河北经贸大学

信息技术学院　网络工程专业

《操作系统》课程设计

题目

班级：网络工程一班＿＿

学号： 20212245064＿＿

姓名：＿田少奇＿＿＿＿

目录

[1. 绪论 3](#_Toc636)

[2. 需求分析 3](#_Toc2340)

[2.1 用户管理 3](#_Toc5984)

[2.2 文件管理 3](#_Toc6656)

[2.3 目录管理 3](#_Toc3670)

[2.4 读写权限控制 4](#_Toc28705)

[2.5 错误提示功能 4](#_Toc13319)

[3. 设计思路 4](#_Toc31602)

[3.1 用户管理 4](#_Toc16142)

[3.2 文件管理 4](#_Toc27919)

[3.3 目录管理 5](#_Toc25921)

[3.4 读写权限控制 5](#_Toc28666)

[3.5 错误提示功能 6](#_Toc18451)

4. 程序流程图............................................................................................................................................7

[5.1 用户结构体（USER） 8](#_Toc8300)

[5.2 目录及文件结构体（DF） 8](#_Toc22820)

[5.3 第一初始化界面->用户登录|注册函数（init\_0()） 8](#_Toc32426)

[5.4 第一初始界面->选项（solving\_0()） 8](#_Toc8252)

[5.5 用户登录（user\_login()） 9](#_Toc16777)

[5.6 用户注册（user\_enroll()） 9](#_Toc22089)

[5.7 系统退出（end()） 9](#_Toc19972)

[5.8 第二初始界面->文件管理系统的操作（init\_1()） 9](#_Toc9492)

[5.9 第二初始界面->选项（solving\_1()） 10](#_Toc1168)

[5.10 查看当前文件目录下的内容（get\_DF(DF\* temp\_df)） 10](#_Toc6597)

[5.11 返回父目录（ret\_DF(DF\* temp\_df)） 10](#_Toc23066)

[5.12 进入子目录（gin\_DF(DF\* temp\_df)） 11](#_Toc2165)

[5.13 在当前目录下进行操作（mak\_DF()） 11](#_Toc8646)

[5.14 第三初始界面->当前路劲下的操作（init\_2()） 11](#_Toc20826)

[5.15 第三初始界面->选项（solving\_2()） 12](#_Toc22003)

[5.16 增加目录/文件（add\_df()） 12](#_Toc15448)

[5.17 删除目录/文件（del\_df()） 13](#_Toc25102)

[5.20 返回上一页（ref\_df()） 14](#_Toc1519)

[5.21 主函数（main()） 14](#_Toc15409)

[5.22 初始化 14](#_Toc20715)

[6. 测试 16](#_Toc2622)

[6.1 测试1 16](#_Toc21233)

[6.2 测试2 18](#_Toc30948)

[6.3 测试3 19](#_Toc4694)

[6.4 测试4 20](#_Toc16845)

[7. 实现过程中遇到的问题 21](#_Toc9971)

[8. 结论 21](#_Toc10261)

[9. 参考文献 22](#_Toc12639)

[10. 附录 22](#_Toc16891)

1. 绪论

本次课程设计目的是基于c++语言简单模拟linux的文件管理系统，旨在提供一个简单的文件操作平台，以便用户能够更方便的管理文件和目录。该系统模拟了用户的登录和注册，以及文件的增删读写和目录的增删访问等功能。

该系统使用了c++语言的标准库，包括iostream、string、cstring等。用户可以通过输入用户名和密码进行登录，并可以创建、删除、编写、读取文件和目录。

该系统的代码实现了模块化设计，使得用户可以方便的修改和扩展功能。同时该系统也进行了充分的测试，以确保了其稳定性和可靠性。

总之，该文件管理系统提供了一个基本的文件操作平台，方便用户管理文件和目录，具有较高的扩展性和实用性。

1. 需求分析
   1. 用户管理

系统需要支持管理员、普通用户等不同用户类型的管理。管理员应该具有更高的权限，可以创建、读取、修改、删除文件及创建、删除目录，而普通用户只能读取文件及目录。

* 1. 文件管理

系统需要支持文件的创建、修改、删除等操作，其文件也应该具有其对外的权限，例如只有系统管理者才有权限对其进行相关的文件管理操作，普通用户（除指定用户）只对其有访问的权限。

* 1. 目录管理

系统需要支持目录的创建、删除、修改等操作，其目录也应该具有其对外的权限，例如只有系统管理者才有权限对其进行相关的目录管理操作，普通用户（除指定用户）只对其有访问的权限。

。

* 1. 读写权限控制

系统需要支持对文件或目录的读写权限进行控制，可以根据用户类型或权限、文件或目录的对外权限进行设置。

* 1. 错误提示功能

系统需要具有一定的安全性和自我保护性。当系统使用者错误使用文件管理系统的操作时应该对该使用者进行提示，当该使用者连续并多次错误使用时，不同错误的提示次数不同，情节严重者，文件管理系统将自行强制退出系统从而进行自我保护。

1. 设计思路

本次课程设计主要以c++语言为主，其中调用了c的输入语句，字符串的标准库，指针，hash表等相关数据结构。

* 1. 用户管理

首先定义一个结构体来存储用户的信息，其中包括了用户的姓名，账户名，密码，权限等信息。

其次建立一个用户数组进行调用结构体来存储多个用户的信息，在用户注册的时候将对上述的用户相关信息进行结构体数组的信息存入。在用户登录的时候将对该用户结构体数组内的信息进行遍历查找，如果没有相对应的用户信息系统将对该用户进行反馈错误。

* 1. 文件管理

首先定义一个结构体来存储文件的信息，其中包括了文件的名字，内容，权限，父目录，同级上一个目录或文件，同级下一个目录或文件等信息，如果其相关信息为空时同样需要声明。

其次通过指针将文件与父目录，上一个目录或文件，下一个目录或文件进行关联，使文件与文件或目录之间形成一个树状结构，在用户进行查找、创建、删除时只需要改变其相关联的父目录、上一目录或文件、下一目录或文件的相关信息即可进行操作。

* 1. 目录管理

首先定义一个结构体来存储目录的信息，其中包括了目录的名字，权限，下一级的第一个目录或文件，下一级最后一个目录或文件，同级上一个目录或文件、同级下一个目录或文件等信息。

其次通过指针将当前目录与父目录，同级上一个目录或文件，同级下一个目录或文件，下一级第一个目录或文件，下一级最后一个目录或文件进行管理，使目录与目录或文件之间形成一个树状结构，并且通过指针可以在目录与相关联目录或文件之间进行调用转换，在用户进行查找、创建、删除使只需要需要改变其相关联的父目录、同级上一目录或文件、同级下一目录或文件的相关信息即可进行操作。

* 1. 读写权限控制

首先在用户结构体、文件结构体当中需要进行定义其本身的权限，在用户结构体的权限设定中一共有三种权限，分别是最高权限，管理者权限，和用户权限，在文件结构体中的权限设定同样应有三种权限，分别是最高保密权限（仅最高权限者可以创建、删除、读取、编写），次高保密权限（仅最高权限、管理员权限者可以创建、删除、读取、编写），普通权限（普通用户可以读取、编写）。

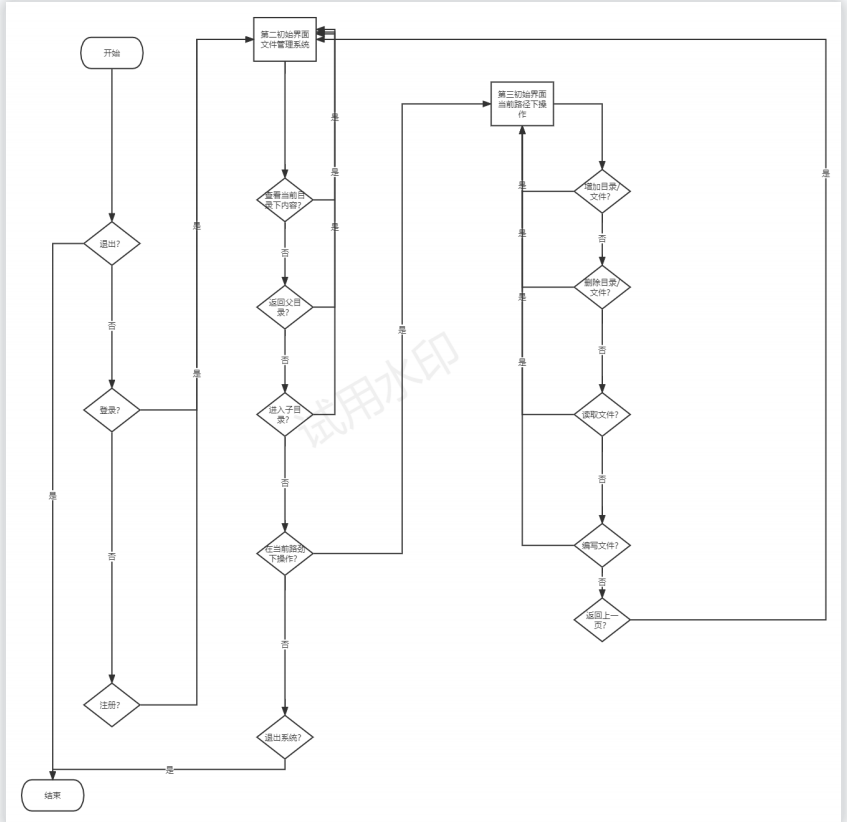
其次在对文件进行请求读写时将在用户结构体数组与文件结构体数组中进行遍历找到当前用户与文件的权限，再进行分析该用户是否对该文件具有读写权限，从而保证系统文件的读写权限控制。

* 1. 错误提示功能

首先将系统内部的各个操作定义成各个函数体，在每个函数体内部定义其相对于的操作连续失败的次数的变量。

其次在用户进行系统内部操作的时候系统将调用该操作的函数体，在用户操作失败时该操作函数体内部进行记录连续失败次数的变量需要更新数值加一赋值，并且对该用户进行相对应操作错误的提示，当该变量的数值达到了一定的数值时系统将采取强行退出的操作，从而保护了系统不被故意错误使用，保证了一定的安全性。

1. 程序流程图



1. 对于设计思路的实现
   1. 用户结构体（USER）

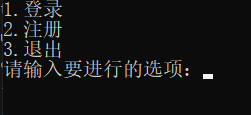
在定义用户结构体（USRR）中定义用户的编号，用户的姓名，用户的账户名，用户的密码，用户的权限（0为最高权限，1为管理员权限，2为普通权限）,并定义用户结构体函数（user[]）来储存多用户的相关信息。

* 1. 目录及文件结构体（DF）

在定义目录及文件结构体时将目录结构体与文件结构体进行合并定义，其中包含了目录及文件的名字，文件的内容（如果当前文本类型是目录时内容为空），当前目录或文件的父目录，当前目录或文件的同级下一个目录或文件，当前目录或文件的同级上一个目录或文件，当前目录的第一个子目录或文件（如果当前文本类型是文件时此内容定义为空），当前目录的最后一个子目录或文件（如果当前文本类型是文件时此内容定义为空），当前文本的类型（0为目录，1为文件），目录及文件的权限（3为最高权限，2为次高权限，1为普通权限）。

* 1. 第一初始化界面->用户登录|注册函数（init\_0()）

在进入系统的第一个页面展示三个选项（1.登录 2.注册 3.退出），如图：



* 1. 第一初始界面->选项（solving\_0()）

在第一初始界面进行选项时，通过switch语句使输入不同的选项将调用不同的函数体:输入1时调用用户登录函数（user\_login()），输入2时调用用户注册函数（user\_enroll()），输入3时调用退出系统函数（end()）.

* 1. 用户登录（user\_login()）

在用户登录函数中首先定义用用户账户输入连续失败次数的变量（st\_account\_0）和用户密码输入连续失败次数的变量（st\_passwd\_0）。

其次定义一个死循环进行用户账户的输入，通过循环遍历用户结构体数组内存储的账户名与当前用户输入的账户名进行判断，如果未查找到将提示用户进行重新输入，如果三次连续错误输入时将调用系统退出函数（end()）进行系统的强制退出，如果在用户结构体数组中查找到相对应的用户名则将进行下一步的密码输入，当密码三次连续错误输入时将调用系统退出函数（end()）进行系统的强制退出，如果密码正确则进行进入系统

* 1. 用户注册（user\_enroll()）

在用户注册时系统将提示用户进行姓名、账户名、密码的输入，用户权限将自行定义为普通用户，并将该新用户的所有信息进行存入到用户结构体数组中。

* 1. 系统退出（end()）

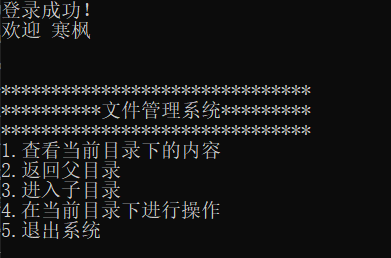
在系统退出时将对用户进行（感谢使用，再见！）的提示，然后exit(0)进行强制退出程序。

* 1. 第二初始界面->文件管理系统的操作（init\_1()）

在用户成功登录进入到系统内部时将展示五个操作：

1. 查看当前目录下的内容
2. 返回父目录
3. 进入子目录
4. 在当前目录下进行操作
5. 退出系统

如图：



* 1. 第二初始界面->选项（solving\_1()）

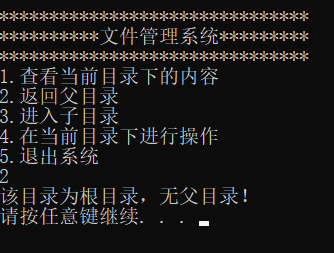
在第二初始界面进行选项时，通过switch语句输入不同选项将调用不同的函数体：输入1时调用查看当前目录下的内容函数（get\_DF(DF\* temp\_df)），输入2时调用返回父目录函数（ret\_DF(DF\* temp\_df)），输入3时调用进入子目录函数（gin\_DF(DF\* temp\_df)），输入4时调用在当前目录下进行操作函数（mak\_DF()），输入5时调用系统退出函数（end()）

* 1. 查看当前文件目录下的内容（get\_DF(DF\* temp\_df)）

在用户查看当前文件目录下的内容时，首先定义一个死循环进行将当前目录或文件进行转化到当前目录或文件的父目录的第一个子目录或文件，然后在从当前目录的父目录的第一个子文件遍历到当前父目录的最后一个子文件并输出，当全部输出完时将返回到第二初始界面及第二初始界面的选项处。

* 1. 返回父目录（ret\_DF(DF\* temp\_df)）

在用户选择返回父目录时只需要将指针指向当前目录或文件的父目录，如果当前目录是根目录时则进行提示（该目录为根目录，无父目录），然后返回第二初始界面及第二初始界面的选项出。如图：



* 1. 进入子目录（gin\_DF(DF\* temp\_df)）

在用户选择进入子目录时系统首先提示用户输入选择进入的子目录的名字，然后通过在目录或文件结构体中遍历查找该名字是否存在。

如果改名字对应的文本不存在时系统将提示（该目录不存在）；

如果该名字存在时需要进行下一步的判该名字对应文本的文本类型是否是目录：

如果不是目录的话系统将提示用户（该文本不是目录，无法进入）；

如果该名字对应文本的文本类型是目录的话系统将直接进入到下一目录并刷新第二初始界面。

* 1. 在当前目录下进行操作（mak\_DF()）

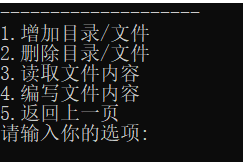
在用户选择在当前目录下进行操作时，系统将调用第三初始界面->当前路劲下的操作函数（init\_2()）和第三初始界面->选项函数（solving\_2()）并进入第三初始界面及选项处。

* 1. 第三初始界面->当前路劲下的操作（init\_2()）

在用户进入第三初始界面时，系统将显示以下五个操作选项：

1. 增加目录/文件
2. 删除目录/文件
3. 读取文件内容
4. 编写文件内容
5. 返回上一页

如图：



* 1. 第三初始界面->选项（solving\_2()）

在第三初始界面进行选项时，通过switch语句输入不同选项将调用不同的函数体：输入1时系统调用增加目录/文件函数（add\_df()），输入2时系统调用删除目录/文件函数（del\_df()），输入3时系统调用读取文件内容函数（red\_df(DF\* temp\_df)），输入4时系统调用编写文件内容函数（wre\_df(DF\* temp\_df)），输入5时系统调用返回上一页函数（ref\_df()）。

* 1. 增加目录/文件（add\_df()）

在用户选择增加文件或目录的时候首先需要判断用户是否具有增加文件或目录的权限：

如果没有，则系统提示用户（对不起！您没有权限增加目录/文件！）；

如果有，那么系统需要进一步判断当前目录是否为根目录，在本次课程设计中默认根目录无法增加并列目录或文件：

如果当前目录是根目录，那么系统提示（对不起！当前目录为根目录，无法增加并列目录或文件）

如果当前目录不是根目录，那么系统将提示用户输入增加文本的文本类型（1为文件，2为目录），紧接着提示用户所要增加的文本的名字，之后将新加的文本的相关信息存入到目录或文件结构体中并更新与该文本相关联的目录或文件及父目录的关联信息。

* 1. 删除目录/文件（del\_df()）

在用户选择删除文件或目录的时候首先需要判断用户是否具有删除文件或目录的权限：

如果没有，则系统提示用户（对不起！您没有权限删除目录/文件！）；

如果有，那么系统需要进一步判断当前目录是否为根目录，在本次课程设计中默认根目录无法增加并列目录或文件：

如果当前目录是根目录，那么系统提示（对不起！当前目录为根目录，无法删除该目录）

如果当前目录不是根目录，那么系统将提示用户输入选择删除的目录或文件名，并在目录或文件结构体中遍历查找是否存在该目录或文件名，如果不存在则提示（文本不存在！）并返回到第三初始界面；如果存在，则将与该文本相关联的目录或文件及父目录的信息更新，使该文本脱离树状结构并系统提示（删除成功！）

* 1. 读取文件内容（red\_df(DF\* temp\_df)）

在用户选择读取文件内容的时候首先需要判断用户是否具有读取文件的权限：

如果没有，系统提示（对不起！您没有权限读取目录/文件！）；

如果有，系统将进行提示用户所要读取的文本名，并在目录或文件结构体中进行遍历查找是否存在该文本：

如果不存在，系统提示（对不起！文本不存在！）；

如果存在，则系统需要进行下一步的判断该文本的文本类型是否为文件：

如果该文本的文本类型是目录时，系统提示（对不起！该文本类型为目录，无法读取！）；

如果该文本的文本类型是文件时，系统将按照字节流的方式输出该文本内容。

* 1. 编写文件内容（wre\_df(DF\* temp\_df)）

在用户选择编写文件内容的时候首先需要判断用户是否具有编写文件的权限：

如果没有，系统提示（对不起！您没有权限编写目录/文件！）；

如果有，系统将进行提示用户所要编写的文本名，并在目录或文件结构体中进行遍历查找是否存在该文本：

如果不存在，系统提示（对不起！文本不存在！）；

如果存在，则系统需要进行下一步的判断该文本的文本类型是否为文件：

如果该文本的文本类型是目录时，系统提示（对不起！该文本类型为目录，无法编写！）；

如果该文本的文本类型是文件时，系统将提示用户输入编写内容按照字节流的方式将内容存入该文本。

* 1. 返回上一页（ref\_df()）

在用户选择返回上一页时系统将调用第二初始界面（init\_1()）与第二初始界面的选项（solving\_1()）并进入到第二初始界面。

* 1. 主函数（main()）

该程序的主函数依次调用了初始化最高特权用户（init\_user()），第一初始界面->用户登录|注册函数（init\_0()），第一初始界面->选项函数（solving\_0()），初始化目录函数（init\_directory()），第二初始界面函数->文件管理系统的操作（init\_1()），第二初始界面->选项函数（solving\_1()）。

* 1. 初始化

由于该文件管理系统是简单模拟linux的文件管理系统，其在初始化时具备了最高权限用户（root），根目录（bin），及一级子目录（home、root、etc）和二级子目录（a、b）等目录

* + 1. 初始化最高权限用户（init\_user()）

在该系统的初始化最高权限用户时，设定的最高权限用户的信息为：

用户编号：1

用户姓名：寒枫

用户账户名：root

用户密码：123456

用户权限：0（最高）

* + 1. 初始化目录（init\_directory()）

在该系统进行初始化目录时调用了初始化根目录函数（init\_DF(DF\* temp\_df)），

初始化自定义的目录函数（init\_df\_0(DF\* temp\_df)）

* + - 1. 初始化根目录（init\_DF(DF\* temp\_df)）

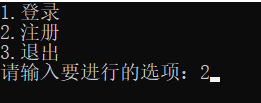
在进行初始化根目录的时候，根目录的权限设定为最高保密权限（3），文本类型为目录（0），父目录为空，第一子目录为空，最后一子目录为空，同级上一目录为空，同级下一目录为空，根目录的名字为bin。

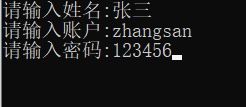
* + - 1. 初始化自定义的目录（init\_df\_0(DF\* temp\_df)）

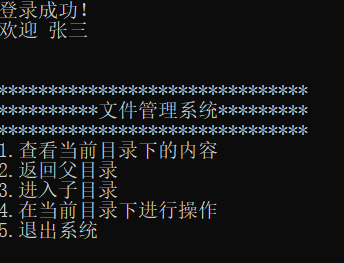
在初始化系统自定义的目录时，一共定义了7个目录或文件：

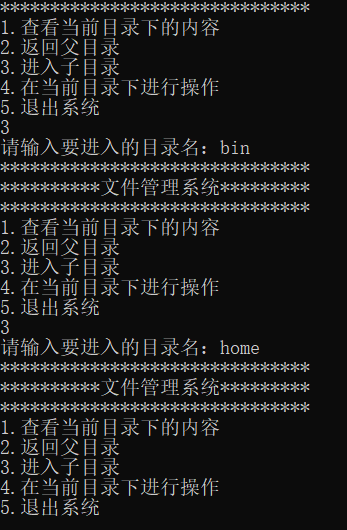
1. root：根目录（bin）下的第一个子目录，同级上一个目录或文件为空，同级下一个目录为home，文本类型为目录（0），第一个子目录为空，最后一个子目录为空，权限为最高保密权限（3）。
2. home：根目录（bin）下的一个子目录，同级上一个目录或文件为root，同级下一个目录为etc，文本类型为目录（0），第一个子目录为a，最后一个子目录为b，权限为最高保密权限（0）。
3. etc：根目录（bin）下的最后一个子目录，同级上一个目录或文件为home，同级下一个目录为空，文本类型为目录（0），第一个子目录为空，最后一个子目录为空，权限为最高保密权限（0）。
4. a：根目录（home）下的第一个子文件，同级上一个目录或文件为空，同级下一个目录为b，文本类型为文件（1），第一个子目录为空，最后一个子目录为空，权限为最高保密权限（0），文件内容为（hanfengzhenshuai）。
5. b：根目录（home）下的最后一个子目录，同级上一个目录或文件为a，同级下一个目录为b，文本类型为目录（0），第一个子文件为tsq，最后一个子文件为hfeng，权限为最高保密权限（0）
6. tsq：根目录（b）下的第一个子文件，同级上一个目录或文件为空，同级下一个目录为hfeng，文本类型为文件（1），第一个子目录为空，最后一个子目录为空，权限为最高保密权限（0），文件内容为（666）。
7. hfeng：根目录（b）下的第一个子文件，同级上一个目录或文件为tsq，同级下一个目录为空，文本类型为文件（1），第一个子目录为空，最后一个子目录为空，权限为最高保密权限（0 ），文件内容为（plus）。
8. 测试
   1. 测试1

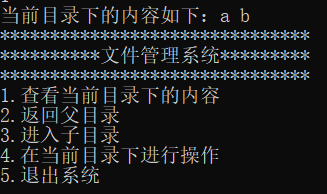
测试要求：注册用户姓名：张三、账户名：zhangsan、密码：123456的用户访问home路径并查看home路径下的内容。





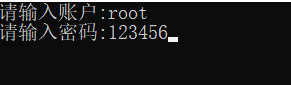


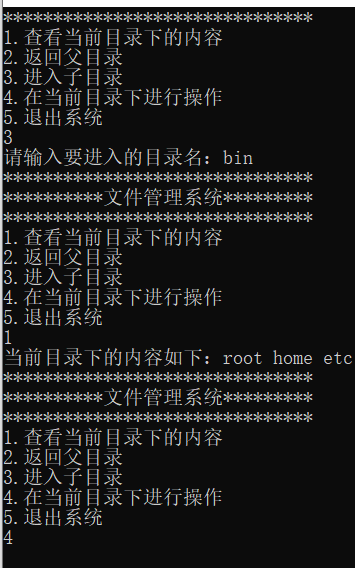


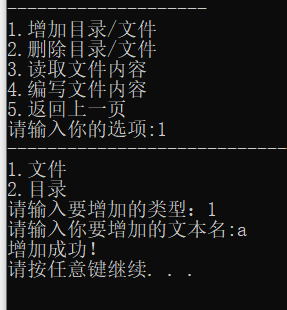


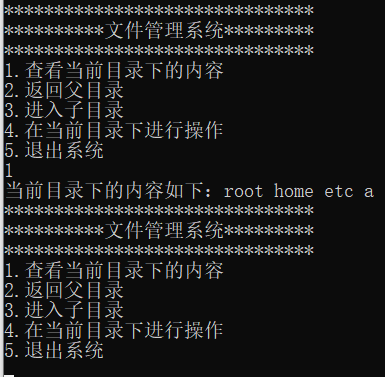
* 1. 测试2

测试要求：以最高权限用户(root)在bin目录下建立文本类型为文件、文本名为a的文件。





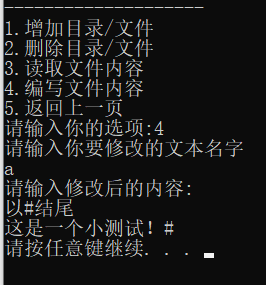


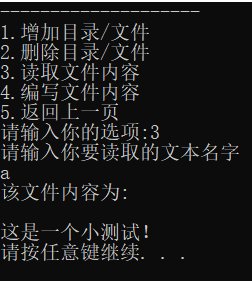


* 1. 测试3

测试要求：以最高权限用户(root)在bin目录下编写文本类型为文件、文本名为a的文件的如下内容：

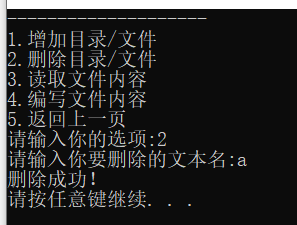
这是一个小测试！

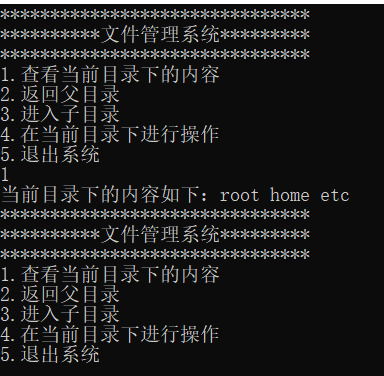




* 1. 测试4

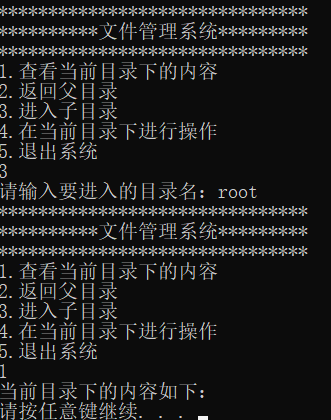
测试要求：以最高权限用户(root)在bin目录下删除文本类型为文件、文本名为a的文件。





1. 实现过程中遇到的问题

在进入空目录之后不管选择什么选项都会强制进行退出。如图：



1. 结论

通过简单模拟 Linux 的文件管理系统，可以更深入地了解操作系统的工作原理和基本功能。在学习操作系统的过程中，我发现操作系统是一个非常复杂和庞大的系统，它涉及到多个方面的知识和技能。但是，通过对操作系统的学习，我们可以更好地理解计算机系统的工作原理，为后续的编程和软件开发打下坚实的基础。

1. 参考文献

计算机操作系统第四版

1. 附录

#include<iostream>

#include<algorithm>

#include<string>

#include<cstring>

#include<math.h>

#include<cstdio>

#include<iomanip>

#include<polarity.h>

#include<vector>

#include<fstream>

#include<windows.h>

#include<stdio.h>

using namespace std;

typedef long long ll;

//用户结构体

struct USER

{

    ll user\_id;//用户的编号

    string user\_name;//用户的姓名

    string user\_account;//用户的账户

    string user\_passwd;//用户的密码

    ll user\_Permissions;//用户的权限 0为最高，1为读写，2为只读

}user[1010];

//目录及文件结构体

struct DF//Directories and files -> 目录及文件

{

    char df\_name[1010];//目录及文件的名字

    char df\_main[1010];//文件的内容

    struct DF\* df\_father;//父目录

    struct DF\* df\_next;//同级下一个目录

    struct DF\* df\_front;//同级上一个目录

    struct DF\* df\_son\_0;//第一个子目录或子文件

    struct DF\* df\_son\_1;//最后一个子目录或子文件

    ll df\_type;//df的类型:目录为 0 | 文本为 1

    ll df\_Permissions;//目录及文件的权限

};

void solving\_0();

void init\_0();

void init\_directory();

void init\_user();

int end();

void user\_login();

void user\_enroll();

void solving\_1();

void gin\_DF(DF\* temp\_df);

void ret\_DF(DF\* temp\_df);

void get\_DF(DF\* temp\_df);

void init\_directory();

void init\_df\_0(DF\* df\_now);

void init\_DF(DF\* temp\_df);

void init\_1();

void mak\_DF();

void init\_2();

void solving\_2();

ll user\_num;//设定用户总数

ll choice\_0;//第一初始界面的选择

ll temp\_id;//临时编号

string temp\_name;//临时姓名

string temp\_account;//临时账户

string temp\_passwd;//临时密码

ll temp\_Permissions;//临时权限

ll choice\_1;//第二初始界面的选择

DF\* df,\* df\_now;//提前定义结构体df全局变量

ll choice\_2;//第三初始界面的选择

ll choice\_3;//增加文本类型的选择

bool st\_choice\_3=true;//表示文本类型选择的正确性

bool st\_del=false;//表示删除文本是否查询到

bool st\_gin\_3=true;//表示删除文本是否查询到

//由于所有的登录失败的可能全部去掉，那么剩下的就只剩下成功的可能，所以不需要记录是否成功

//bool st\_login=false;//定义记录是否登录成功的变量

//初始化最高特权用户

void init\_user()

{

    user\_num=1;

    user[1].user\_id=1;

    user[1].user\_name="寒枫";

    user[1].user\_account="root";

    user[1].user\_passwd="123456";

    user[1].user\_Permissions=0;

}

//第一初始界面->用户登录|注册1

void init\_0()

{

    cout<<"1.登录"<<endl;

    cout<<"2.注册"<<endl;

    cout<<"3.退出"<<endl;

    cout<<"请输入要进行的选项：";

}

//第一初始界面->选项2

void solving\_0()

{

    cin>>choice\_0;

    system("cls");

    switch(choice\_0)

    {

        case 1://用户登录

            user\_login();

            break;

        case 2://用户注册

            user\_enroll();

            break;

        case 3://退出系统

            end();

    }

}

//系统退出

int end()

{

    cout<<"感谢使用，再见！"<<endl;

    exit(0);

}

//用户登录

void user\_login()

{

    bool st\_account\_0=true;//定义死循环，直到用户登录成功|三次登录失败，否则重新输入账户

    bool st\_passwd\_0=true;//定义死循环，直到用户登录成功|三次登录失败,否则重新输入密码

    ll login\_account\_num=0;//记录账户错误的次数

    ll login\_num=0;//记录密码错误的次数

    while(st\_account\_0)

    {

        cout<<"请输入账户:";

        cin>>temp\_account;

        for(ll i=1;i<=user\_num;i++)

        {

            if(user[i].user\_account==temp\_account)

            {

                st\_account\_0=false;//更新已经查询到账户

                while(st\_passwd\_0)

                {

                    cout<<"请输入密码:";

                    cin>>temp\_passwd;

                    if(user[i].user\_passwd==temp\_passwd)

                    {

                        st\_passwd\_0=false;

                        temp\_id=i;

                        temp\_name=user[i].user\_name;

                        temp\_Permissions=user[i].user\_Permissions;//更新临时权限为当前用户的权限

                        system("cls");

                        //st\_login=true;//记录成功登录

                        cout<<"登录成功！\n欢迎 "<<temp\_name<<endl;

                        //st\_passwd\_0=false;//更新账户密码正确

                        break;

                    }

                    else

                    {

                        login\_num++;

                        if(login\_num==3)

                        {

                            cout<<"您已连续三次登录失败，系统将关闭"<<"\n"<<"如需使用，请重新进入！"<<endl;

                            system("pause");

                            end();

                        }

                        else

                        {

                            cout<<"密码错误！请重新输入！"<<endl;

                        }

                    }

                }

            }

        }

        login\_account\_num++;//账户不存在，更新登录的失败次数

        if(login\_account\_num==3)

        {

            cout<<"您已连续三次登录失败，系统将关闭"<<"\n"<<"如需使用，请重新进入！"<<endl;

            system("pause");

            end();

        }

        else if(st\_passwd\_0)//如果密码仍为未查询到的状态，代表账户不存在

        {

            cout<<"账户不存在！请重新输入！"<<endl;

            system("pause");

            system("cls");

        }

    }

}

//用户注册

void user\_enroll()

{

    user\_num++;//更新用户总数

    cout<<"请输入姓名:";

    cin>>user[user\_num].user\_name;

    cout<<"请输入账户:";

    cin>>user[user\_num].user\_account;

    cout<<"请输入密码:";

    cin>>user[user\_num].user\_passwd;

    user[user\_num].user\_id=user\_num;//初始化新用户的编号

    user[user\_num].user\_Permissions=2;//初始化新用户的特权为2

    //更新临时用户的所有资料

    temp\_account=user[user\_num].user\_account;

    temp\_id=user[user\_num].user\_id;

    temp\_name=user[user\_num].user\_name;

    temp\_passwd=user[user\_num].user\_passwd;

    temp\_Permissions=user[user\_num].user\_Permissions;

    system("cls");

    cout<<"登录成功！\n欢迎 "<<temp\_name<<endl;

    //st\_login=true;//更新登录成功

}

/\*

    以上是第一初始界面->用户登录|注册问题的所以相关程序

\*/

//第二初始界面->文件管理系统的操作

void init\_1()

{

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*文件管理系统\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    cout<<"1.查看当前目录下的内容"<<endl;

    cout<<"2.返回父目录"<<endl;

    cout<<"3.进入子目录"<<endl;

    cout<<"4.在当前目录下进行操作"<<endl;

    cout<<"5.退出系统"<<endl;

}

//初始化DF根目录

void init\_DF(DF\* temp\_df)

{

    DF\* t=temp\_df;

    strcpy(t->df\_name, "bin");

    t->df\_father=NULL;

    t->df\_front=NULL;

    t->df\_next=NULL;

    t->df\_Permissions=3;

    t->df\_type=0;

    t->df\_son\_0=NULL;

    t->df\_son\_1=NULL;

}

//初始化自定义的目录

void init\_df\_0(DF\* df\_now)

{

    DF\* df\_new\_1,\* df\_new\_2,\* df\_new\_3,\* df\_new\_4,\* df\_new\_5,\* df\_new\_6,\* df\_new\_7;

    df\_new\_1=(DF\*)malloc(sizeof(DF));

    df\_new\_2=(DF\*)malloc(sizeof(DF));

    df\_new\_3=(DF\*)malloc(sizeof(DF));

    df\_new\_4=(DF\*)malloc(sizeof(DF));

    df\_new\_5=(DF\*)malloc(sizeof(DF));

    df\_new\_6=(DF\*)malloc(sizeof(DF));

    df\_new\_7=(DF\*)malloc(sizeof(DF));

    //df\_now=(DF\*)malloc(sizeof(DF));

    strcpy(df\_new\_1->df\_name,"root");

    strcpy(df\_new\_2->df\_name,"home");

    strcpy(df\_new\_3->df\_name,"etc");

    strcpy(df\_new\_4->df\_name,"a");

    strcpy(df\_new\_5->df\_name,"b");

    strcpy(df\_new\_6->df\_name,"tsq");

    strcpy(df\_new\_7->df\_name,"hfeng");

    df\_now->df\_son\_0=df\_new\_1;

    df\_now->df\_son\_1=df\_new\_3;

    df\_new\_1->df\_father=df\_now;

    df\_new\_1->df\_type=0;

    df\_new\_1->df\_son\_1=NULL;

    df\_new\_1->df\_son\_0=NULL;

    df\_new\_1->df\_Permissions=3;

    df\_new\_1->df\_next=df\_new\_2;

    df\_new\_1->df\_front=NULL;

    df\_new\_2->df\_father=df\_now;

    df\_new\_2->df\_next=df\_new\_3;

    df\_new\_2->df\_front=df\_new\_1;

    df\_new\_2->df\_son\_0=df\_new\_4;

    df\_new\_2->df\_son\_1=df\_new\_5;

    df\_new\_2->df\_type=0;

    df\_new\_2->df\_Permissions=0;;

    df\_new\_3->df\_father=df\_now;

    df\_new\_3->df\_next=NULL;

    df\_new\_3->df\_front=df\_new\_2;

    df\_new\_3->df\_son\_0=NULL;

    df\_new\_3->df\_son\_1=NULL;

    df\_new\_3->df\_Permissions=0;

    df\_new\_3->df\_type=0;

    df\_new\_4->df\_father=df\_new\_2;

    strcpy(df\_new\_4->df\_main,"hanfengzhenshuai");

    df\_new\_4->df\_next=df\_new\_5;

    df\_new\_4->df\_front=NULL;

    df\_new\_4->df\_son\_0=NULL;

    df\_new\_4->df\_son\_1=NULL;

    df\_new\_4->df\_type=1;

    df\_new\_4->df\_Permissions=0;

    df\_new\_5->df\_father=df\_new\_2;

    df\_new\_5->df\_next=NULL;

    df\_new\_5->df\_front=df\_new\_4;

    df\_new\_5->df\_son\_0=df\_new\_6;

    df\_new\_5->df\_son\_1=df\_new\_7;

    df\_new\_5->df\_Permissions=0;

    df\_new\_5->df\_type=0;

    df\_new\_6->df\_father=df\_new\_5;

    df\_new\_6->df\_next=df\_new\_7;

    df\_new\_6->df\_front=NULL;

    df\_new\_6->df\_son\_0=NULL;

    df\_new\_6->df\_son\_1=NULL;

    df\_new\_6->df\_type=1;

    df\_new\_6->df\_Permissions=0;

    strcpy(df\_new\_6->df\_main,"666");

    df\_new\_7->df\_father=df\_new\_5;

    df\_new\_7->df\_front=df\_new\_6;

    df\_new\_7->df\_next=NULL;

    df\_new\_7->df\_Permissions=0;

    df\_new\_7->df\_type=1;

    df\_new\_7->df\_son\_0=NULL;

    df\_new\_7->df\_son\_1=NULL;

    strcpy(df\_new\_7->df\_main,"plus");

    cout<<"\n"<<endl;

    /\*

    cout<<"0000000000000000000000"<<endl;

    cout<<df\_now->df\_name<<endl;

    cout<<df\_now->df\_son\_0->df\_name<<endl;

    cout<<df\_now->df\_son\_1->df\_name<<endl;

    cout<<"1111111111111111111111"<<endl;

    \*/

}

//初始化目录

void init\_directory()

{

    df = (DF\*)malloc(sizeof(DF));

    df\_now = (DF\*)malloc(sizeof(DF));

    init\_DF(df);//初始化DF根目录

    df\_now=df;

    init\_df\_0(df\_now);//初始化自定义的目录

}

//查看当前文件下目录的内容

void get\_DF(DF\* temp\_df)

{

    cout<<"当前目录下的内容如下：";

    DF\* t=temp\_df;

    while(1)

    {

        if(t->df\_front!=NULL)

        {

            t=t->df\_front;

        }

        else

        {

            break;

        }

    }

    while(1)

    {

        cout<<t->df\_name<<" ";

        if(t->df\_next!=NULL)

        {

            t=t->df\_next;

        }

        else

        {

            break;

        }

    }

    cout<<endl;

    init\_1();

    solving\_1();

}

//返回父目录

void ret\_DF(DF\* temp\_df)

{

    DF\* t=temp\_df;

    if(t->df\_father==NULL)

    {

        cout<<"该目录为根目录，无父目录！"<<endl;

    }

    else

    {

        t=t->df\_father;

        df\_now=t;

    }

    system("pause");

    system("cls");

    init\_1();

    solving\_1();

}

//进入当前目录的子目录

void gin\_DF(DF\* temp\_df)

{

    //cout<<df\_now->df\_name<<endl;

    cout<<"请输入要进入的目录名：";

    char temp\_name[1010];

    string s;

    cin>>s;

    ll len\_s=ll(s.size())-1;

    for(ll i=0;i<=len\_s;i++)

    {

        temp\_name[i]=s[i];

    }

    DF\* t=temp\_df;

    bool flag=false;//定义是否寻找到目录

    while(1)

    {

        if(t->df\_front!=NULL)

        {

            t=t->df\_front;

        }

        else if(t->df\_front==NULL)

        {

            break;

        }

    }

    while(1)

    {

        bool st\_gin=true;

        for(ll i=0;i<=(ll)max((ll)strlen(t->df\_name),(ll)strlen(temp\_name))-(ll)1;i++)//表示查找到该名字的文件

        {

             if(t->df\_name[i]!=temp\_name[i])

            {

                st\_gin=false;

            }

        }

        if(st\_gin)

        {

            flag=true;

            break;

        }

        else if(t->df\_next==NULL)

        {

            break;

        }

        else

        {

            t=t->df\_next;

        }

    }

    if(flag==true)

    {

        //cout<<"+++++++++"<<endl;

        //cout<<t->df\_type<<endl;

        if(t->df\_type==0)//如果该文件的类型是目录的话，那么直接进入子目录

        {

            //cout<<t->df\_son\_0->df\_name;

            df\_now=t->df\_son\_0;

            df=t->df\_son\_0;

        }

        else

        {

            cout<<"该文件不是目录，无法进入"<<endl;

        }

    }

    else

    {

        cout<<"该目录名不存在！"<<endl;

    }

    //system("pause");

    init\_1();

    solving\_1();

}

//第三初始化界面->当前路劲下的操作

void init\_2()

{

    system("cls");

    cout<<"--------------------"<<endl;

    cout<<"1.增加目录/文件"<<endl;

    cout<<"2.删除目录/文件"<<endl;

    cout<<"3.读取文件内容"<<endl;

    cout<<"4.编写文件内容"<<endl;

    cout<<"5.返回上一页"<<endl;

}

//增加目录/文件

void add\_df()

{

    if((temp\_Permissions==1||temp\_Permissions==0)!=1)

    {

        cout<<"对不起！您没有权限增加目录/文件！"<<endl;

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else if(df\_now->df\_father==NULL)

    {

        cout<<"对不起！当前目录为根目录，无法增加并列目录或文件！\n";

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else

    {

        st\_choice\_3=true;

        cout<<"----------------------------"<<endl;

        cout<<"1.文件"<<endl;

        cout<<"2.目录"<<endl;

        while(st\_choice\_3)

        {

            cout<<"请输入要增加的类型：";

            cin>>choice\_3;//增加文本类型的选择；

            if(choice\_3==1||choice\_3==2)

            {

                st\_choice\_3=false;

            }

        }

        cout<<"请输入你要增加的文本名:";

        char temp\_name[1010];

        scanf("%s",&temp\_name);

        //cout<<"---------"<<endl;

        DF\* temp\_df;

        temp\_df=(DF\*)malloc(sizeof(DF));

        //cout<<"---------"<<endl;

        strcpy(temp\_df->df\_name,temp\_name);

        //cout<<"---------"<<endl;

        temp\_df->df\_father=df\_now->df\_father;//将新加的文本的父目录改为当前的目录

        //cout<<"---------"<<endl;

        temp\_df->df\_front=df\_now->df\_father->df\_son\_1;//将新加的文本的上一个改为当前目录的最后一个

        //cout<<"---------"<<endl;

        //cout<<df\_now->df\_son\_1->df\_name<<endl;

        //cout<<temp\_df->df\_name<<endl;

        df\_now->df\_father->df\_son\_1->df\_next=temp\_df;//将当前文本的最后一个的下一个改为新加的文本

        //cout<<"---------"<<endl;

        df\_now->df\_father->df\_son\_1=temp\_df;//最后更改当前目录的最后一个为新加的文本

        //cout<<"---------"<<endl;

        temp\_df->df\_next=NULL;//更新新加的文本的下一个为空

        //cout<<"---------"<<endl;

        if(choice\_3==1)

        {

            temp\_df->df\_type=1;

            temp\_df->df\_Permissions=1;

        }

        else

        {

            temp\_df->df\_type=0;

            temp\_df->df\_Permissions=0;

        }

        cout<<"增加成功！"<<endl;

        system("pause");

        system("cls");

        init\_1();

        solving\_1();

    }

}

//删除目录文件

void del\_df()

{

    if((temp\_Permissions==1||temp\_Permissions==0)!=1)

    {

        cout<<"对不起！您没有权限删除目录/文件！"<<endl;

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else if(df\_now->df\_father==NULL)

    {

        cout<<"对不起！当前目录为根目录，无法删除该目录！\n";

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else

    {

        cout<<"请输入你要删除的文本名:";

        char temp\_name[1010];

        scanf("%s",&temp\_name);

        DF\* temp\_df;

        temp\_df=(DF\*)malloc(sizeof(DF));

        temp\_df=df\_now->df\_father->df\_son\_0;

        //cout<<"================"<<endl;

        //cout<<df\_now->df\_father->df\_name<<" "<<df\_now->df\_father->df\_son\_0->df\_name<<" "<<df\_now->df\_father->df\_son\_1->df\_name<<endl;

        //cout<<temp\_df->df\_name<<" "<<temp\_name<<endl;

        //cout<<"================"<<endl;

        while(1)

        {

            st\_gin\_3=true;

            for(ll i=0;i<=(ll)max((ll)strlen(temp\_df->df\_name),(ll)strlen(temp\_name))-(ll)1;i++)//表示查找到该名字的文件

            {

                if(temp\_df->df\_name[i]!=temp\_name[i])

                {

                    st\_gin\_3=false;

                }

            }

            if(st\_gin\_3)

            {

                bool st\_gin\_3\_1=true;

                bool st\_gin\_3\_2=true;

                //cout<<"================"<<endl;

                for(ll i=0;i<=(ll)max((ll)strlen(df\_now->df\_father->df\_son\_0->df\_name),(ll)strlen(temp\_name))-(ll)1;i++)

                {

                    if(df\_now->df\_father->df\_son\_0->df\_name[i]!=temp\_name[i])

                    {

                        st\_gin\_3\_1=false;

                    }

                }

                for(ll i=0;i<=(ll)max((ll)strlen(df\_now->df\_father->df\_son\_1->df\_name),(ll)strlen(temp\_name))-(ll)1;i++)

                {

                    if(df\_now->df\_father->df\_son\_1->df\_name[i]!=temp\_name[i])

                    {

                        st\_gin\_3\_2=false;

                    }

                }

                if(st\_gin\_3\_1)

                {

                    //cout<<"1==============="<<endl;

                    df\_now->df\_father->df\_son\_0=temp\_df->df\_next;

                    //cout<<"==============="<<endl;

                    df\_now->df\_father->df\_son\_0->df\_front=NULL;

                    df\_now=temp\_df->df\_next;

                    //cout<<"==============="<<endl;

                    temp\_df->df\_father=NULL;

                    //cout<<"==============="<<endl;

                    temp\_df->df\_next=NULL;

                    //cout<<"==============="<<endl;

                    st\_del=true;

                    //cout<<"==============="<<endl;

                    break;

                }

                else if(st\_gin\_3\_2)

                {

                    //cout<<"2==============="<<endl;

                    df\_now->df\_father->df\_son\_1=temp\_df->df\_front;

                    df\_now->df\_father->df\_son\_1->df\_next=NULL;

                    temp\_df->df\_next=NULL;

                    temp\_df->df\_father=NULL;

                    st\_del=true;

                    break;

                }

                else

                {

                    temp\_df->df\_next->df\_front=temp\_df->df\_front;

                    temp\_df->df\_front->df\_next=temp\_df->df\_next;

                    temp\_df->df\_father=NULL;

                    temp\_df->df\_next=NULL;

                    temp\_df->df\_front=NULL;

                    st\_del=true;

                    //cout<<"====3==="<<endl;

                    break;

                }

            }

            else if(temp\_df==df\_now->df\_son\_1)

            {

                //cout<<"====4==="<<endl;

                break;

            }

            else

            {

                temp\_df=temp\_df->df\_next;

            }

        }

        //cout<<"==============="<<endl;

        if(st\_del)

        {

            cout<<"删除成功！"<<endl;

            system("pause");

            system("cls");

            init\_1();

            solving\_1();

        }

        else

        {

            cout<<"文件不存在"<<endl;

            system("pause");

            system("cls");

            mak\_DF();

        }

    }

}

//读取文件内容

void red\_df(DF\* temp\_df)

{

    if((temp\_Permissions==1||temp\_Permissions==0||temp\_Permissions==2)!=1)

    {

        cout<<"对不起！您没有权限读取目录/文件！"<<endl;

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else if(df\_now->df\_father==NULL)

    {

        cout<<"对不起！该文本类型为目录，无法读取！\n";

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    bool st\_red\_1=true;

    cout<<"请输入你要读取的文本名字"<<endl;

    char temp\_name[1010];

    scanf("%s",&temp\_name);

    DF\* t\_3;

    t\_3=(DF\*)malloc(sizeof(DF));

    t\_3=temp\_df->df\_father->df\_son\_0;

    while(1)

    {

        st\_red\_1=true;

        for(ll i=0;i<=max((ll)strlen(temp\_name),(ll)strlen(t\_3->df\_name))-(ll)1;i++)

        {

            if(t\_3->df\_name[i]!=temp\_name[i])

            {

                st\_red\_1=false;

            }

        }

        if(st\_red\_1)

        {

            if(t\_3->df\_type==1)

            {

                cout<<"该文件内容为:"<<endl;

                cout<<t\_3->df\_main<<endl;

                system("pause");

                mak\_DF();

                break;

            }

            else

            {

                cout<<"对不起，该文本类型为目录，无法读取！"<<endl;

                system("pause");

                mak\_DF();

                break;

            }

        }

        else if(t\_3->df\_name==df\_now->df\_father->df\_son\_1->df\_name)

        {

            cout<<"对不起，该文本名不存在！"<<endl;

            system("pause");

            mak\_DF();

            break;

        }

        else

        {

            t\_3=t\_3->df\_next;

        }

    }

    cout<<"对不起，该文本不存在！"<<endl;

    system("pause");

    mak\_DF();

}

//编写文件内容

void wre\_df(DF\* temp\_df)

{

    if((temp\_Permissions==1||temp\_Permissions==0)!=1)

    {

        cout<<"对不起！您没有权限增加目录/文件！"<<endl;

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    else if(df\_now->df\_father==NULL)

    {

        cout<<"对不起！该文本类型为目录，无法修改！\n";

        system("pause");

        system("cls");

        init\_2();

        solving\_2();

    }

    bool st\_wre\_1=true;

    cout<<"请输入你要修改的文本名字"<<endl;

    char temp\_name[1010];

    scanf("%s",&temp\_name);

    DF\* t\_3;

    t\_3=(DF\*)malloc(sizeof(DF));

    t\_3=temp\_df->df\_father->df\_son\_0;

    while(1)

    {

        st\_wre\_1=true;

        for(ll i=0;i<=max((ll)strlen(temp\_name),(ll)strlen(t\_3->df\_name))-(ll)1;i++)

        {

            if(t\_3->df\_name[i]!=temp\_name[i])

            {

                st\_wre\_1=false;

            }

        }

        if(st\_wre\_1)

        {

            if(t\_3->df\_type==1)

            {

                ll len\_rem=0;

                cout<<"请输入修改后的内容:"<<endl;

                cout<<"以#结尾"<<endl;

                char ch;

                while(ch=getchar())

                {

                    if(ch=='#')

                    {

                        break;

                    }

                    t\_3->df\_main[len\_rem]=ch;

                    len\_rem++;

                }

                system("pause");

                mak\_DF();

                break;

            }

            else

            {

                cout<<"对不起，该文本类型为目录，无法修改！"<<endl;

                system("pause");

                mak\_DF();

                break;

            }

        }

        else if(t\_3->df\_name==df\_now->df\_son\_1->df\_name)

        {

            cout<<"对不起，该文本名不存在！"<<endl;

            system("pause");

            mak\_DF();

            break;

        }

        else

        {

            t\_3=t\_3->df\_next;

        }

    }

    cout<<"对不起，该文本不存在！"<<endl;

    system("pause");

    mak\_DF();

}

//返回上一页

void ref\_df()

{

    init\_1();

    solving\_1();

}

//第三初始化界面->选项

void solving\_2()

{

    cout<<"请输入你的选项:";

    cin>>choice\_2;//第三初始界面的选择

    switch(choice\_2)

    {

        case 1:

            add\_df();//增加目录/文件

            break;

        case 2:

            del\_df();//删除目录/文件

            break;

        case 3:

            //cout<<df\_now->df\_name<<endl;

            red\_df(df\_now);//读取目录/文件

            break;

        case 4:

            wre\_df(df\_now);//编写目录/文件

            break;

        case 5:

            ref\_df();//返回上一页

            break;

    }

}

//当前路径下的操作

void mak\_DF()

{

    init\_2();//第三初始化界面->当前路劲下的操作

    solving\_2();//第三初始界面->选项

}

//第二初始界面->选项

void solving\_1()

{

    //cout<<endl<<df\_now->df\_son\_0->df\_name<<endl;

    cin>>choice\_1;

    switch(choice\_1)

    {

        case 1:

            get\_DF(df\_now);//查看当前文件目录下的内容

            break;

        case 2:

            ret\_DF(df\_now);//返回当前文件的父目录

            break;

        case 3:

            gin\_DF(df\_now);//进入当前目录的子目录

            break;

        case 4:

            mak\_DF();//当前路劲下的操作

            break;

        case 5:

            end();//退出系统

            break;

    }

}

//主函数

int main()

{

    init\_user();//初始化最高特权用户

    init\_0();//第一初始界面->用户登录|注册

    solving\_0();//第一初始界面->选项

    init\_directory();//初始化目录

    init\_1();//第二初始界面->文件管理系统的操作

    solving\_1();//第二初始界面->选项

    return 0;

}

《操作系统》课程设计评分表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **学 院** |  | **专业** | |  | | |
| **学生姓名** |  | **学号** | |  | **任课教师** | **高双喜** |
| **题目** |  | | | | | |
| **评价项目** | **具体项目** | | | | | 评分 |
| **能力水平**  **（80分）** | 分析设计内容，给出解决方案（10分） | | | | |  |
| 画出程序的基本结构框图或流程图（10分） | | | | |  |
| 对程序的每一部分要有适当的设计分析说明（10分） | | | | |  |
| 源代码格式规范（10分） | | | | |  |
| 设计合适的测试用例，对得到的运行结果有分析（10分） | | | | |  |
| 设计中遇到的问题，设计的心得体会（10分） | | | | |  |
| 综合运用本课程所学理论和专业知识（20分） | | | | |  |
| **报告质量**  **（20分）** | 报告完整并符合标准，写作规范，条理清楚（10分） | | | | |  |
| 论述充分，文字通顺，符合技术用语要求（10分） | | | | |  |
| **评阅人** |  | | 总分 | | |  |