目 录

1. 引言

为了对计算机操作系统内核的理解，提高对操作系统内核的分析与扩展能力。操作系统是管理计算机系统的全部硬件资源包括软件资源及数据资源、控制程序运行、改善人机界面、为其它应用软件提供支持等，使计算机系统所有资源最大限度地发挥作用，为用户提供方便的、有效的、友善的服务界面。操作系统是一个庞大的管理控制程序，大致包括 5 个方面的管理功能:进程与处理机管理、作业管理、存储管理、设备管理、文件管理。本文主要实现了模拟生产者消费者以及文件管理系统。

信息化时代电子文件管理系统正逐渐应用于电子文件管理的各种活动中。通过电子文件管理,不仅可以将纸质档案和数据信息进行录入,还可以将电子文件进行收集,也可以连接电子文件与办公自动化系统,将大批量的电子文件直接转入所需的档案数据库中。[1]本通过编译器对文件管理系统进行了模拟。

1.1 任务要求

本文中实现的小型文件系统能够对文件进行模拟管理，具体包含以下几个功能：

（1）用户登录/退出操作；

（2）进行目录转换操作；

（3）新建文件夹/文件操作；

（4）文件上锁功能，即某用户自己创建并上锁的文件/文件夹，无法被其他用户访问；

（5）文件夹/文件名的修改操作；

（6）文件夹/文件具体信息查看操作。

1.1　选题

1.2.1 生产者—消费者问题

（1）通过 Java 语言中的 wait 和 notify 命令模拟操作系统中的 P/V 操作；

（2）为每个生产者 / 消费者产生一个线程，设计正确的同步算法；

（3）每个生产者和消费者对有界缓冲区进行操作后，即时显示有界缓冲区的当前全部内容、当前指针位置和生产者 / 消费者线程的自定义标识符；

（4）生产者和消费者各有两个以上；

（5）多个生产者或多个消费者之间须共享对缓冲区进行操作的函数代码。

1.2.2 文件系统

设计一个多用户文件系统，理解文件系统的层次结构，完成基本的文件系统create、open、close、read/write等基本功能，并实现文件保护操作。实现以此为基础加入自己设计功能的小型文件系统。

2 需求分析与设计

2.1 文件管理系统

该文件管理系统主要正对一批用户使用同一块文件存储区。每个用户自己创建并上锁的文件/文件夹无法被其他用户打开或修改。除“文件上锁”之外，系统害包括了目录转换、查看文件信息、修改文件内容、删除文件、重命名文件等功能。

2.1.1 系统框架和流程

文件管理系统流程主要分为以下几点：

（1）用户输入用户名进入文件管理系统；

（2）用户进入文件管理系统后，根据终端提示命令来执行操作，具体内容包括以下十个功能，返回上级目录，新建文件夹，新建文件，转到下一级目录，删除文件/文件夹，修改文件名称，查看文件信息，写入文件，文件上锁；

（3）执行结束后，用户可以退出自己的账号。

流程图如图2-3所示。

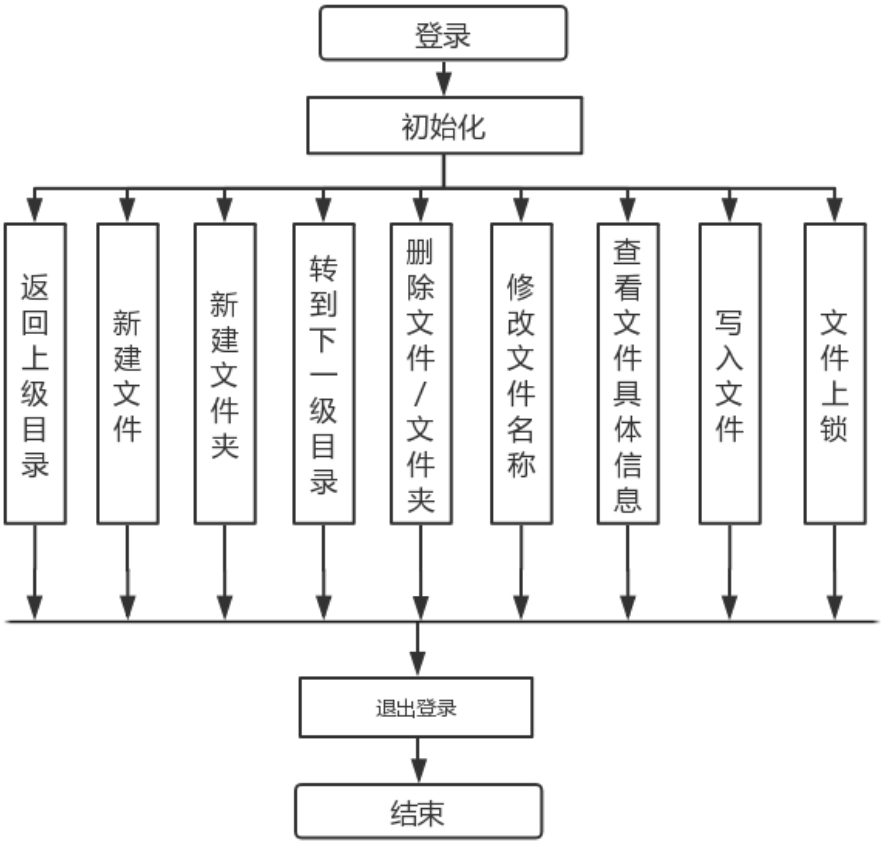

图 2-3 文件管理系统流程图

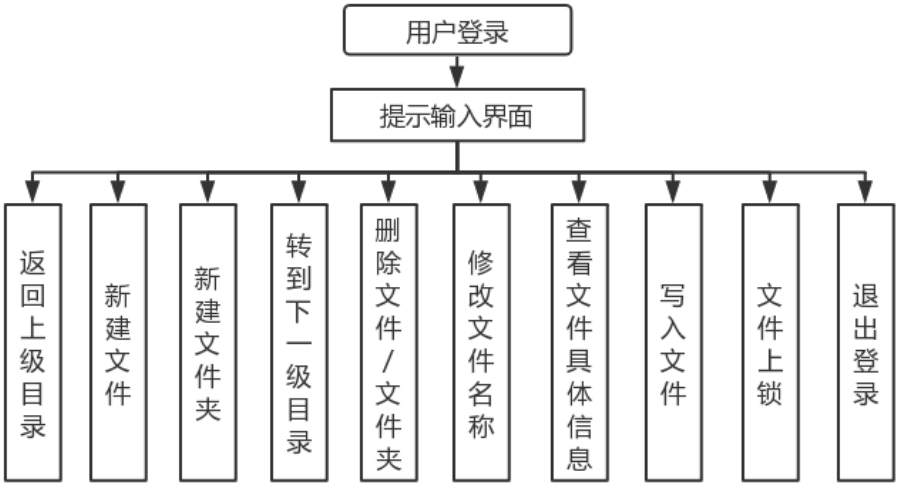

图 2-4 文件管理系统系统框架图

3 数据结构

3.1 用户登录模块

在用户登录模块中，对应相关数据为字符数组型的全局变量，定义如下：

```
char username[10];
```

3.2 其他文件操作模块

在其他文件以及目录相关操作模块中，如返回上级目录、新建文件（夹）、转到下级目录、删除文件（夹）、修改文件名称、查看文件具体信息以及文件上锁模块。使用到的相关数据为文件节点的相关数据结构体。具体定义如下：

```
typedef struct fileInfo {
    int data;                           //层数
    struct fileInfo *Child;             //子文件
    struct fileInfo *Brother;           //子文件
    struct fileInfo *Parent;            //父文件
    char filename[11];                  //文件名
    char attribute;                     //文件属性字段，0：目录文件，1：数据文件
    char time[20];                      //文件创建时间
    char content[100];                  //文件内容
    char createAuthor[10];              //文件作者
    int lock = 0;                       //文件上锁
} Node;
```

具体操作时，在全局定义了根节点以及标记当前所在文件位置的标记节点，具体定义如下：

```
Node* node=(Node*)malloc(sizeof(Node));          //创建一个根节点
Node* i_node=(Node*)malloc(sizeof(Node));        //标记当前位置的节点
```

4 关键技术

4.1 二叉树

二叉树是一种重要的非线性数据结构,在计算机领域有着广泛的应用。[3]在计算机科学中，二叉树是每个结点最多有两个子树的树结构。通常子树被称作“左

子树”（left subtree）和“右子树”（right subtree）。二叉树常被用于实现二叉查找树和二叉堆。二叉树是一个连通的无环图，并且每一个顶点的度不大于 3。有根二叉树还要满足根结点的度不大于 2。有了根结点之后，每个顶点定义了唯一的父结点，和最多 2 个子结点。[4]在本例中定义的二叉树如下（省略部分节点信息）：

```
typedef struct fileInfo {
    int data;                           //层数
    struct fileInfo *Child;             //子文件
    struct fileInfo *Brother;           //同级文件
    struct fileInfo *Parent;            //父文件
} Node;

```

其中，Child 表示当前目录（节点）下的子文件夹，若该节点代表的不是文件夹，则其 Child 子节点为空。Brother 表示该节点（文件）的下一个同级（节点），即一个节点与其兄弟节点在同一级目录下。Parent 表示当前节点（文件）的父节点，即一个节点的父节点即为该节点的上一级目录。如图 4-1 所示，

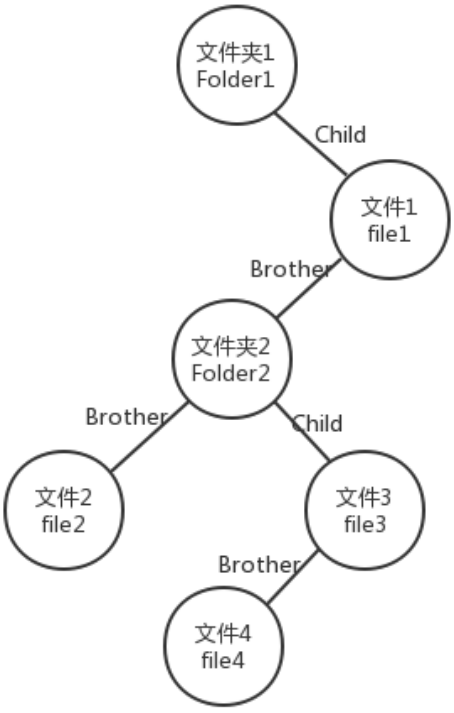

图 4-1

该图表示的文件结构为，文件夹 1 中包含文件 1、文件夹 2、文件 2；文件夹 2 中包含了文件 3、文件 4,即“Forder1[file1, Forder2[file3, file4], file2]”。

4.1.1 删除节点（文件管理系统中的删除文件操作）

二叉树的节点删除操作运用在本文件系统中，具体实现了文件以及文件夹的删除操作，结构体中节点的 data 变量表示文件所在目录的层数，通过待删除节

点的上下节点的层数来判断进行“Child-Brother”或“Brother-Brother”之间的连接。如在图 4-1 中，若要删除 file1 文件，则需将上级文件夹 Forder1 的 Child 节点与文件 file1 的 Brother 节点相连，进行的是“Child-Brother”之间的连接。

删除节点（文件）的核心代码如下：

```
1.
2.  void deleteNode(Node* del_node){
3.      if(del_node -> Brother != NULL){
4.          if(del_node -> Parent -> data == del_node -> data - 1){
5.              del_node -> Parent -> Child = del_node -> Brother;
6.              del_node -> Brother -> Parent = del_node -> Parent;
7.          } else {
8.              del_node -> Parent -> Brother = del_node -> Brother;
9.              del_node -> Brother -> Parent = del_node -> Parent;
10.         }
11.     } else {
12.         if(del_node -> Parent -> data == del_node -> data - 1){
13.             del_node -> Parent -> Child = NULL;
14.         } else {
15.             del_node -> Parent -> Brother = NULL;
16.         }
17.     }
18.     //释放变量
19.    free(del_node);
20. }
21.
```

4.1.2 增加节点（文件管理系统中的新建文件操作）

二叉树的节点添加操作运用在本文件系统中，具体实现了文件以及文件夹的新建操作，通过待新增节点的上下节点的层数来判断进行“Child-new”或“Brother-new”之间的连接。如在图 4-1 中，若要在文件夹 Folder2 目录下新建文件，则首先需要遍历到 Folder2 节点（文件目录）下 Brother 值为 NULL 的节点（图中为 file4），将新增节点与该节点进行连接，即为“Brother-new”之间的连接；若 Folder2 节点下的 Child 值为 NULL，则进行的是“Child-new” 之间的连接。

增加节点（新建文件）的核心代码如下：

```
1.  void add(Node* add_node){
2.      if(i_node -> Child == NULL){
3.          i_node -> Child = add_node;
4.          add_node -> Parent = i_node;
```

```
    } else {
        Node* ls_node=(Node*)malloc(sizeof(Node));
        ls_node = i_node -> Child;
        while(ls_node -> Brother != NULL){
            ls_node = ls_node -> Brother;
        }
        ls_node -> Brother = add_node;
        add_node -> Parent = ls_node;
    }
}
```

4.1.3 标记节点（标记文件当前目录）

除了根节点之外，在文件管理系统中设置了标记节点来标记文件的当前目录，具体定义如下：

```
Node* i_node=(Node*)malloc(sizeof(Node));        //标记当前位置的节点
```

在文件管理系统中的主要功能中涉及到的该节点操作有展示当前目录文件，返回上级目录，展示当前目录，文件查找（这一功能通过文件删除、文件重命名、输出文件信息、写入文件、文件上锁等功能总结得出）。

（1）展示当前目录

在该功能中标记节点的主要作用为，可以直接从标记节点直接从节点位置向上遍历，依次存入数组中，遍历结束后进行倒序输出，遍历节点的形式为“Parent->Parent”。如图 4-2 所示，若当前所在目录为 Folder2 文件夹（即标记节点为位置指向 Folder2），则遍历后得到的数组应为[Folder2, Folder1]，倒序输出结果即为“folder1 > Folder2”。

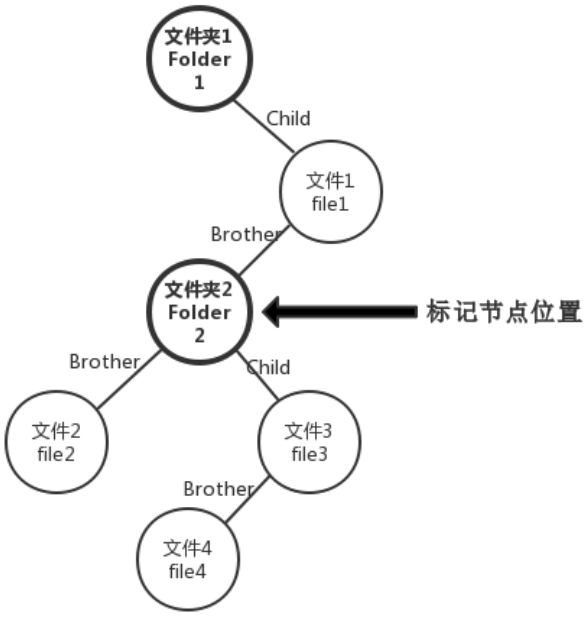

图 4-2

展示当前目录文件的核心代码如下：

```
1.  //展示当前目录
2.  void printcurDir(){
3.      cout << "#" << username;
4.      char dir[10][11];
5.      Node* ls_node=(Node*)malloc(sizeof(Node));
6.      ls_node = i_node;
7.      int i = 1;
8.      if(i_node -> data != 0){
9.          strcpy(dir[0], ls_node -> filename);
10.         while(ls_node -> data != 0){
11.             while(ls_node -> Parent -> data == ls_node -> data){
12.                 ls_node = ls_node -> Parent;
13.             }
14.             ls_node = ls_node -> Parent;
15.             strcpy(dir[i], ls_node -> filename);
16.             i += 1;
17.         }
18.         for(i-=1;i>=0; i--){
19.             cout << ">" << dir[i];
20.         }
21.     } else {
22.         cout << ">localhost";
23.     }
24.     cout << endl;
25. }
```

（2）返回上级目录

与“展示当前目录”相同，标记节点的主要作用在于只需简单的几步判断，就能够对标记节点进行重定向。虽然增加了占用内存的大小，但操作更加灵活，出错率更加小。遍历方法与展示目录相同，但不同在于只要遍历到与当前目录的 data 值不同的节点（标记节点的 data = 上级节点的 data + 1），即停止遍历，进行重定向操作。

返回上级目录的核心代码如下：

```
1.  //返回上级目录
2.  void cdreturn(){
3.      if(i_node -> data == 0){
4.          cout << "no Parent dir!";
5.      } else {
```

```
6.          while(i_node -> Parent -> data == i_node -> data){
7.              i_node = i_node -> Parent;
8.          }
9.          i_node = i_node -> Parent;
10.     }
11. }
```

（3）文件查找

文件查找功能通过文件删除、文件重命名、输出文件信息、写入文件、文件上锁等功能合并总结得出。与前两条功能不同，文件查找主要为向下遍历，即从标记节点出发，进行形式为“Child -> Brother*”的遍历。如图 4-3 所示，若需要在 Folder1 目录下查找到 file2 文件，则从标记节点出发，依次遍历“Folder1 -> file1 -> Folder2 -> file2”

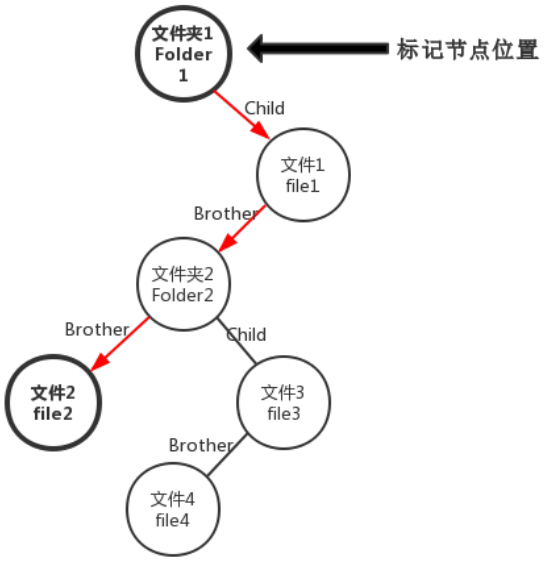

图 4-3

文件查找的核心代码如下

```
1.
2.  //删除文件(夹)
3.  void deleteFile(){
4.      int flag = 1;
5.      if(i_node -> Child != NULL){
6.          cout << "please input the folder you want to delete:";
7.          char ch[11];
8.          cin >> ch;
9.          Node* ls_node=(Node*)malloc(sizeof(Node));
10.         ls_node = i_node -> Child;
11.         if(strcmp(ls_node -> filename, ch) == 0){
12.             flag = 0;
13.         } else {
```

```
            while(ls_node -> Brother != NULL){
                ls_node = ls_node -> Brother;
                if(strcmp(ls_node -> filename, ch) == 0){
                    flag = 0;
                    break;
                }
            }
        }
        if(flag){
            cout << "no folder named [" << ch << "] exit!";
        } else if(ls_node -> lock == 1 && strcmp(ls_node -> createAuthor, us
ername) != 0){
            cout << "You have no power to delete!";
        } else {
            deleteNode(ls_node);
            cout << "successful!";
        }
    } else {
        cout << "No file exit in Directory!";
    }
}
```

4.2 文件上锁

在文件管理系统中，在用户创建文件（夹）后，可以对文件进行上锁，上锁之后，其他用户无法再对该文件进行读写删除等操作，实现该操作主要通过对结构体中的 lock 变量进行该值操作，以及在新建文件（夹）时对结构题中的 createAuthor 进行赋值，查看、修改该用户的文件或进入用户创建的文件夹需要先对 lock 值进行判断，若文件未上上锁（lock 为 0），可继续操作；若文件上锁，则对用户名（username）和文件创建者（createAuthor）进行比较，若一致，则可继续操作，若不一致，则无法继续进行操作。

文件上锁的核心代码如下：

```
//文件上锁
void lockfile(){
    int flag = 1;
    if(i_node -> Child != NULL){
        cout << "please input the filename:";
        char ch[11];
        cin >> ch;
        Node* ls_node=(Node*)malloc(sizeof(Node));
        ls_node = i_node -> Child;
```

```
        if(strcmp(ls_node -> filename, ch) == 0){
            flag = 0;
        } else {
            while(ls_node -> Brother != NULL){
                ls_node = ls_node -> Brother;
                if(strcmp(ls_node -> filename, ch) == 0){
                    flag = 0;
                    break;
                }
            }
        }
        if(flag){
            cout << "no folder named [" << ch << "] exit!";
        } else {
            ls_node -> lock = 1;
        }
    } else {
        cout << "No file exit, please input '2' to create new file.";
    }
}
```

5 运行结果

5.1 运行环境

处理器：Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 GHz
操作系统:Windows10 专业版
文件管理系统编译器：Sublime Text3

5.2 文件管理系统运行结果

用 username 为 jason 的用户创建如下文件目录：

名称	类型	上级目录
Locahost	根目录	/
└— jfolder	文件夹	Locahost
└—jfoder2	文件及	jfolder
└—J2.txt	txt 文件	jfolder

└— j.txt txt 文件（上锁） Locahost

5.2.1 用户登录

（1）输入用户名称：jack

```
Please enter your name:jack
```

（2）进入命令提示界面

```
************Conmand List**************
*                                      *
*  0:back to previous list             *
*  1:create a new folder               *
*  2:create a new file                 *
*  3:turn to next list                 *
*  4:delete file/folder                *
*  5:rename                            *
*  6:show file infomation              *
*  7:write                             *
*  8:lock file                         *
*  9:log out                           *
*                                      *
****************************************

***********Current Directory************
*  #jack>localhost
****************************************

*************File Content***************
*
*  jfolder      j.txt
*
****************************************

jack, Please input the conmand:
```

5.2.2 删除文件/文件夹

（1）输入命令“4”，再输入“j.txt”,将文件删除

```
jack, Please input the conmand:4
please input the folder you want to delete:j.txt
successful!
Enter any key to continue...
```

（2）看到文件目录中“j.txt”已不存在

```
***********Current Directory************
*  #jack>localhost
****************************************

*************File Content***************
*
*  jfolder
*
****************************************

jack, Please input the conmand:
```

5.2.3 新建文件（夹）

（1）输入命令“2”，再输入“jack.txt”，新建文件

```
jack, Please input the conmand:2
Input filename:jack.txt
File [jack.txt] successfully added!
Create time:  Wed Dec 18 12:50:25 2019

Enter any key to continue...
```

（2）看到文件目录中“jack.txt”已新建成功

```
***********Current Directory***********
*  #jack>localhost
**************************************

*************File Content***************
*
*  jfolder      jack.txt
*
**************************************

jack, Please input the conmand:
```

5.2.4 写入文件

（1）输入命令“7”，再输入“jack.txt”，对自己创建的文件进行写操作

```
jack, Please input the conmand:7
please input the filename:jack.txt
```

（2）输入写入的内容

```
please input the file content:helloimjack
```

5.2.5 查看文件具体信息

（1）输入命令“6”，再输入“jack.txt”，查看文件的详细内容

```
jack, Please input the conmand:6
please input the filename:jack.txt
filename:               jack.txt
createtime:             Wed Dec 18 12:50:25 helloimjack
Author:                 jack
lock:                   0
content:                helloimjack

Enter any key to continue...
```

5.2.6 转到下一级目录

（1）输入命令“3”，再输入“jfolder”，进入“jfolder”文件夹

```
jack, Please input the conmand:3
please input the folder you want in:jfolder
```

（2）进入后的路径以及文件目录

```
**********Current Directory**********
*  #jack>localhost>jfolder
****************************************

*************File Content**************
*
*  jfoder2       j2.txt
*
****************************************

jack, Please input the conmand:
```

5.2.7 修改文件（夹）名称

（1）输入命令“5”，再输入要修改的文件夹名称“jfoder”，在输入新名称“jackf”

```
jack, Please input the conmand:5
please input the filename you want to rename:jfoder2
please input the new name:jackf
```

（2）修改完后文件目录也进行更新

```
**********Current Directory**********
*  #jack>localhost>jfolder
****************************************

*************File Content**************
*
*  jackf         j2.txt
*
****************************************

jack, Please input the conmand:
```

5.2.8 文件上锁

用户 jason 已对文件 localhost > jfolder 下的文件“j2.txt”进行上锁，使用用户 jack 对该文件进行信息读取操作。

（1）输入命令“6”，再输入文件名称“j2.txt”，显示“没有权利读”

```
jack, Please input the conmand:6
please input the filename:j2.txt
You have no power to read!
Enter any key to continue...
```

（2）输入命令“9”，退出登录，使用用户 jason 登录读取文件信息，成功读取

```
jason, Please input the conmand:6
please input the filename:j2.txt
filename:                j2.txt
createtime:              Wed Dec 18 12:30:30 2019

Author:                  jason
lock:                    1
content:                 2019


Enter any key to continue...
```

5.2.9 返回上一目录

当前目录

```
***********Current Directory***********
*  #jack>localhost>jfolder
*****************************************
```

（1）输入命令“0”，返回到上一目录

```
jack, Please input the conmand:0
```

（2）执行后目录

```
***********Current Directory***********
*  #jack>localhost
*****************************************
```

6 调试和改进

（1）调试过程中，误将节点的父节点理解成为上一级目录的节点，导致文件目录输出，查找文件等功能出错。对算法进行修改后遍历找到节点的 data 数值等于标记节点 data 减一的节点停止遍历（具体改进后内容见 4.2.3 的“返回上级目录”）。

（2）初步完成时，没有将文件夹与数据文件进行区分，导致可以在数据文件节点下新建文件。后期对每个节点添加了 attribute 变量进行区分，每次进行进入文件夹操作时，即可通过判断节点的 attribute 值进行异常处理。

（3）在输出上，对文件列表输出进行了优化，文件夹用绿色字体输出，文件则用白色输出，报错时，用红色字体输出，这样优化了用户体验。

7 心得和结论

7.1 结论和体会

通过对文件管理系统相关资料的查阅，最后确定使用二叉树的结构对文件系统进行存储管理。期间不仅对之前所学的二叉树相关的知识有了巩固，文件系统的管理有了更高的理解。部分结论和体会在撰写关键技术的时候就有提到。在文件安全这一部分，我最后添加了文件上锁功能，实现的比较简单，但我知道一个成熟的文件管理系统在安全方面一定是设置了重重保障的。

7.2 进一步改进方向

（1）在总体上将文件系统搬运至 web 网页端，或者通过其他可视化界面达成文件管理的效果。
（2）在数据文件的保存写入方面，能够换行写入。读取文件信息时也可以展示换行、缩进后的效果，并且实现动态改变存储区域的功能。
（3）通过数据库读取的方式验证用户名、密码进行登录操作。
（4）本文件管理系统中设置了文件上锁的功能，将来可以拓展到设置文件只读等功能。
（5）本文件管理系统中，文件信息都是以模拟的形式存储，即关闭进程后没有实际存在的文件保存下来，将来可以连接云端数据对各个文件进行保存操作。
（6）设置私人文件夹以及共享文件功能，可以将自己的文件分享给其他单个或多个用户。
（7）在管理方面可以给单个或多个文件（夹）设置管理员，即可以设置哪些人可以修改文件，哪些人只能只读文件。
（8）在删除文件的功能中，若对没有上锁的文件进行删除，同时恰好该文件夹下有被上锁的文件，则删除依然成功，后期可以优化这一点。

7.3 分析设计方案对系统安全的影响

增加了文件上锁功能，可以对文件进行上锁，这一点加强了文件的安全系数。