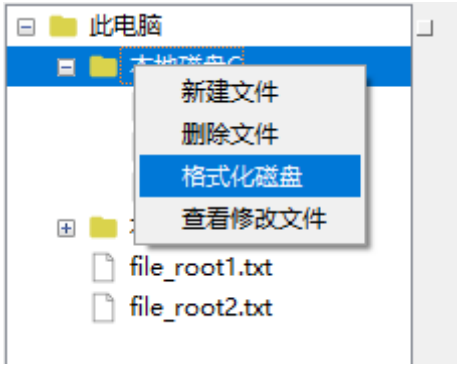


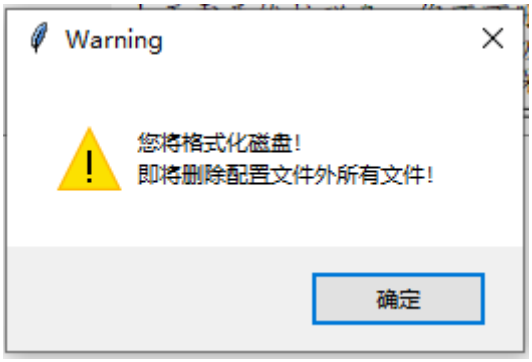
- 选中文件右键选择查看修改文件，即可在右侧查看并对文件进行修改

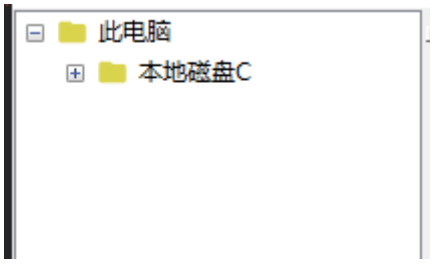




- 您可在输入框内修改，点击保存确认修改
- 右键任意文件点击格式化磁盘，即可删除除必要文件外的所有文件







方法与架构分析

哈夫曼编码模拟存储树

本文件系统中，由于进行模拟文件的分级管理，因此对文件进行哈夫曼编码，这样有利于文件的存储，且没冗余，作为FCB的一项存储到每个节点类中。

位图、FAT表

磁盘空闲空间管理在**位图**的基础上进行改造，将存放磁盘上文件位置信息的**FAT表**与传统的位图进行结合，磁盘空闲的位置使用0标识，放有文件的盘块存放文件所在的下一个盘块的位置，文件存放结束的盘块位置使用-1标识。

类设计

```
- class MyCollectApp(Toplevel)
--- 点击类
- class Application_UI(object)
--- 辅助窗体UI类
- class Application(Application_UI)
--- 主UI类
- class Content
--- 目录列表类
- class NodeFCB
--- FCB模拟文件类
```

类与架构设计

结点类设计

```
class NodeFCB:  # 目录节点
    def __init__(self, filename, iscont, tag):
        self.name = filename
        self.iscont = iscont
        self.tag = tag
        self.MDloc = 0
        self.status = 0
        self.linknum = 0
        self.endnum = 0
        # 位图 5*32*32
        # 模拟外存 5*1024
```

配置文件与全局变量设计

```
from tkinter import *
from tkinter import ttk, messagebox, filedialog
import os
import re
from PIL import Image, ImageTk
import codecs
FileName = ""
varfile = ""
inichange = NONE
```

目录表设计

```
class Content:  # 读存储目录 修改目录
    def __init__(self):
        self.contlist = []
        self.readfile()

    def readfile(self):
        f = codecs.open('memory.txt', mode='r', encoding='utf-8')  # 打开txt
文件，以‘utf-8’编码读取
        line = f.readline()  # 以行的形式进行读取文件
        while line:
            a = line.split()
            ltag = a[0:1]
            lname = a[1:2]
            lisRoot = a[2:3]
            node = NodeFCB(lname, lisRoot, ltag)
            self.contlist.append(node)
            line = f.readline()

        f.close()
```

- 本模块对memory文件进行读，将所存储的文件FCB信息读到一个NodeFCB构成的列表内，便于进行相关操作

窗体类

```
class Application(Application_UI):
    def __init__(self):
        Application_UI.__init__(self)
class Application_UI(object):
    # path = r"E:\\python开发工具\\project\\tkinter"
    path = os.path.abspath(".")
    file_types = [".png", ".jpg", ".jpeg", ".ico", ".gif"]
    scroll_visiblity = True

    font = 11
    font_type = "Courier New"
```

UI为Application的子类，这两个类实现了图形界面的绘制，具有删除，新建等基本功能的逻辑部分

```
self.load_self_tree("", self.content.contlist,
self.content.contlist[0].name[0], "0", 1)

def load_self_tree(self, root, filelist, filename, filetag, depth):
    is_open = False
    if root == "":
        is_open = True
    tag = 0
    for file in filelist:
        exists = (file.tag[0][0:depth] == filetag and len(file.tag[0]) == 1
+ len(filetag))
        if exists:
            tag = 1
            break
```

```
    print(tag)
    print(filename)
    print(filetag)
    if tag:
        root = self.tree.insert(root, END, text=" " + filename,
open=is_open, values=filetag
                                , image=self.folder_img)
    else:
        root = self.tree.insert(root, END, text=" " + filename,
open=is_open, values=filetag
                                , image=self.file_img)

    for file in filelist:
        exists = (file.tag[0][0:depth] == filetag and len(file.tag[0]) == 1
+ len(filetag))

        if exists:
            self.load_self_tree(root, filelist, file.name[0], file.tag[0],
depth + 1)
        else:
            continue
```

本函数是将列表根据哈夫曼编码转化成相应的树，并进行显示操作

```
def deleteme(self):
    print("deletefile")
    for item in self.tree.selection():
        item_text = self.tree.item(item, "values")
        print(item_text)
        if item_text[0] == '0' or item_text[0] == '00' or item_text[0] ==
'000':
            messagebox.showwarning('Warning', '由于需要保障系统安全性\n您不可以删
除此电脑或本地磁盘C与配置文件！')
            return 0
        self.tree.delete(item)
        for i in self.content.contlist:
            print(i.tag[0], i.name[0])
        print("*****")
        j = 100
        while j > 0:
            for index, file in enumerate(self.content.contlist):
                if file.tag[0] == item_text[0] or file.tag[0]
[0:len(item_text[0])] == item_text[0]:
                    self.content.contlist.pop(index)
                    break
            j -= 1
        for i in self.content.contlist:
            print(str(i.name[0]))
    self.writefile()


def newfile(self):
    print("newfile")
    for item in self.tree.selection():
        item_text = self.tree.item(item, "values")
        print(item_text)
        if item_text[0] == '0':
```

```
            messagebox.showwarning('Warning', '注意新建文件位置！\n请在此电脑根目
录下建立文件！')
            return 0
        min = '0'
        num = 0

        app = MyCollectApp()
        app.mainloop()
        global varfile, FileName

        for subitem in self.tree.get_children(self.tree.parent(item)):
            print(self.tree.item(subitem, "values"))
            if min < subitem[-1]:
                min = subitem[-1]
                num += 1
        filetag = self.tree.item(self.tree.parent(item), "values")[0]
        filetag = filetag[0:num] + str(num)
        filename = FileName + varfile

        for node in self.content.contlist:
            if len(node.tag[0]) == len(filetag) and (
                    node.name[0] == FileName or node.name[0] == FileName +
'.txt') and node.tag[0][0:len(
                node.tag[0]) - 1] == filetag[0:len(node.tag[0]) - 1]:
                messagebox.showwarning('Warning', '新建文件不可与该根目录下文件重
名！')
                return 0

        lisroot = "1"
        if varfile == "":
            self.tree.insert(self.tree.parent(item), END, text=" " +
filename, open=1, values=filetag
                             , image=self.folder_img)
            lisroot = "1"
        else:
            self.tree.insert(self.tree.parent(item), END, text=" " +
filename, open=1, values=filetag
                             , image=self.file_img)
            lisroot = "0"
            file = open("./EM/EM.txt", "a+", encoding='utf-8')
            content = 'FILENAMEIS ' + str(filetag)
            file.write(content)
            file.close()
        addnode = NodeFCB([str(filename)], [str(lisroot)], [str(filetag)])
        addnode.iscont = lisroot
        print(addnode.name)
        self.content.contlist.append(addnode)
        for i in self.content.contlist:
            print(i.tag[0], i.name[0])
        self.writefile()


def format(self):
    messagebox.showwarning('Warning', '您将格式化磁盘！\n即将删除配置文件外所有文
件！')
    self.content.contlist = self.content.contlist[0:3]
    x = self.tree.get_children()
    for item in x:
```

```
        self.tree.delete(item)
    file = open("./memory.txt", "w", encoding='utf-8')
    content = '0 此电脑 1\n00 本地磁盘C 1\n000 系统配置文件 0'
    file.write(content)
    file.close()
    self.load_self_tree("", self.content.contlist,
self.content.contlist[0].name[0], "0", 1)
```

此三函数为对文件进行新建、删除、格式化，进行操作的同时要对目录列表、磁盘、树进行及时的更新与修改，要注意是否能够删除必要的配置文件。

```
def saveText(self):
    # INSERT  表示在光标处插入 self.w1.insert(END, "rrrr")
    result = self.w1.get("1.0", "end")  # 获取文本框输入的内容
    print(result)
    print(len(str(result)))

    file = open("./EM/EM.txt", "r", encoding='utf-8')
    content = file.read()
    global inichange
    name = inichange
    filename = 'FILENAMEIS ' + name
    pos = content.find(filename)
    addcontent = '\n' + result
    elsefile = content[pos + len(filename):]
    len1 = len('FILENAMEIS ')
    pl = 0
    for each in range(len(elsefile) - len1):
        if elsefile[each:each + len1] == 'FILENAMEIS ':  # 找出与子字符串首字符
相同的字符位置
            pl = each
            break
    elsefile = elsefile[pl:]

    if pos != -1:
        content = content[:pos + len(filename)] + addcontent + elsefile

    file = open("./EM/EM.txt", "w", encoding='utf-8')
    file.write(content)
    file.close()


def showf(self):
    print("showfile")
    for item in self.tree.selection():
        item_text = self.tree.item(item, "values")
        print(item_text)
        global inichange
        inichange = item_text[0]
        for node in self.content.contlist:
            if node.tag[0] == item_text[0]:
                if node.iscont[0] == '1':
                    messagebox.showwarning('Warning', '您仅可以查看文件信息！')
                    return 0

        file = open("./EM/EM.txt", "r", encoding='utf-8')
```

```
        content = file.read()
        filename = 'FILENAMEIS ' + item_text[0]

        pos = content.find(filename)

        elsefile = content[pos + len(filename):]

        len1 = len('FILENAMEIS ')
        pl = 0
        for each in range(len(elsefile) - len1):
            if elsefile[each:each + len1] == 'FILENAMEIS ':  # 找出与子字符串首字符相同的字符位置
                pl = each
                break
        elsefile = elsefile[1:pl - 1]
        print(elsefile)
        self.w1.delete('1.0', 'end')
        self.w1.insert(END, elsefile)

        file.close()


def writefile(self):
    f = codecs.open('memory.txt', mode='w', encoding='utf-8')  # 打开txt文件，以‘utf-8’编码读取

    for i in self.content.contlist:
        f.write(str(i.tag[0]) + " " + str(i.name[0]) + " " + str(i.iscont[0]) + "\n")

    f.close()
```

这三个函数较为重要，这是对位图、模拟磁盘以及文件存储进行修改操作，模拟读取以及修改保存文件。

点击类

```
class MyCollectApp(Toplevel):  # 重点
    def __init__(self):
        super().__init__()  # 重点
        self.title('用户信息')
        self.setupUI()

    def setupUI(self):
        row1 = Frame(self)
        row1.pack(side=TOP, fill=X)
        l1 = Label(row1, text="新建文件名与类型：", height=5, width=20)

        l1.pack(side=LEFT)  # 这里的side可以赋值为LEFT  RTGHT TOP  BOTTOM
        self.xls_text = StringVar()
        Button(row1, text="点击确认", command=self.on_click).pack(side=RIGHT)
        Entry(row1, textvariable=self.xls_text).pack(side=RIGHT)

        row2 = Frame(self)
        row2.pack(side=BOTTOM, fill="x")
```

```
        l = Label(row1, text="请输入文件名并选择新建文件类型", width=30, font=("楷
体", 12))
        l.pack()

        varchoice = StringVar()
        r1 = Radiobutton(row1, text='目录/文件夹',
                         variable=varchoice, value='file',
command=self.radioreturn1)
        r1.pack()
        r2 = Radiobutton(row1, text='.txt',
                         variable=varchoice, value='txt',
command=self.radioreturn2)
        r2.pack()

    def radioreturn1(self):
        global varfile
        varfile = ""

    def radioreturn2(self):
        global varfile
        varfile = ".txt"

    def on_click(self):
        # print(self.xls_text.get().lstrip())
        global FileName
        FileName = self.xls_text.get().lstrip()
        if len(FileName) == 0:
            # print("用户名必须输入!")
            messagebox.showwarning(title='系统提示', message='请输入新建文件
名!')
            return False
        self.quit()
        self.destroy()
        print("新建文件：%s" % (FileName))
```

模拟点击类是一个纯UI类，进行模拟点击文件及右侧目录弹窗的实现操作，主要有以上四个功能即新建等，这里command调用UI类的操作。