



QG工作室考核任务

**题 目 笔记管理系统**

**组** **别 嵌入式****第三小组**

2019年4月28日

目录

**[1. 项目简介](#_Toc5493_WPSOffice_Level1)** **[1](#_Toc5493_WPSOffice_Level1)**

**[2. 总体设计](#_Toc11896_WPSOffice_Level1)** **[1](#_Toc11896_WPSOffice_Level1)**

**[3. 详细设计](#_Toc5852_WPSOffice_Level1)** **[2](#_Toc5852_WPSOffice_Level1)**

[3.1. 用户登陆](#_Toc5852_WPSOffice_Level2) [2](#_Toc5852_WPSOffice_Level2)

[3.2. 返回上一级目录](#_Toc3485_WPSOffice_Level2) [3](#_Toc3485_WPSOffice_Level2)

[3.3. 跳转到目标路径](#_Toc23551_WPSOffice_Level2) [4](#_Toc23551_WPSOffice_Level2)

[3.4. 文件(文件夹)的移动](#_Toc20888_WPSOffice_Level2) [5](#_Toc20888_WPSOffice_Level2)

[3.5. 删除文件或文件夹](#_Toc9350_WPSOffice_Level2) [6](#_Toc9350_WPSOffice_Level2)

[3.6. 创建文件或文件夹](#_Toc20158_WPSOffice_Level2) [7](#_Toc20158_WPSOffice_Level2)

[3.7. 重命名文件或文件夹](#_Toc27775_WPSOffice_Level2) [8](#_Toc27775_WPSOffice_Level2)

[3.8. 对目标文件夹进行排序](#_Toc28084_WPSOffice_Level2) [9](#_Toc28084_WPSOffice_Level2)

[3.9. 树状显示所有文件和文件夹或所有文件夹](#_Toc2974_WPSOffice_Level2) [10](#_Toc2974_WPSOffice_Level2)

[3.10. 显示当前目录下所有的笔记文件、笔记本文件夹](#_Toc31345_WPSOffice_Level2) [10](#_Toc31345_WPSOffice_Level2)

[3.11. 显示命令指定文件夹下所有的内容](#_Toc3247_WPSOffice_Level2) [12](#_Toc3247_WPSOffice_Level2)

[3.12. 显示指定文件夹下的带搜索内容的文件以及文件夹名](#_Toc20610_WPSOffice_Level2) [13](#_Toc20610_WPSOffice_Level2)

[3.13. 为指定的文件或文件夹添加标签](#_Toc4431_WPSOffice_Level2) [14](#_Toc4431_WPSOffice_Level2)

[3.14. 为删除指定的文件或文件夹标签](#_Toc30281_WPSOffice_Level2) [15](#_Toc30281_WPSOffice_Level2)

[3.15. 显示指定的文件或文件夹标签](#_Toc11435_WPSOffice_Level2) [16](#_Toc11435_WPSOffice_Level2)

[3.16. 输出匹配tag的笔记文件名](#_Toc1862_WPSOffice_Level2) [17](#_Toc1862_WPSOffice_Level2)

[3.17. 根据标签名字查找所有该用户的符合标签的文件](#_Toc13603_WPSOffice_Level2) [18](#_Toc13603_WPSOffice_Level2)

**[4. 数据结构与数据处理](#_Toc3485_WPSOffice_Level1)** **[19](#_Toc3485_WPSOffice_Level1)**

[4.1. 数据结构](#_Toc3508_WPSOffice_Level2) [19](#_Toc3508_WPSOffice_Level2)

[4.2. 基础数据处理](#_Toc19447_WPSOffice_Level2) [19](#_Toc19447_WPSOffice_Level2)

[4.3. 用户输入时的数据处理](#_Toc11773_WPSOffice_Level2) [19](#_Toc11773_WPSOffice_Level2)

**[5. 程序亮点](#_Toc23551_WPSOffice_Level1)** **[19](#_Toc23551_WPSOffice_Level1)**

**[6. 程序兼容性说明](#_Toc20888_WPSOffice_Level1)** **[20](#_Toc20888_WPSOffice_Level1)**

**[7. 总结与体会](#_Toc9350_WPSOffice_Level1)** **[20](#_Toc9350_WPSOffice_Level1)**

# 项目简介

该项目是基于c语言设计的笔记管理系统，采用多文件编程，实现了模块化设计。

该程序主要是用于我们方便地管理自己的笔记。

实现的功能有：

多用户且可自定义自己的密码和工作目录的名字。

显示当前目录下所有的笔记本文件夹，以及笔记文件。

按照树状结构显示所有的笔记文件夹以及笔记文件。

按照树状结构显示所有的笔记文件夹。

显示命令指定文件下所有的内容(子笔记文件夹和笔记文件)。

将路径切换为当前目录的父目录.若是根目录则则不进行切换。

显示命令指定文件下所有的带有搜索内容的文件夹名及文件名。

将路径切换为命令中输入的文件夹路径。

将笔记文件移动到指定的文件夹目录下。

将笔记文件进行重命名，从原文件名改为目标文件名。

将指定的笔记文件夹移动到目标文件夹目录下。

对笔记文件进行删除。

删除制定文件以及其内部所有内容。

显示指定笔记/文件夹的标签。

为指定笔记/笔记文件夹增加标签。

删除指定笔记/文件夹的指定标签。

根据标签内容，在当前目录下进行笔记搜索，输出匹配的标签所对应的笔记文件名。

根据标签内容，对所有笔记搜索，输出匹配的标签所对应的笔记的文件名以及绝对路径。

创建笔记文件夹或笔记文件。

对指定文件夹内的笔记文件进行排序。

返回登陆界面，退出程序，帮助界面（输出所有可用的命令，以及命令的格式）。

# 总体设计

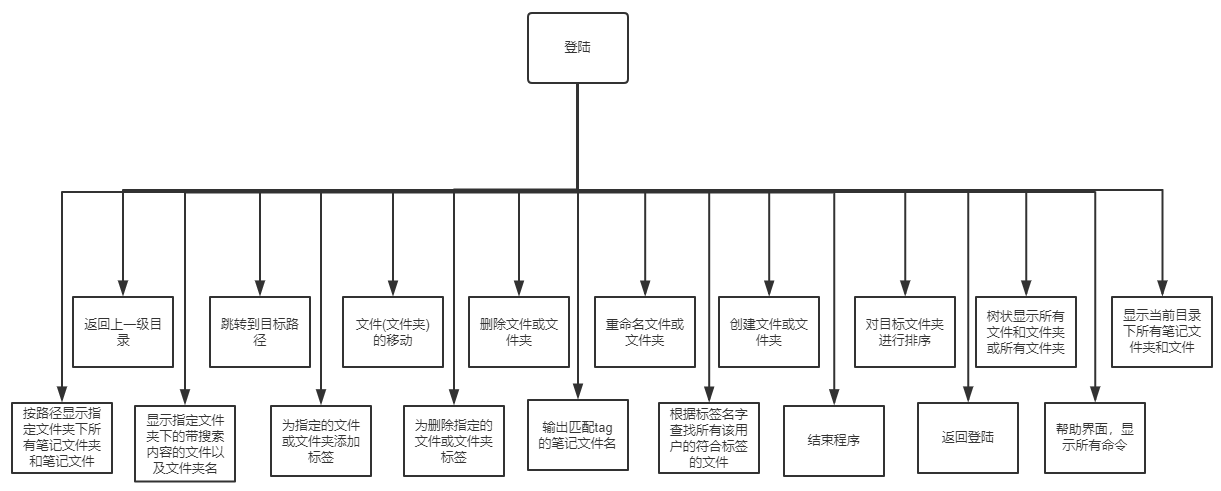


图 1 总体设计

登陆系统时，若输入不存在的用户名则认为用户要新建一个用户，则要求输入密码，用户的工作目录。若输入存在的用户名，则需要输入对应用户正确的密码才可以登陆成功。

该系统存储文件信息的方式为，在每一个文件夹下都建一个save.txt文件，以链表的方式存储文件信息，每个节点存储有：文件的名字，对应文件夹与非文件夹的判断条件，对应标签的数量以及所有标签的名字（链表的方式存储）。录入文件信息与取出文件信息为统一的两个函数Input\_Save与Read\_Save。

# 详细设计

## 用户登陆

流程图如下：

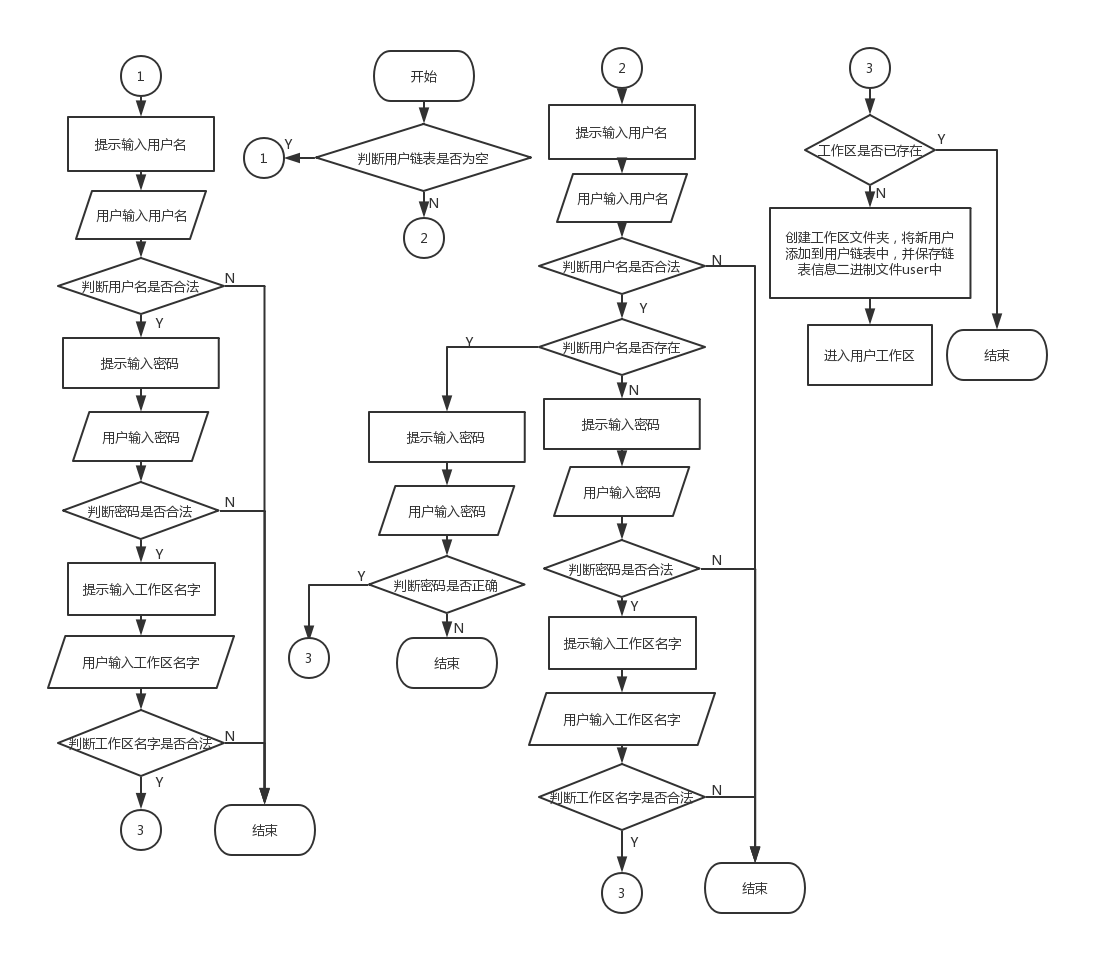


图 2 用户登陆

先对用户链表进行判断，如果为空，新建一个用户结点，请求输入用户名、密码、以及工作区名字，判断是否都是合法字符，如果不是，退出登陆，回到主界面；如果是，增加结点到用户链表中，同时保存链表信息到二进制文件**use**r中，进入该用户的工作区。

如果不为空请求输入用户名，如果用户名已存在请求输入密码，密码正确进入该用户工作区，错误返回主界面。如果用户不存在，请求输入密码和工作区名字，判断工作区名字是否存在，如果存在，提示工作区已存在并返回主界面，否则新建一个用户结点，保存该用户信息，增加结点到用户链表中，同时保存链表信息到二进制文件**user**中，进入该用户的工作区。

## 返回上一级目录

流程图如下：

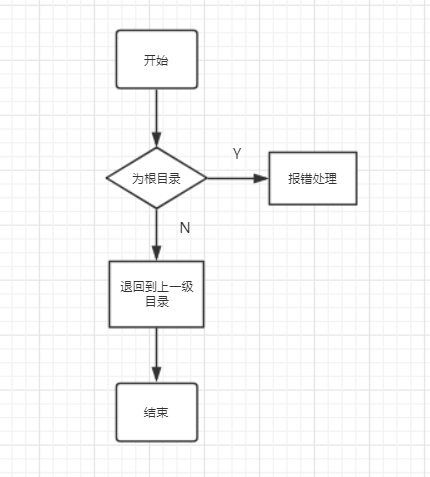


图 3 返回上一级流程图

当判断当前目录不是用户的根目录时，就执行返回上一级目录的操作，对记录当前层所有文件(文件夹)的链表和当前目录信息进行更改。当然，如果，当前是用户的根目录，就报错并跳出该函数。

## 跳转到目标路径

流程图如下：

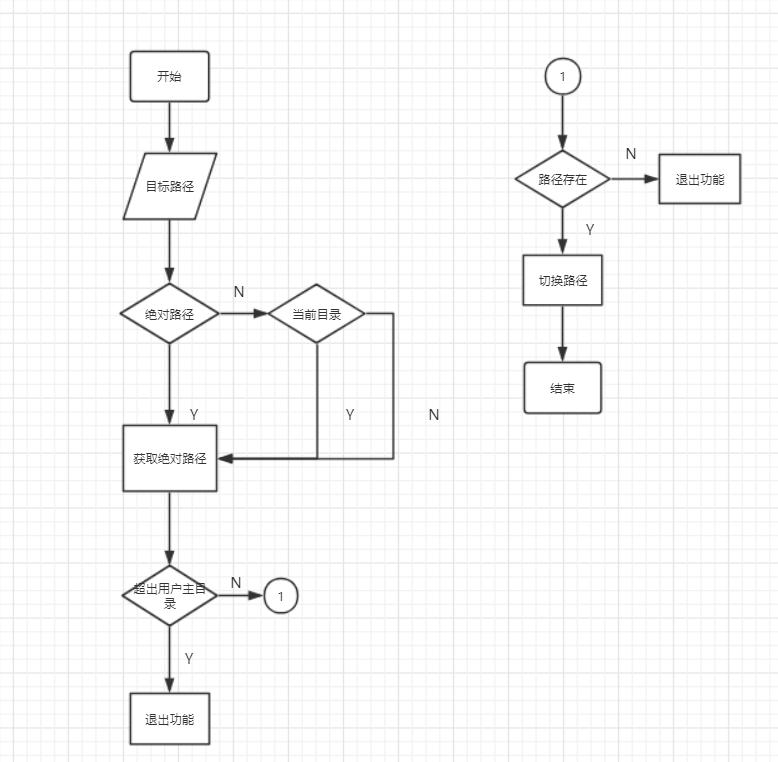


图 4 跳转到目标路径流程图

传入的路径可能为绝对路径、相对路径、当前目录下的路径，而我则希望将它们全部转化为绝对路径。绝对路径本身可不作处理。相对路径需要先找到文件夹名，然后在当前目录之后加上文件夹名就得到了绝对路径。当前目录下的路径，只需要用当前目录加上它就得到了绝对路径。

之后需要判断用户的操作是否超出了自己的根目录。由于有用户根目录的记载，所以只需将跳转路径的根目录与当前根目录比较，相同则允许，不同则报错并跳出该函数。再经过一系列的路径存在、路径名称检查之后，就可以读到待改变路径的链表去，从而实现路径的跳转。

* 1. **文件(文件夹)的移动**

流程图如下：

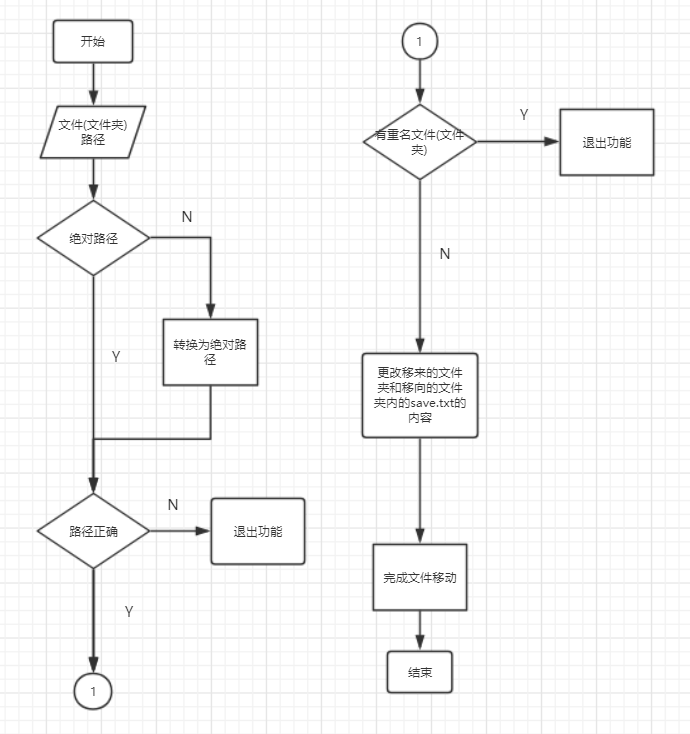


图 5 移动文件(文件夹)流程图

对于作为参数传递进来的文件(文件夹)路径，其有可能是绝对路径或相对路径，由于对绝对的路径的操作会方便很多，所以，先选择将传入的相对路径转化为绝对路径，而对于传入的绝对路径不作更改。此时认为传入的绝对路径一定包含盘符(**C**、**D**、**E**、**F**)和冒号(**:**)，不包含这两个部分的一律认为是在当前目录下。经过检测，对传入的绝对路径不作任何更改，而对于传入的当前目录下文件名，则在其前面加上当前用户所在的路径使它成为绝对路径。在这里，待移动的文件(文件夹)和移向的文件夹是作同等处理的，检测和转化的过程一致。在完成了文件(文件夹)路径的转化之后，检查该路径是否存在。如果不存在就报错并返回。

利用路径截取信息后，得到文件名，将文件名和移动到的路径链接，然后判断是否在移向文件夹中存在同名文件(文件夹)了。文件夹中有重名文件(文件夹)，报错并返回。

在移动之前，更改移来和移向文件夹的**save.txt**文件。更改方法为用所得的两个路径按顺序打开里面的save.txt文件夹，用链表存储信息，更改链表后再存入**save.txt**。

最后，进行文件(文件夹)的移动，利用**system**(**move** **待移动文件(文件夹)的绝对路径 移向文件夹的绝对路径**)来实现。

## 删除文件或文件夹

流程图如下：

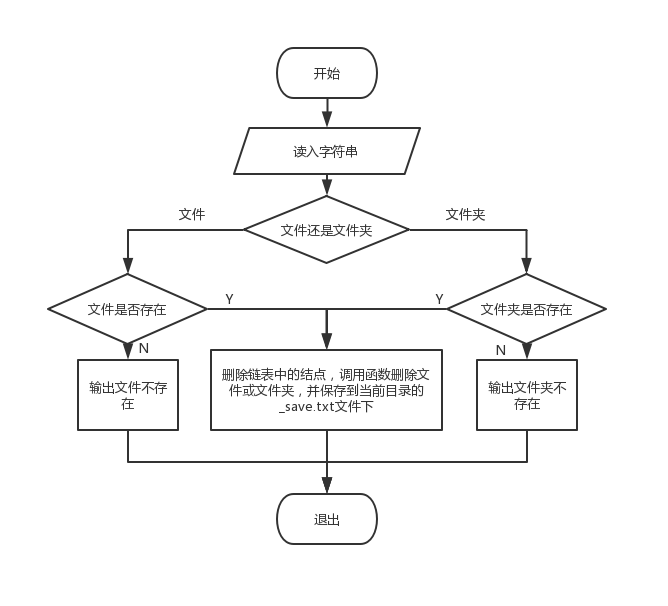


图 6 删除文件与文件夹

以文件为例：

外部判断后传入到函数DeleteFOF(CurFloder \*current, char \*s)执行删除功能，读入字符串，并通过是否有字符’.’来标记文件或是文件夹，新建字符串path复制当前路径并加上文件名调用函数RoteExistence()判断文件是否存在，不存在输出对应信息退出函数,如果存在并且是文件，使用字符串"del "加上当前路径调用system函数删除对应文件,并删除对应结点，保存当前的目录链表信息到当前目录的\_save.txt文件下，退出函数结束。

文件夹同理，删除时使用字符串"rd /s "加上当前路径调用system函数删除对应文件夹,并删除对应结点，保存当前的目录链表信息到当前目录的\_save.txt文件下，退出函数结束。

## 创建文件或文件夹

流程图如下：

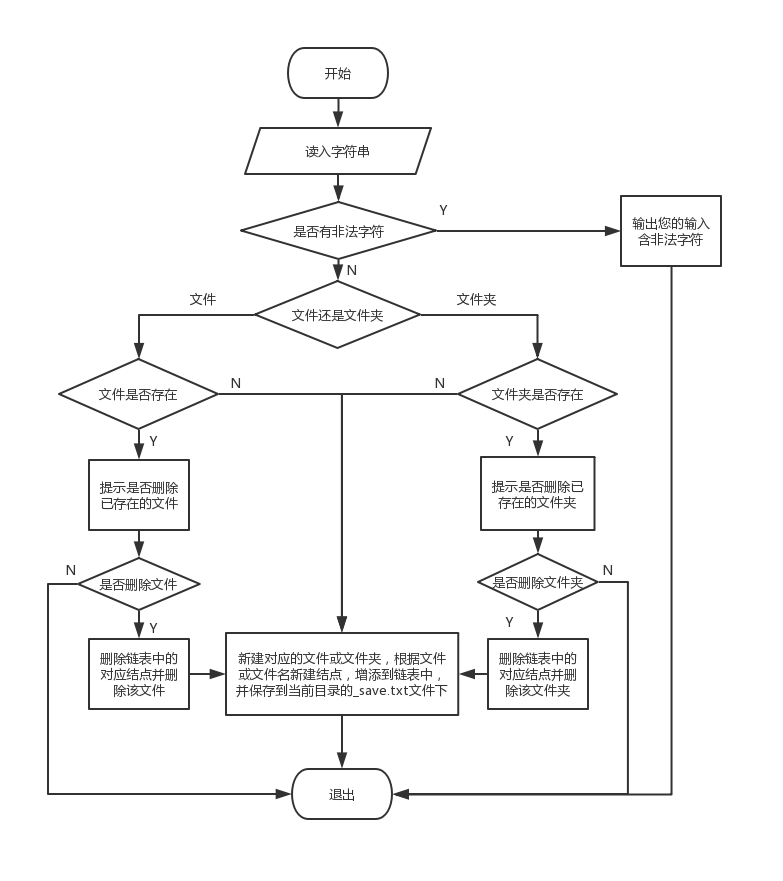


图 7 新建文件或文件夹

以文件为例：

读入用户的字符串,调用函数MakeFOF(CurFloder \*current, char \*s)执行新建功能，读入字符串，并通过是否有字符’.’来标记文件或是文件夹，根据文件名创建一个新的结点e。新建字符串path复制当前路径并加上文件名调用函数RoteExistence()判断文件是否存在，如果已存在询问是否覆盖掉原文件，如果不是就退出函数，如果是就删除已存在的文件，删除文件对应的结点，并把已经创建好的结点e加入链表中，使用fopen函数用’w’写入模式新建文件，保存当前的目录链表信息到当前目录的\_save.txt文件下，退出函数结束。

文件夹同理，如果是删除已存在的文件夹，就删除文件对应的结点，并把已经创建好的结点e加入链表中，创建的话使用字符串"md "加上当前路径后调用system函数新建文件夹，保存当前的目录链表信息到当前目录的\_save.txt文件下，退出函数结束。

如果文件或文件夹不存在时直接创建并保存信息。

## 重命名文件

流程图如下：

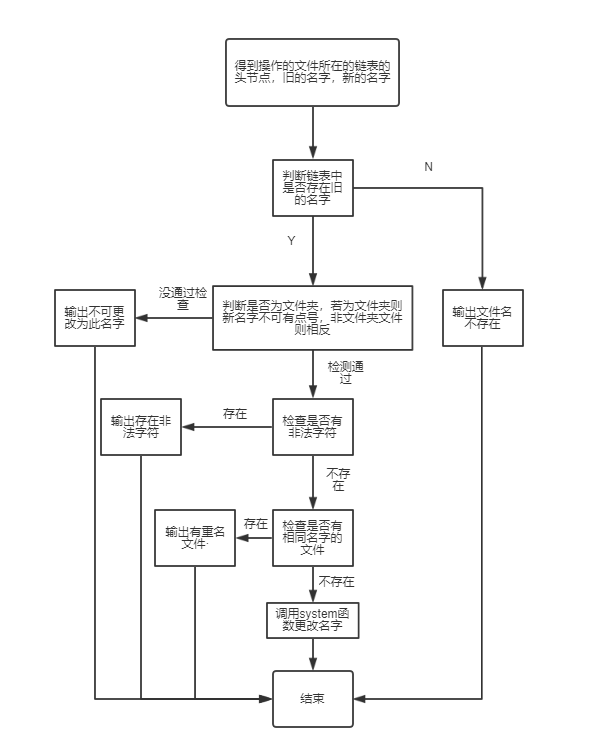


图 8 重命名文件

先从主函数得到存有当前路径下所有文件信息的链表的头节点，以及用户输入的两条有用信息，要更改的名字，与更改后的名字。先判断要改名的文件是否存在，若不存在，提示文件不存在，若存在，开始判断更改的文件是文件夹文件还是非文件夹文件，文件夹文件则不允许重命名中含有.这个符号存在，非文件夹文件则相反，然后判断更改后的名字与当前文件夹内名字是否有冲突，如果有提示有重名文件，若无冲突，开始判断新名字是否存在非法字符 <>\/:?\*|”与空格，若不存在非法字符，则更改传入的链表内相应节点内存储名字，再将该链表存进相应的文件内，最后调用system函数更改文件名字。

## 对目标文件夹进行排序

流程图如下：

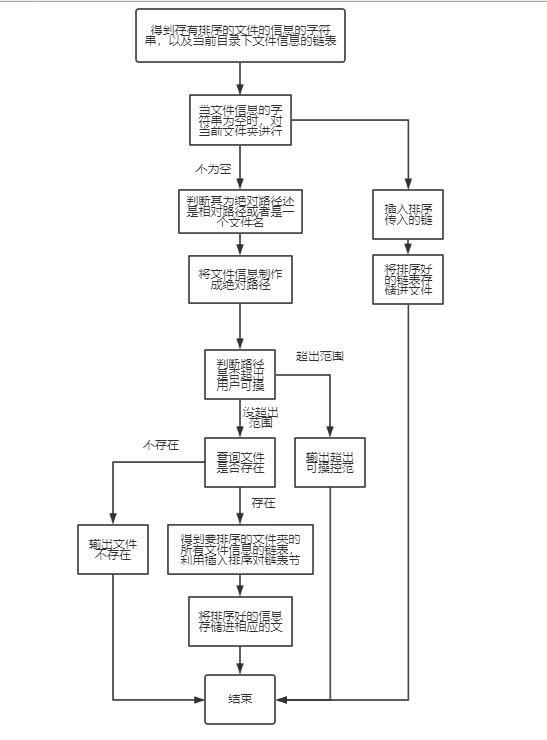


图 9 对目标文件夹进行排序

先从主函数得到要存有查找的文件夹信息的字符串，与存有当前目录所有文件信息的链表的头节点。先判断传入的字符串为绝对路径还是相对路径或者是文件夹名字，然后将其制作成绝对路径，判断路径正确，后利用该路径打开存有文件信息的文件夹制作成链表后，利用插入排序对其存有的名字信息的字符串的节点进行排序，然后该链表存入对应的文件中。

## 树状显示所有文件和文件夹或所有文件夹

流程图如下：

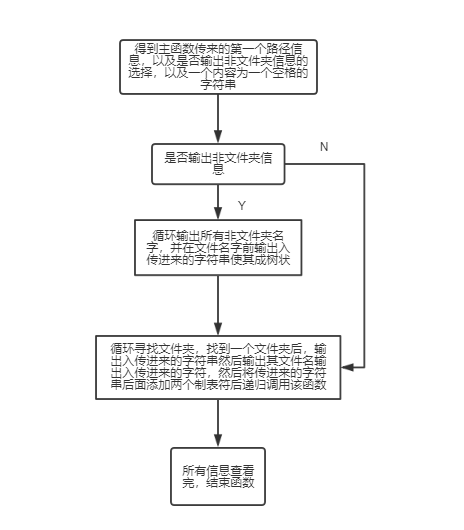


图 10 树状显示所有文件和文件夹或所有文件夹

此功能用递归遍历文件树实现，用int型变量做形参判断是否输出非文件夹文件，两个字符串一个存储这一次输出的文件的路径，一个存储输出信息前的空格的数量。先传入第一个路径，根据这个路径得到遍历的根节点，先输出所有文件名，然后再查找该路径下的文件夹文件，输出其名字然后再在其名字后面输出一串横线，然后进入该文件夹，重复上面的操作，直到没有文件夹存在。

## 显示当前目录下所有的笔记文件、笔记本文件夹

流程图如下：

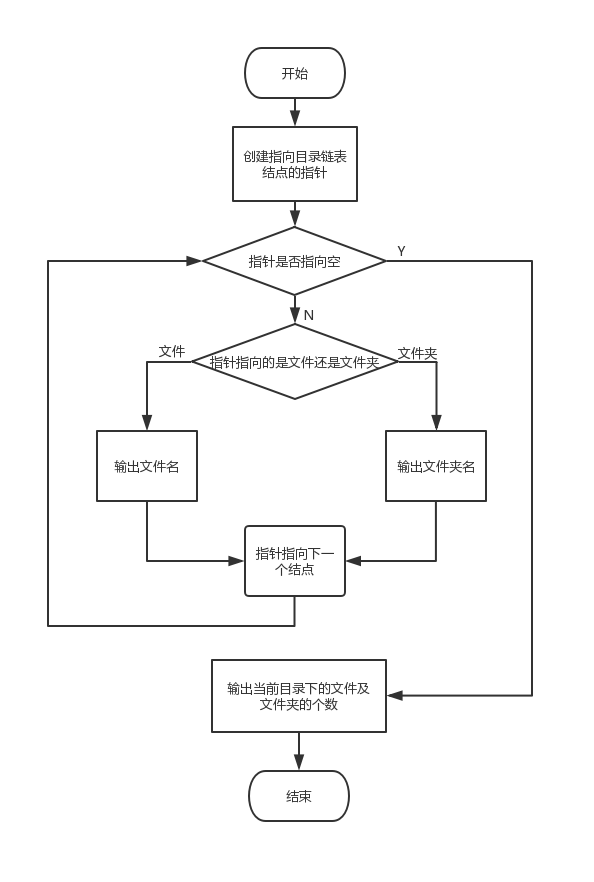


图 11显示当前目录下所有的笔记文件、笔记本文件夹

遍历当前的目录链表，根据链表节点的标记flag判断结点代表是文件还是文件夹，然后打印文件或文件夹名，最后输出当前目录下的笔记文件以及文件夹个数。

## 显示命令指定文件夹下所有的内容

流程图如下：

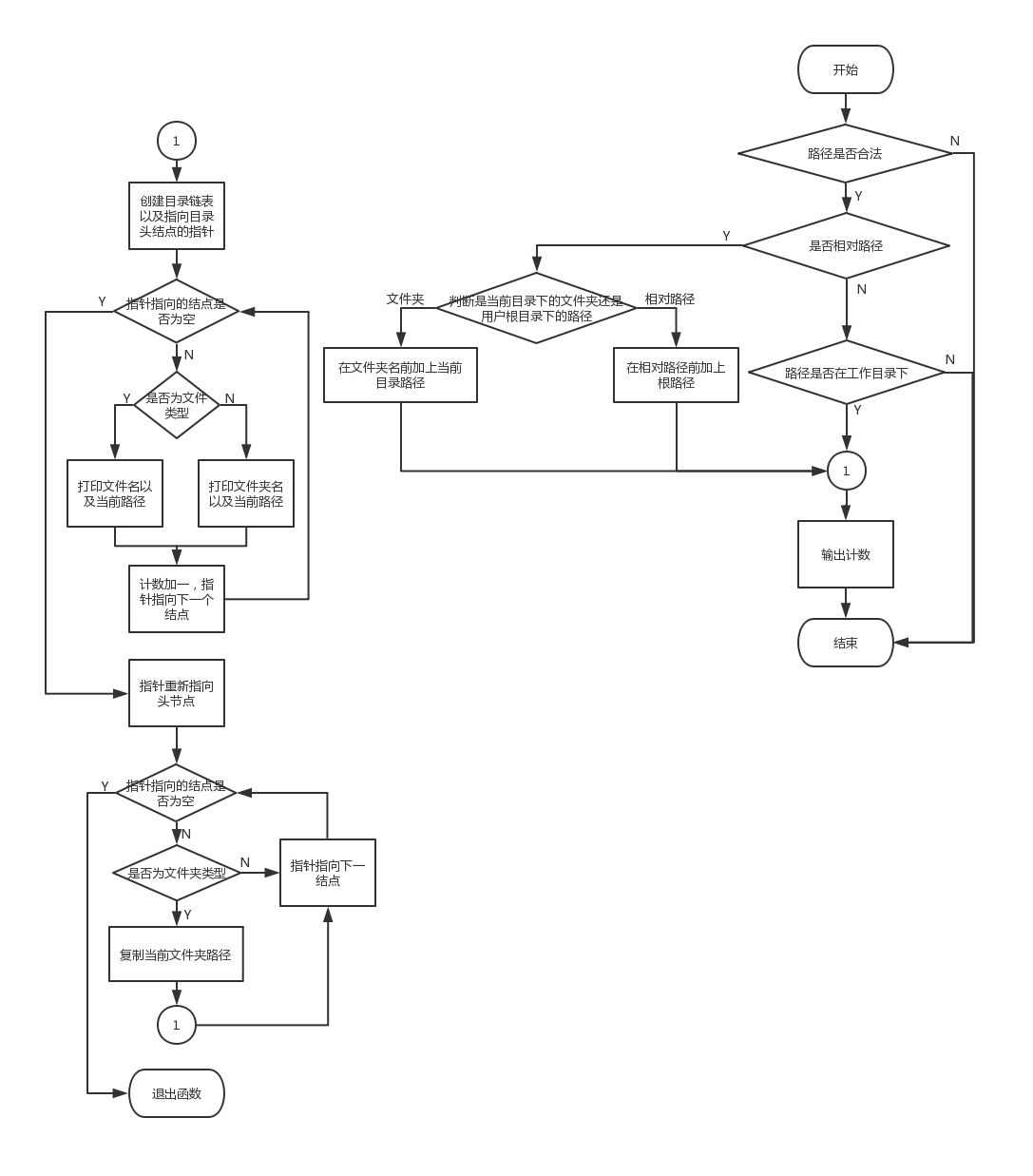


图 12显示命令指定文件夹下所有的内容

先对读入的路径判断是否合法，然后补全成绝对路径，调用函数RecursionPrin(char \*prinpath, int \*sum)打印，输出sum[[1]](#footnote-0)。

RecursionPrin(char \*prinpath, int \*sum)函数读入路径和sum,根据路径调用Read\_Save()函数使用该路径下的\_save.txt文件创建目录链表，然后先遍历链表打印一遍，然后再遍历一遍链表，如果是文件夹就递归调用RecursionPrin()函数，将当前文件夹的路径作为参数，最后销毁链表，退出函数。

## 显示指定文件夹下的带搜索内容的文件以及文件夹名

流程图如下：

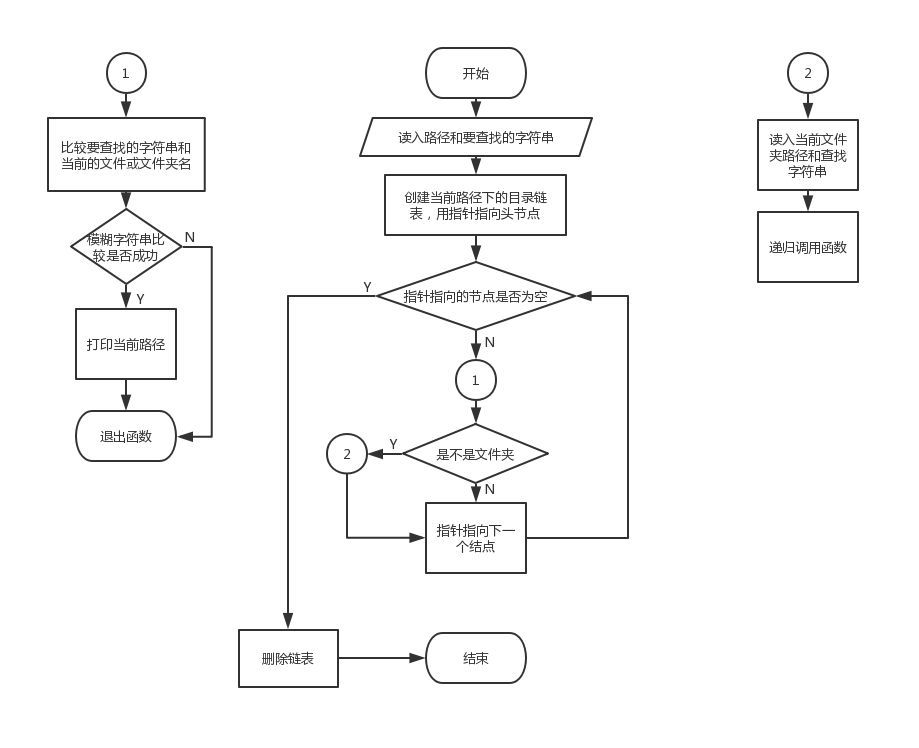


图 13 显示指定文件夹下的带搜索内容的文件以及文件夹名

通过使用函数Read\_Save(CurFloder \*read, const char \*filerote)从当前路径下的\_save.txt文件读取信息并创建链表，新建一个指向结点类型的指针，使用该指针遍历链表，遍历结束销毁链表，退出函数。

指针每指向一个结点调用函数PrintGrep(FOF \*current, char \*path, char \*Find)来决定是否打印信息，在函数里面调用Checkstr()来比对文件名和搜索内容，返回值为真就打印信息，为假就退出PrintGrep()函数。下一步判断此节点是否为文件结点，如果是的话就递归调用函数，并把当前的这个文件结点对应的的文件夹的路径和搜索内容当作参数传入。

# 为指定的文件或文件夹添加标签

流程图如下：

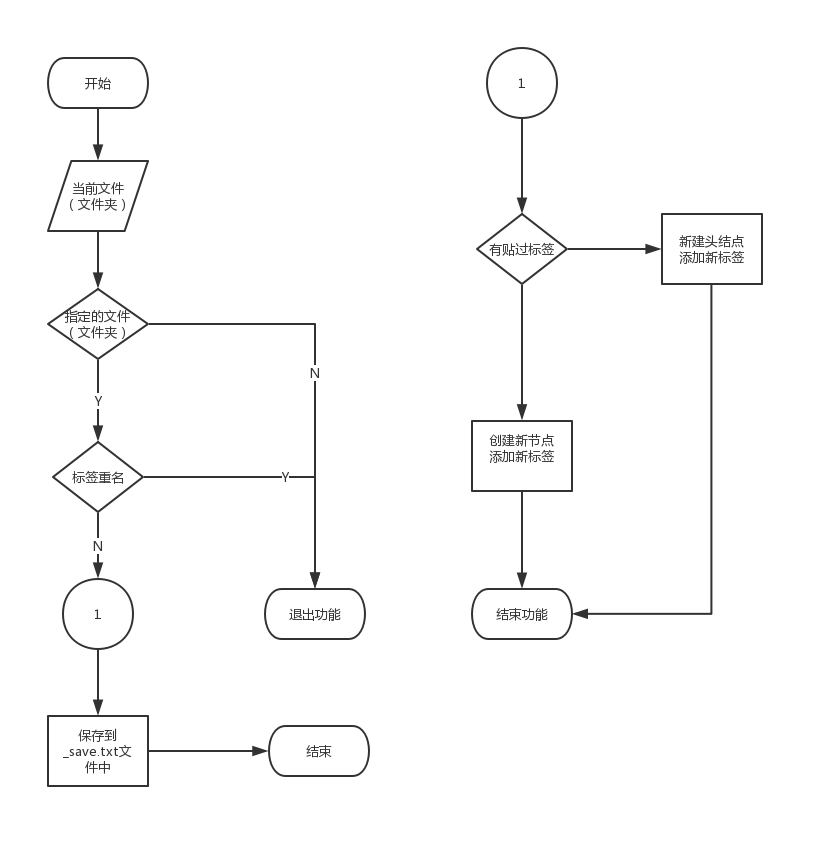


图 14 为指定的文件或文件夹添加标签

该功能并没有将文件(文件夹)路径作为参数进行传入，因为那样操作有些复杂，为了简单化，就将当前文件信息结构体的头指针，好处就是不需要对路径进行改变，同时能更好的兼容该系统。通过传入的当前文件信息结构体的头指针进行读取当前目录下所有文件（文件夹的信息），包括标签内容以及标签数量。同时在用户添加标签的过程中，还加了一个判断，以防该文件（文件夹）中的标签重名。

若是不重名，就进入添加标签函数。在该函数中，增添一个判断，判断用户所指定的文件（文件夹）此前是否有贴过标签。有则直接在其后面新建一个标签节点，无则创建一个新的标签头节点。

最后对用户指定的文件（文件夹）进行添加用户输入的标签，将文件（文件夹）的信息保存到save.txt中。

# 为删除指定的文件或文件夹标签

流程图如下：

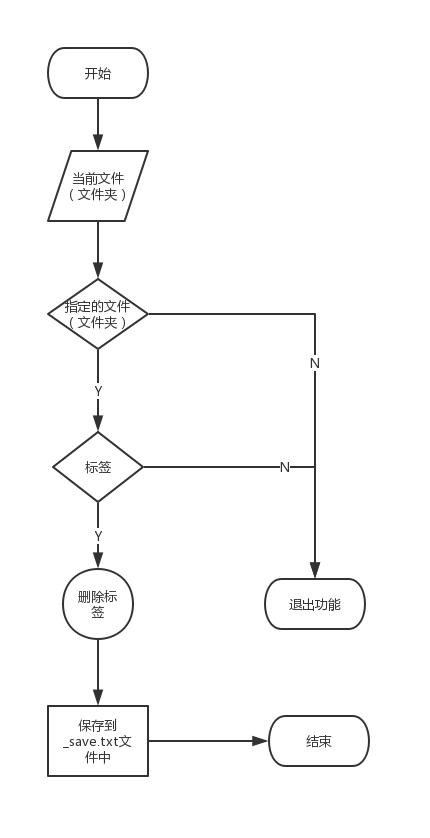


图 15 为删除指定的文件或文件夹标签

该功能跟为指定的文件或文件夹添加标签功能相似，也是没有将文件(文件夹)路径作为参数进行传入。原因与上面所说的一样。因为那样操作有些复杂，为了简单化，就将当前文件信息结构体的头指针，好处就是不需要对路径进行改变，同时能更好的兼容该系统。通过传入的当前文件信息结构体的头指针进行读取当前目录下所有文件（文件夹的信息），包括标签内容以及标签数量。

然后依次遍历当前目录下所有的文件（文件夹），找到用户指定的文件（文件夹），在指定的文件（文件夹）中，依次遍历所有标签，找到用户所要删除的标签，进行删除

最后对用户指定的文件（文件夹）进行删除用户输入的标签，将文件（文件夹）的信息保存到save.txt中。

# 显示指定的文件或文件夹标签

流程图如下：

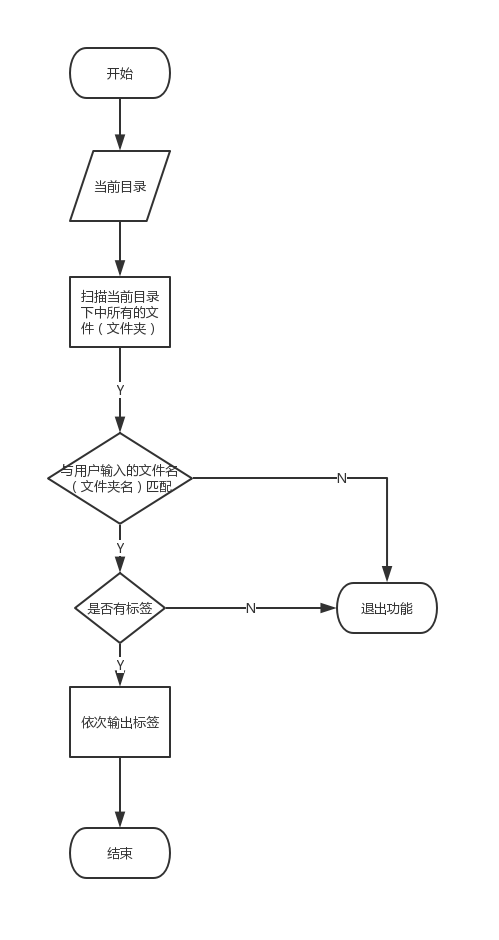


图 16显示指定的文件或文件夹标签

该函数仅是将当前目录下，遍历寻找用户指定的文件（文件夹），读取当前目录的save.txt文件，通过参数当前文件信息结构体的头指针传入，文件信息是通过单向链表存储的。我们将其依次遍历，并与用户输入的文件名（文件夹名）进行匹配。寻找用户指定的文件（文件夹）。

然后判断该文件（文件夹）节点中是否存储标签链表头结点。无则退出，有则从标签头结点依次读取标签，将其依次显示出来。

# 输出匹配tag的笔记文件名

流程图如下：

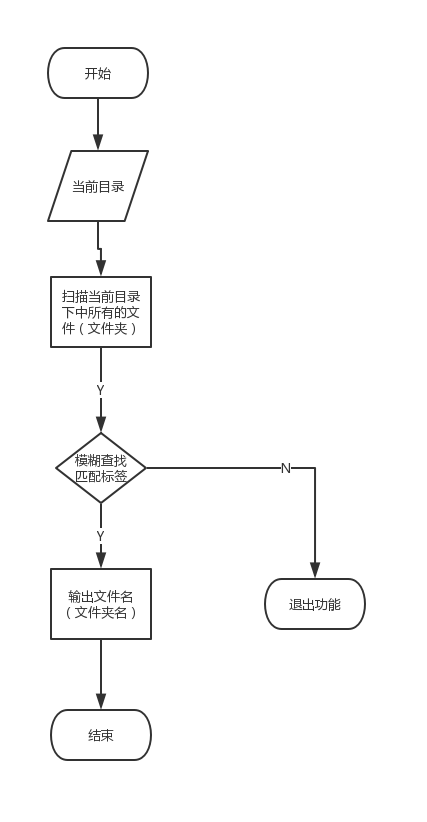


图 17 输出匹配tag的笔记文件名

该功能同样的没有将文件(文件夹)路径作为参数进行传入。通过传入的当前文件信息结构体的头指针进行读取当前目录下所有文件（文件夹的信息），包括标签内容以及标签数量。

然后依次遍历当前目录下所有的文件（文件夹），找到用户指定的文件（文件夹），在指定的文件（文件夹）中，依次遍历所有标签，找到用户所要寻找的标签。

最后输出该标签所在的文件名（文件夹名）。

# 根据标签名字查找所有该用户的符合标签的文件

流程图如下：

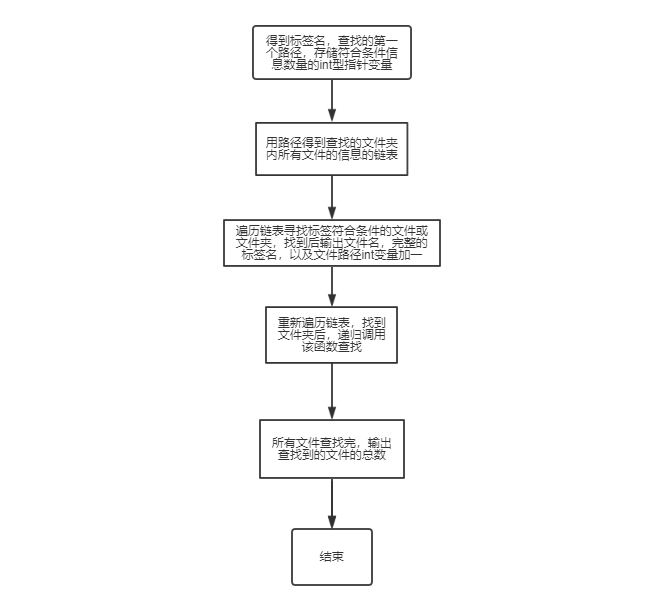


图 18 根据标签名字查找所有该用户的符合标签的文件

先从主函数中得到要查找的标签，以及要查找的根目录的路径，以及记录查到的文件数量的int型变量。利用传入的路径打开存有文件信息的文件，取出信息做成一条链表，再在链表中查找标签中存在有一段信息与传入的标签信息相吻合的文件或文件夹，然后输出该文件的名字与路径，int型变量加一。查找完后，重新遍历链表，查找其中的文件夹，然后进入该文件夹，重复上面的操作，直到遍历完整个文件树。

# 数据结构与数据处理

## 数据结构

主要以链表为主，分成三类链表，第一类是结点类型为USER的用户链表，第二类是结点类型为File\_Or\_Floder的目录链表，第三类是结点类型为TAG的标签链表。

用户链表结点成员为name,password,root,next,分别对应用户的用户名，密码，工作区名字，以及指向下一个结点的指针。

目录结构体CurFloder下只有一个成员，就是以File\_Or\_Floder结构体为节点类型的指针，用来存放当前目录的目录链表的头指针。

目录链表结点成员为flag,name,num,taghead,next。flag是文件和文件夹的标记，1为文件，2为文件夹，name是文件或文件夹名字，num是该文件或文件夹的标签数量，taghead是指向标签链表的头指针，next是指向下一个结点的指针。

标签链表结点成员只有info和指向下一结点的指针，info存储标签信息。

文件信息的存储为，在每一个目录下创建一个\_save.txt的文件，存储该文件夹下所有的文件信息（标签，名字，标签数量，文件夹或文件的判断条件），每次进入一层目录时，就会从\_save.txt中取出文件信息。

用户信息（密码，账户名，工作目录）用名叫user的文件存储，登陆时提取出信息，以确认登陆。

## 基础数据处理

程序在运行时根据二进制文件创建用户链表，在用户登陆时可能新增用户，此时会同步保存到用户链表以及二进制文件中。进入工作区之后根据工作目录文件夹下的\_save.txt文件创建目录链表和标签链表，并且同步更改链表和\_save.txt的内容。

## 用户输入时的数据处理

先根据用户的输入判断是否为Return,Close,help，这三个命令，如果是，则进行相应处理，如果不是，继续检测输入的字符串的前缀，选择相应的判断语句，如果不属于任何一种判断语句，则提示命令输入错误，若找到，则将用户输入的有用信息分别存储下来，放到对应的函数中，然后再在函数中判断这些信息是否为正确信息。若信息正确则进行相应操作，否则提示出错。

# **程序亮点**

增加了新增了创建文件的功能。

增加了直接退出程序的功能（Close）。

增加了重新登陆的功能（Return）。

增加了显示所有命令的功能(help)。

增加了树形显示所有文件夹的功能（ls -a1）。

许多功能都支持绝对路径或相对路径或者直接在当前目录下操作，减少了用户的输入。

# 程序兼容性说明

Win10下可完美运行。程序不可放置于电脑用户目录下，可能会存在无权限移动文件的问题。

请勿将程序放置在带空格的目录下，不然程序无法正常运行。

# 总结与体会

陈澎炜：

对于此次的大考核，收获还是蛮多的吧，跟几位大佬一起合作，虽说许多难点都让大佬们解决的，我只是小打小闹，不过我还是多少有学到一点东西。例如，在设计整个系统工程，大概的步骤有哪些，在开始设计前需要做什么。这些在大考核之前，我都只是一个模糊的概念，并不是很清楚的知道，数据结构是那么的重要。在设计整个系统之前，若是没有确认好采取什么数据结构进行存储，后期是会出现混乱的。这样我想起当初我自己一人设计图书管理系统的时候，老是改变数据的存储结构，最终的效率十分慢，而且做出来的系统也十分混乱。逻辑性没有处理好。

其次，不仅从大佬们上学到了一些东西，还从整个团队合作中学到了合作是多么重要，假如你一句，我一句的，没有得到统一的话，到后期是什么困难的。因为整个设计思维没有统一，每个人写的数据结构也不同，根本就很难用到一起，更别说要将几个人所写的函数整合成一套系统这是不可能的。所以在团队合作中，设计思维及思想的统一是极其重要的。

虽然，我在整个团队合作中仅仅做了一些简简单单的活，甚至是那种让我感觉起来是可无可有的那种。但我还是用心做了，我承认我实力不够，甚至设计一个简简单单的功能还出现这么多bug。所以最后我也不抱怨什么，只恨自己实力不够，让我见识到在软件方面与大佬们的差距，还是需要加把劲，尽量缩短这之间的距离吧。

陈余：

第一次和别人一起打代码，体验到了许多和一个人打码的不同之处，得到了很多锻炼。比如说团队合作比一个人做系统更加容易完美，因为一个人是很难把每个细节都掌控好的，但是经过团队合作，相互交流就可以更容易地把系统做得更完美，比如说张海森同学就给系统找出了不少的隐藏的小bug;打码时想到了各种各样的优化，但是具体结构已经定下来了，所以无法轻易更改，只能继续思考如何在已经定好的结构下进行相同作用的优化，这让我的思维得到了极大的锻炼；在想好了系统该怎么做，然后把想法告诉其他队员时，别人不一定会像自己理解地那样透彻，有些地方会出现分歧，所以不可以笼统地说完就结束了，还要把每个细节支末都讲清楚，而且要尽快把重要的函数写好把整体的数据结构规定好。

林文琛：

我学习到了团队合作是需要磨合和队员之间相互包容的，不能出现巨大分歧和矛盾，在合理的情况下可以提出自己建议，但应以团队为主。在讨论交流中发现错误，要积极地去改进。我们小队的每个人都努力地去完成自己的任务，虽然有些地方可能不如预期，但是我们还是努力的去做了，不放弃，付出，我觉得我们小队还是很棒的(夸夸群群主就是我)。

张海森：

在这次笔记管理系统的合作中，我认为，我学到了：在自己开始动手写程序之前一定要与其他人交流好，弄明白自己的函数的功能是什么、可以得到什么参数、需要输出哪些并有哪些要完成的操作；当大家一起合作的时候，要事先商量好大家一起使用的数据结构、参数、共用函数，方便大家有共同的交流语言；当你弄不清楚对方的意图时，一定要与对方多交流，而不是闷声埋头苦干；不要低估一个团队的力量，大家一起可以做出你意想不到的事情。

1. RecursionPrin()函数中打印的文件和文件夹的数目 [↑](#footnote-ref-0)