

# 目录

<b>1</b>	<b>实验概述</b>	<b>1</b>
1.1	实验目的	1
1.2	实验要求	1
1.3	设计报告要求	3
1.4	实验环境	3
<b>2</b>	<b>需求分析</b>	<b>4</b>
2.1	程序输入形式	4
2.2	程序输出形式	10
2.3	程序功能	11
<b>3</b>	<b>概要设计</b>	<b>13</b>
3.1	任务分解	13
3.2	数据结构定义	13
3.3	模块间调用关系	17
3.4	算法说明	17
<b>4</b>	<b>详细设计</b>	<b>21</b>
4.1	内存管理模块	21
4.1.1	内存块读写	21
4.1.2	分配一个 Block 块	22
4.1.3	回收一个 Block 块	23
4.1.4	Inode 中逻辑块号与物理块号的映射	23
4.2	目录管理模块	24
4.2.1	创建目录	24
4.2.2	删除目录	24
4.2.3	打开目录	25
4.2.4	获取当前目录	25
4.3	文件管理模块	25
4.3.1	创建文件	25
4.3.2	更改文件指针	26
4.3.3	读文件	26
4.3.4	更改文件权限	26
4.3.5	删除文件	26

4.3.6	打开文件 . . . . .	27
4.3.7	写文件 . . . . .	27
4.4	用户管理模块 . . . . .	28
4.4.1	登录用户 . . . . .	28
4.4.2	退出登录 . . . . .	28
4.4.3	创建用户 . . . . .	28
4.4.4	删除用户 . . . . .	29
4.4.5	改变用户所属组 . . . . .	29
<b>5</b>	<b>运行结果分析</b>	<b>30</b>
5.1	运行结果 . . . . .	30
5.2	命令测试 . . . . .	32
5.2.1	课设要求测试 . . . . .	32
5.2.2	命令逐条测试 . . . . .	35
5.2.3	其他测试 . . . . .	40
5.3	结果分析 . . . . .	43
<b>6</b>	<b>用户使用说明</b>	<b>44</b>
<b>7</b>	<b>实验总结</b>	<b>46</b>
7.1	实验收获 . . . . .	46
7.2	问题及解决方案 . . . . .	46
7.3	思考与体会 . . . . .	47
	<b>参考文献</b>	<b>48</b>

# 1 实验概述

## 1.1 实验目的

UNIX 文件系统提供了层次结构的目录和文件，负责系统内文件信息的管理，是 UNIX 系统中极为重要的一个部分。本次实验的目的是通过模拟 UNIX V6++ 编写一个 UNIX 文件系统，实现基本文件操作，从而掌握 UNIX 文件系统结构。

## 1.2 实验要求

使用一个普通的大文件（如 c: myDisk.img，称之为一级文件）来模拟 UNIX V6++ 一个文件卷（把一个大文件当一张磁盘用）。

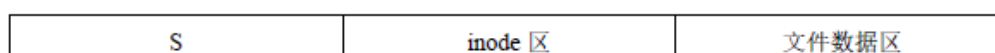


图 1: 文件卷结构

一个文件卷实际上就是一张逻辑磁盘，磁盘存储的信息以块为单位，每块 512 字节。

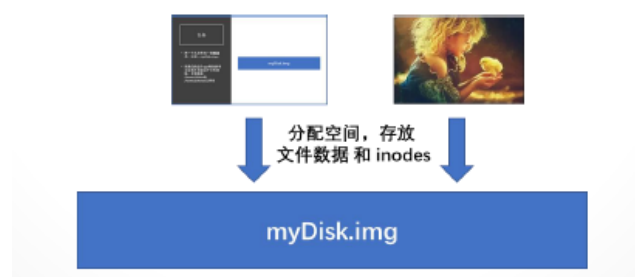


图 2: 使用文件卷模拟逻辑磁盘

### (1) 磁盘文件结构

- 定义自己的磁盘文件结构
- SuperBlock 结构
- 磁盘 Inode 节点结构，包括：索引结构
- 磁盘 Inode 节点的分配与回收算法设计与实现
- 文件数据区的分配与回收算法设计与实现

### (2) 文件目录结构

- 目录文件结构

- 目录检索算法的设计与实现

(3) 文件打开结构

(4) 磁盘高速缓存：选作

(5) 文件操作接口

- fformat：格式化文件卷
- ls：列目录
- mkdir：创建目录
- fcreat：新建文件
- fopen：打开文件
- fclose：关闭文件
- fread：读文件
- fwrite 写文件
- fseek 定位文件读写指针
- fdelete：删除文件
- .....

(6) 主程序：

- 格式化文件卷
- 用 mkdir 命令创建子目录，建立如图所示的目录结构

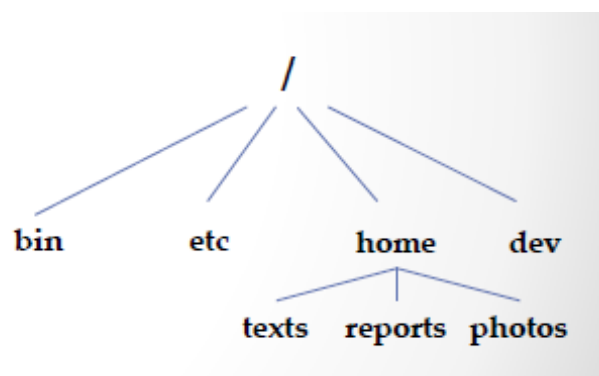


图 3: 文件目录结构

- 把你的课设报告，关于课程设计报告的 ReadMe.txt 和一张图片存进这个文件系统，分别放在 /home/texts， /home/reports 和 /home/photos 文件夹；

- 图形界面或者命令行方式，等待用户输入；
- 根据用户不同的输入，返回结果。

#### 1. 通过命令行方式测试：

- 新建文件 /test/Jerry，打开该文件，任意写入 800 个字节；
- 将文件读写指针定位到第 500 字节，读出 500 个字节到字符串 abc。
- 将 abc 写回文件。

观察结果是否正确，并详细解释每一步的工作过程

### 1.3 设计报告要求

- 实验报告内容
  - 需求分析（10%）：说明程序任务，包括输入、输出形式，程序功能。
  - 概要设计（15%）：任务分解；数据结构定义；模块间的调用关系，算法说明等。
  - 详细设计（30%）：重点函数的重点变量需说明，重点功能部分要绘制清晰的程序流程图。画出函数调用关系。
  - 运行结果分析（30%）：
    - \* 程序运行结果展示说明；
    - \* 测试命令及输出结果，结果分析。
  - 用户使用说明（5%）
  - 实验总结（5%）：包括综合实验过程的收获、遇到问题及解决问题过程的思考、在综合实验过程中对课程的认识等内容。
- 评分标准
  - 视系统及实验报告的完成情况：不及格 良
  - 加分项（完成下列内容之一）
    - \* 实现内存高速缓存，实验报告中需说明缓存数据结构与管理算法
    - \* 允许多个用户同时访问二级文件系统

### 1.4 实验环境

操作系统：Windows 10 家庭版开发语言：C++ 编译器：Microsoft Visual Studio Community 2019

## 2 需求分析

### 2.1 程序输入形式

根据对题目要求分析和用户使用分析，文件系统采用控制台命令作为输入，按照系统给定的命令即可完成相应的功能。本文件模拟系统使用尽可能详细的控制台引导用户输入指令。

用户可以在控制台输入的指令如下：

- HELP——帮助文档。
- ATTRIB——显示或更改文件属性。
- CD——显示当前目录的名称或将其更改。
- DEL——删除至少一个文件。
- DIR——显示一个目录中的文件和子目录。
- EXIT——退出文件系统。
- MKDIR——创建一个目录。
- RMDIR——删除目录。
- PRINT——打印文件内容。
- WRITE——向文件中写入内容
- OPEN——打开一个文件
- CLOSE——关闭一个文件
- CREATE——创建一个文件
- OPENLIST——当前打开文件列表
- FSEEK——更改一个文件的指针
- LOGOUT——用户退出登录
- WHOAMI——显示当前用户信息
- FORMAT——格式化文件卷

- REGISTER——-用户注册
- DELETEACCOUNT-删除用户（root 用户下）
- SU———-改变用户
- CHGRP———-改变用户所属组（root 用户下）
- USERLIST——-显示所有用户信息（root 用户下）

使用 help 命令查看各个指令的详细内容如下：

- ATTRIB 显示或更改文件属性。

```
root>help attrib
更改文件的读写属性

ATTRIB [+R | -R] [+W | -W] [+E | -E] [O | G | E] [path][filename]

+  设置属性。
-  清除属性。
R  读文件属性。
W  写文件属性。
E  执行文件属性。
/O 文件所有者用户。
/G 文件同组者用户。
/E 非文件所有者、非文件同组者用户
[path][filename]
指定属性要处理的文件。
```

图 4: ATTRIB 指令

- CD 显示当前目录的名称或将其更改。

```
root>help cd
显示当前目录名或改变当前目录。

CD [path]

不带参数只键入 CD，则显示当前驱动器和目录。
path规则：.代表本目录，..代表父目录，路径间以/或\分隔
```

图 5: CD 指令

- DEL 删除至少一个文件。

```
root>help DEL
删除一个或多个文件。

DEL names

删除多个或一个文件。
有用户未关闭文件时不能删除文件。
```

图 6: DEL 指令

- DIR 显示一个目录中的文件和子目录。

```
root>help DIR
显示目录中的文件和子目录列表。

DIR [/Q]

/Q 显示详细信息。
```

图 7: DIR 指令

- EXIT 退出文件系统。

```
root>help EXIT
退出程序(文件系统)。

EXIT [exitCode]

exitCode 指定一个数字号码。如果退出，则用那个数字设置过程退出代码。
```

图 8: EXIT 指令

- MKDIR 创建一个目录。

```
root>help MKDIR
创建目录。

MKDIR dir

dir 创建的目录名称。
```

图 9: MKDIR 指令

- RMDIR 删除目录。

```
root>help RMDIR
删除一个目录。

RMDIR dir

dir 待删除的目录名称（仅有空目录可以被删除）。
```

图 10: RMDIR 指令

- PRINT 打印文件内容。



```
root>help PRINT
打印文本文件。

PRINT filename [-l length] [-p path]

filename 打印的文件名称。
length   打印的长度，默认全部打印
path     可选，打印到文件系统所在目录的文件名称。
```

图 11: PRINT 指令

- WRITE 向文件中写入内容

```
root>help WRITE
打印文本文件。

WRITE [filename] [-s|-f] [path|content]

filename 写入的文件名称。
-s       从屏幕写入。
-f       从文件写入。
path     写入到文件系统所在目录的文件名称。。
content  写入文件内容。
```

图 12: WRITE 指令

- OPEN 打开一个文件

```
root>help open
打开文件。

OPEN [filename]

filename 打开的文件名称。
```

图 13: OPEN 指令

- CLOSE 关闭一个文件

```
root>help CLOSE
关闭文件。

CLOSE [filename]

filename 关闭的文件名称。
```

图 14: CLOSE 指令

- CREATE 创建一个文件

```
root>help CREATE
创建文件。

CREATE filename

filename 文件名称。
```

图 15: CREATE 指令

- OPENLIST 当前打开文件列表

```
root>help openlist
显示所有打开文件列表

OPENLIST
```

图 16: OPENLIST 指令

- FSEEK 更改一个文件的指针

```
root>help FSEEK
更改文件指针。

FSEEK filename pos

filename 文件名称。
pos 文件指针定位位置。
```

图 17: FSEEK 指令

- LOGOUT 用户退出登录

```
root>help LOGOUT
用户登出。

LOGOUT
```

图 18: LOGOUT 指令

- WHOAMI 显示当前用户信息

```
root>help WHOAMI
获取当前登录用户。

WHOAMI
```

图 19: WHOAMI 指令

- FORMAT 格式化文件卷

```
root>help FORMAT
格式化文件卷。

FORMAT
```

图 20: FORMAT 指令

- REGISTER 用户注册

```
root>help REGISTER
创建新用户。

REGISTER username password
username 用户名。
password 用户密码。
```

图 21: REGISTER 指令

- DELETEACCOUNT 删除用户（root 用户下）

```
root>help DELETEACCOUNT
删除用户。

DELETEACCOUNT username
username 用户名。
```

图 22: DELETEACCOUNT 指令

- SU 改变用户

```
root>help SU
改变用户

SU username password
username 用户名。
password 用户密码。
```

图 23: SU 指令

- CHGRP 改变用户所属组（root 用户下）

```
root>help CHGRP
改变用户所属组

CHGRP username usergroupid
username 用户名。
usergroupid 用户所属组。
```

图 24: CHGRP 指令

- USERLIST 显示所有用户信息（root 用户下）

```
root>help USERLIST
显示所有用户信息

USERLIST
```

图 25: USERLIST 指令

## 2.2 程序输出形式

程序通过控制台命令行交互的方式进行用户输入引导和显示反馈输出，具体输出将在后续进行介绍。

```
=====
仿UNIX V6++文件系统
鞠璇 1851846
=====

有关某个命令的详细信息，请键入 HELP 命令名
HELP          帮助文档。
ATTRIB        显示或更改文件属性。
CD            显示当前目录的名称或将其更改。
DEL           删除至少一个文件。
DIR           显示一个目录中的文件和子目录。
EXIT          退出文件系统。
MKDIR         创建一个目录。
RMDIR         删除目录。
PRINT         打印文件内容。
WRITE         向文件中写入内容
OPEN          打开一个文件
CLOSE         关闭一个文件
CREATE        创建一个文件
OPENLIST      当前打开文件列表
FSEEK         更改一个文件的指针
LOGOUT        用户退出登录
WHOAMI        显示当前用户信息
FORMAT        格式化文件卷
REGISTER      用户注册
DELETEACCOUNT 删除用户（root用户下）
SU            改变用户
CHGRP         改变用户所属组（root用户下）
USERLIST      显示所有用户信息（root用户下）

是否进行文件系统初始化？（y/n）

用户未登录，请输入用户名和密码
用户名: root
密码: root
root>help openlist
显示所有打开文件列表
```

图 26: 程序输出

## 2.3 程序功能

实验模拟 UNIX 文件操作系统实现了多用户下文件的一系列操作。具体可以实现的功能如下所示：

- 多用户下文件的创建、删除
- 多用户下文件的打开、关闭、读、写、读写指针移动
- 当前打开文件列表的显示
- 多用户下更改文件读写属性（权限）
- 多用户下目录的创建、删除
- 多用户下目录的更改
- 一个目录下文件和子目录的显示

- 用户的创建、删除
- 用户的登录、退出、更改
- 修改用户所属组
- 显示当前用户信息
- 显示所有用户信息
- 格式化文件卷
- 退出文件系统
- 帮助文档

## 3 概要设计

### 3.1 任务分解

整个多用户文件系统可以划分为以下几个模块：

- 内存管理模块：主要负责 SuperBlock、Inode、Block 的初始化、申请和释放。提供读入读出 SuperBlock、Inode、Block 块以及申请释放 Block 块的接口，同时提供每个 Inode 中逻辑 Block 块号和物理 Block 块号映射关系转换接口。
- 文件管理模块：主要负责文件相关的一系列操作。提供创建文件、删除文件、显示文件列表、打开文件关闭文件、写文件、读文件、更改文件指针、更改文件权限等操作的相关接口。
- 用户管理模块：主要负责用户相关的一系列操作。提供用户登录、用户登出、创建用户、删除用户、获取当前登录用户信息、更改用户所属组、输出用户列表信息等操作的相关接口。
- 目录管理模块：主要负责文件目录相关的一系列操作。提供创建目录、删除目录、打开目录、获取当前目录等操作的相关接口
- 顶层管理模块：主要负责用户输入信息处理、将用户输入命令转化为内核操作、反馈信息输出、错误反馈等。

### 3.2 数据结构定义

整个文件系统包含以下几个数据结构的定义：

- Inode 结构体：

Inode 结构体中存储一个 Inode，大小为 64 字节，4 个 Inode 占满一个 BLock 块。

其内部有两个枚举类型 INodeMode 和 INodePermission，分别用于枚举 Inode 指向的为文件还是目录、该目录/文件的读写属性，其中读写属性分为文件主、文件主同组和其他用户三组，OWNER\_R 为文件主可读，OWNER\_W 为文件主可写，OWNER\_E 为文件主可执行，GROUP\_R 为文件主同组可读，GROUP\_W 为文件主同组可写，GROUP\_E 为文件主同组可执行，ELSE\_R 为其他用户可读，ELSE\_W 为其他用户可写，ELSE\_E 为其他用户可执行。

Inode 结构体主要包含的成员如下：i\_addr 用于文件逻辑块和物理块的映射，其映射关系如图27所示，最多可以容纳  $128*128*2+128*2+6$  个 Block 块。i\_size 存储文件大小，以自己为单位，目录大小统一设置为 0，i\_count 为当前文件被打开的引用计数，

$i\_number$  为当前 Inode 的编号,  $i\_permission$  为文件的读写权限, 其值为 OWNER\_R、OWNER\_W、OWNER\_E、GROUP\_R、GROUP\_W、GROUP\_E、ELSE\_R、ELSE\_W、ELSE\_E 的组合, 当  $i\_permission$  与这几个值与运算后不为 0, 则代表当前 Inode 拥有这一权限,  $i\_uid$  为文件/目录创建者用户 id,  $i\_gid$  为文件/目录创建者用户组 id,  $i\_time$  为文件最后访问时间。

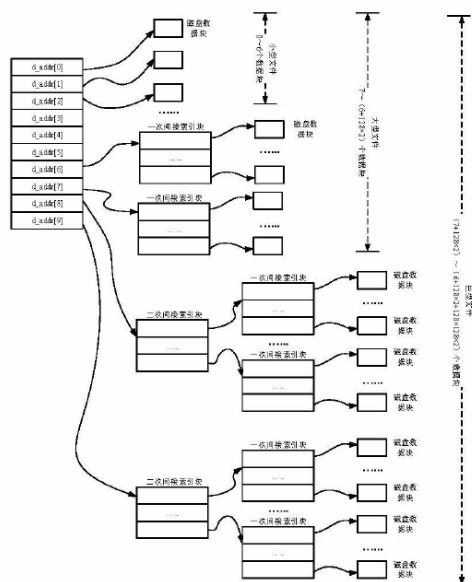


图 27:  $i\_addr$  索引图

```

1 //Inode结构体
2 struct Inode {
3
4     enum INodeMode {
5         IFILE = 0x1, //是文件
6         IDIRECTORY = 0x2 //是目录
7     };
8     enum INodePermission { //分为文件主、文件主同组和其他用
9         OWNER_R = 0400,
10        OWNER_W = 0200,
11        OWNER_E = 0100,
12        GROUP_R = 040,
13        GROUP_W = 020,
14        GROUP_E = 010,
15        ELSE_R = 04,

```



```

16         ELSE_W = 02,
17         ELSE_E = 01,
18     };
19
20     unsigned int i_addr[NINODE]; //逻辑块号和物理块号转换的
        索引表
21     unsigned int i_size; //文件大小, 字节为单位
22     unsigned short i_count; //引用计数
23     unsigned short i_number; //Inode的编号
24     unsigned short i_mode; //文件工作方式信息
25     unsigned short i_permission; //文件权限
26     unsigned short i_uid; //文件所有者的用户标识
27     unsigned short i_gid; //文件所有者的组标识
28     time_t i_time; //最后访问时间
29 };
    
```

- SuperBlock 结构体

SuperBlock 结构体中存储整个文件的 SuperBlock（文件系统超级块）。其中包含以下成员：s\_inodenum 存储 Inode 总数，s\_finodenum 存储空闲 Inode 数，s\_blocknum 存储 Block 总数，s\_fblocknum 存储空闲 Block 数，s\_nfree 和 s\_free 通过成组链接法管理所有空闲 Block，每一组的最大空闲 Block 数为 50（其中 0 号位用于存储下一个空闲表位置）。

```

1 //SuperBlock 结构体
2 struct SuperBlock {
3     unsigned short s_inodenum; //Inode 总数
4     unsigned short s_finodenum; //空闲 Inode 数
5     unsigned short s_blocknum; //Block 总数
6     unsigned short s_fblocknum; //空闲 Block 数
7     unsigned int s_nfree; //直接管理的空闲块数
8     unsigned int s_free[FREE_BLOCK_GROUP_NUM]; //空闲块索引
        表
9 };
    
```

- Directory 结构体

Directory 结构体用于存储目录列表，对于一个 i\_mode 为 IDIRECTORY 的 Inode，其

i\_addr[0] 中所指向的 Block 存储一个 Directory，该 Directory 结构体中的信息即为该目录的信息。

Directory 结构体有两个成员 d\_inodenum 和 d\_filename，分别用于存储该目录下的文件/目录的 Inode 号和名称。

```

1 //Directory结构体
2 struct Directory {
3     unsigned int d_inodenum [SUBDIRECTORY_NUM]; //子目录
        Inode号
4     char d_filename [SUBDIRECTORY_NUM] [FILE_NAME_MAX]; //子
        目录文件名
5 };

```

- User 结构体

User 结构体中存储所有的用户信息。User 结构体放置在 1 号 Block 中。

User 结构体包含四个成员：u\_id 存储用户 id、u\_gid 存储用户所在组 id、u\_name 存储用户名、u\_password 存储用户密码。用户 i 信息不存在时则 u\_name[i] 为空。

```

1 //User结构体
2 struct User {
3     unsigned short u_id [USER_NUM]; //用户 id
4     unsigned short u_gid [USER_NUM]; //用户所在组 id
5     char u_name [USER_NUM] [USER_NAME_MAX]; //用户名
6     char u_password [USER_NUM] [USER_PASSWORD_MAX]; //用户密
        码
7 };

```

- File 结构体

File 结构体存储打开文件信息。其包含的成员如下：f\_inodeid 为文件的 inode 编号、f\_offset 为文件的读写指针位置、f\_uid 为文件打开用户。File 结构体伴随打开文件和关闭文件存在消失。

```

1 //File结构体
2 struct File {
3     unsigned int f_inodeid; //文件的inode编号
4     unsigned int f_offset; //文件的读写指针位置
5     unsigned int f_uid; //文件打开用户
6 };

```

### 3.3 模块间调用关系

文件系统 myDisk.img 的空间分配如图28所示

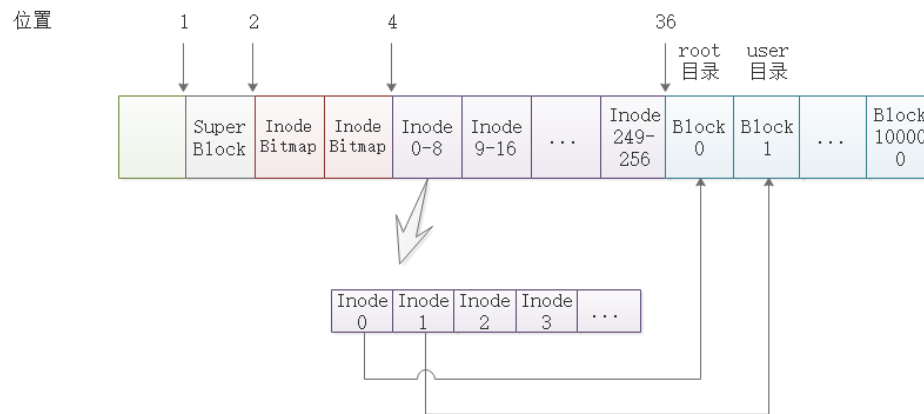


图 28: myDisk.img 的空间分配

各个模块之间的调用关系如图29所示：

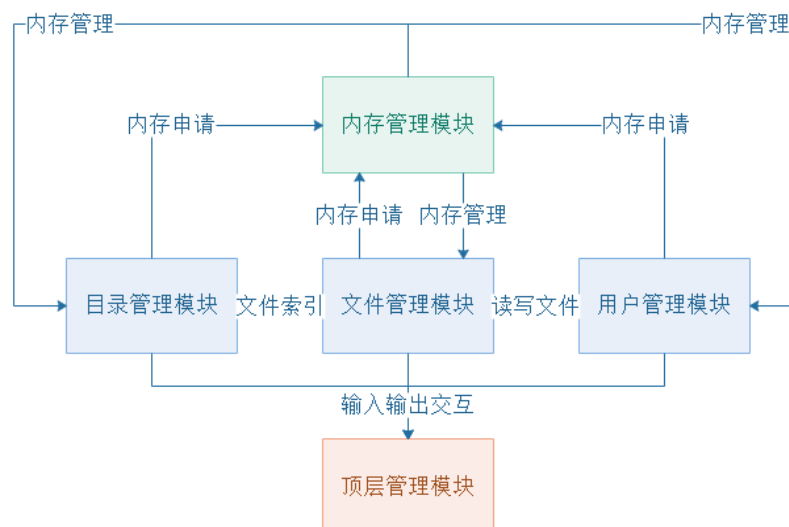


图 29: 模块间调用关系图

### 3.4 算法说明

整个算法的整体流程如图所示：

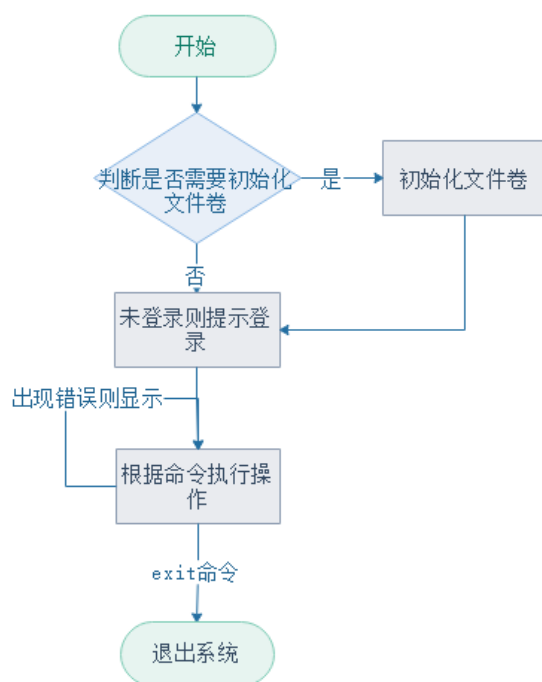


图 30: 整体流程图

其中，根据命令执行操作的流程具体如下：

- HELP

根据 help 后有无参数以及参数名，选择不同的提示输出。（具体可参考 2 需求分析模块）

- ATTRIB

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Edit\_File\_Permission 函数更改文件权限。

- CD

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则：如果 cd 后有参数，则根据参数调用 Open\_Directory 函数更改文件目录，如果 cd 后无参数，则输出当前所在目录。

- DEL

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数逐一调用 Delete\_File 函数删除文件。

- DIR

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Show\_File\_List 函数显示文件信息。

- EXIT

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则退出程序。

- MKDIR

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Create\_Directory 函数创建文件。

- RMDIR

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Remove\_Directory 函数删除目录。

- PRINT

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Read\_File 函数将文件内容读出到文件或屏幕。

- WRITE

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Write\_File 函数将文件或屏幕内容写入文件。

- OPEN

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Open\_File 函数打开文件。

- CLOSE

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Close\_File 函数打开文件。

- CREATE

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Create\_File 函数打开文件。

- OPENLIST

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则遍历输出所有打开文件信息。

- FSEEK

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Seek\_File 函数更改文件指针。

- LOGOUT

首先检查命令输入的正确性,若不正确则输出错误信息,若输入正确则调用 User\_Logout 函数使用户登出。

- WHOAMI

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则调用 Get\_User 函数输出当前用户信息。

- FORMAT

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则调用 Init 函数初始化文件卷。

- REGISTER

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 User\_Register 函数进行用户注册。

- DELETEACCOUNT

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 User\_Delete 函数删除文件。

- SU

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 User\_Logout 函数和 User\_Login 函数更改用户。

- CHGRP

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Change\_User\_Group 函数更改用户组。

- USERLIST

首先检查命令输入的正确性，若不正确则输出错误信息，若输入正确则根据参数调用 Show\_User\_List 函数展示用户表。

## 4 详细设计

顶层管理模块所涉及的部分已在前几部分进行了一定说明，下面对内存管理模块、目录管理模块、文件管理模块和用户管理模块的详细设计进行说明。

### 4.1 内存管理模块

#### 4.1.1 内存块读写

在文件操作的过程中，经常需要对 SuperBlock 块、Inode 块、User 表等部分进行读写，因此分离出 Read\_SuperBlock、Write\_SuperBlock、Read\_InodeBitMap、Write\_InodeBitMap、Read\_User、Write\_User、Read\_Inode、Write\_Inode 几个函数对内存进行读写。

其流程图如图31所示：

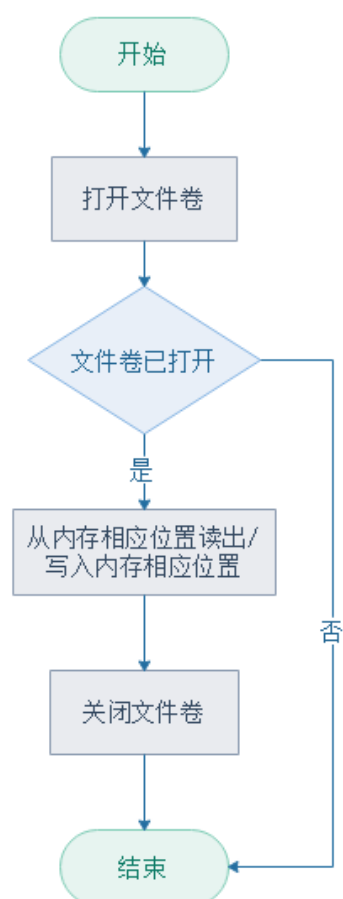


图 31: 内存块读写

#### 4.1.2 分配一个 Block 块

空闲盘块采用成组链接法进行管理（如图32），其流程图如图33所示。

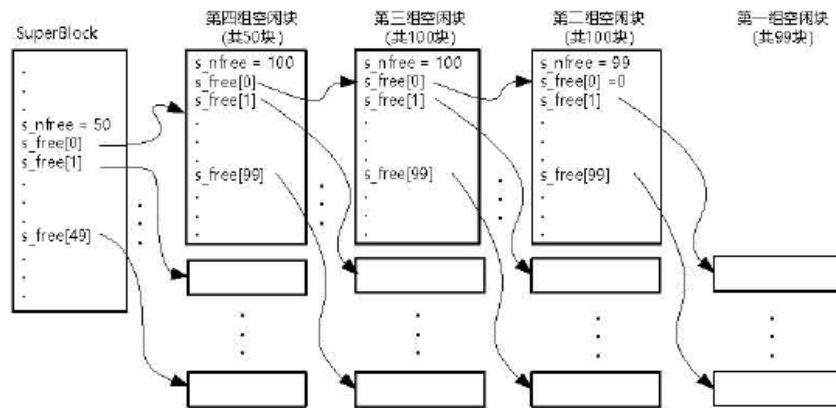


图 32: 成组链接法

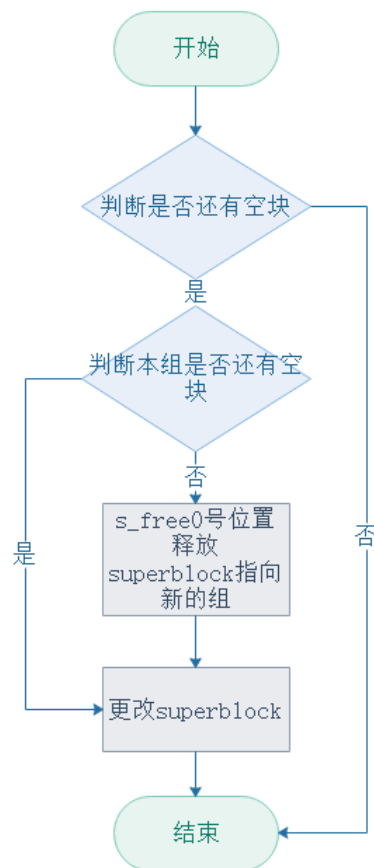


图 33: 分配一个 Block 块



### 4.1.3 回收一个 Block 块

回收 Block 块为分配 Block 块的逆过程，两者流程相似。

```

1 //如果该空闲块已满，需要使 superblock 指向一个新的空块
2     if (superblock.s_nfree == FREE_BLOCK_GROUP_NUM - 1) {
3         fd.open(DISK_NAME, ios::out | ios::in | ios::
4             binary);
5         fd.seekg((BLOCK_POSITION + block_num) * BLOCK_SIZE
6             , ios::beg);
7         fd.write((char*)&superblock.s_free, sizeof(
8             superblock.s_free));
9         fd.close();
10        superblock.s_free[0] = block_num;
11        superblock.s_nfree = 0;
12        superblock.s_fblocknum++;
13    }
14    else {
15        superblock.s_nfree++;
16        superblock.s_free[superblock.s_nfree] = block_num;
17    }

```

### 4.1.4 Inode 中逻辑块号与物理块号的映射

- 小型文件的索引结构

小型文件使用文件索引表中 `i_addr[0]-i_addr[5]` 这 6 项作为直接索引表，也就相当于限制了小型文件长度范围是 0-6 个数据块，每个盘块大小等于 512 字节。其中，文件逻辑块号 `n` 对应的物理块号记录在 `i_addr[n]` 中。譬如，`i_addr[2]` 的值为 2058 表示该文件的第 3 块数据（偏移量 1024 1535 字节）占据的物理盘块编号为 2058。

- 大型文件的索引结构

当文件的长度超出小型文件限制时，就需要采用大型文件的索引结构。大型文件除了使用 `i_addr[0]-i_addr[5]` 记录该文件前 6 个物理块号之外，还用到 `i_addr[6]`、`i_addr[7]` 两项，每一项指向一个含有物理块号明细的一次间接索引表块，每个一次间接索引表块可以容纳 512/4-128 个物理块号，可为 128 个文件的逻辑块和物理块建立对应关系。因此大型文件的长度范围是 7 (6+ 128 \*2) 个数据块。如果内核需要通过一次间接索引表块访问数据，那么先要将一次间接索引表块读入内存，从中找到逻辑块号对应的物理块号，然后再读取该物理块上的文件数据。

## • 巨型文件的索引结构

如果文件长度比大型文件还要大,则除了用到基本索引表中的  $i\_addr[0]-i\_addr[5]$  以及  $i\_addr[6]、i\_addr[7]$  两项记录该文件前  $128 * 2 + 6$  个物理块号之外,还要使用  $i\_addr[8]、i\_addr[9]$  来记录二次间接索引表块的物理共号。二次间接索引表块的作用类似于一次间接索引表,表中的每一项都记录者它所指向的一个一次间接索引表块的物理块号,然后再由该一次间接索引表块中的各项记录文件数据的物理块号。每张二次间接索引表块最多可记录 128 个一次间接索引表的物理块号,因此巨型文件的长度范围为  $(128 * 2 + 7) (128 * 128 * 2 + 128 * 2 + 6)$ 。

## 4.2 目录管理模块

### 4.2.1 创建目录

创建目录的流程图34如图所示,创建目录成功的前提是目录名合法、有空闲的 block 和 inode、当前目录不存在同名目录、当前目录的文件数量未达到限制

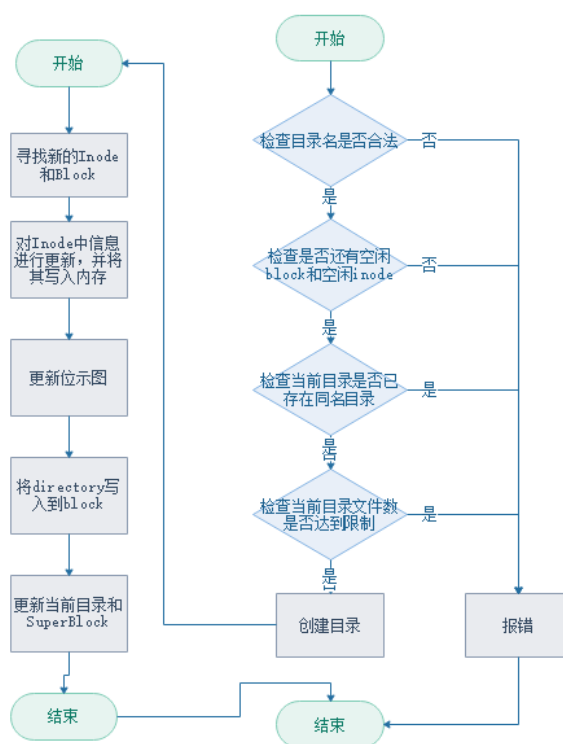


图 34: 创建目录流程图

### 4.2.2 删除目录

删除目录的流程图如图35所示,删除目录的前提是目录名合法、目录为空。

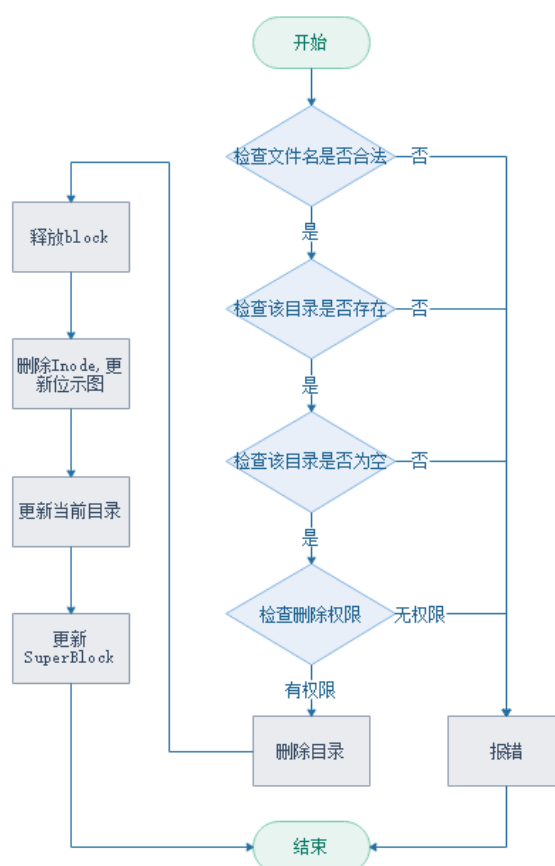


图 35: 删除目录流程图

### 4.2.3 打开目录

打开一个目录需要根据目录路径一级级进入相应的位置，目录路径使用' 或`/' 进行分割，'.' 代表当前目录，'..' 代表父目录，根据目录名寻找到 Inode 号进而更改对应的 directory 即可完成目录的切换

### 4.2.4 获取当前目录

从当前目录向上级目录一层层进行寻找，并使用' 进` 行连接，即可完成当前目录的获取

## 4.3 文件管理模块

### 4.3.1 创建文件

创建文件的流程图如图所示，创建文件成功的前提是文件名合法、有空闲的 block 和 inode、当前目录不存在同名文件、当前目录的文件数量未达到限制

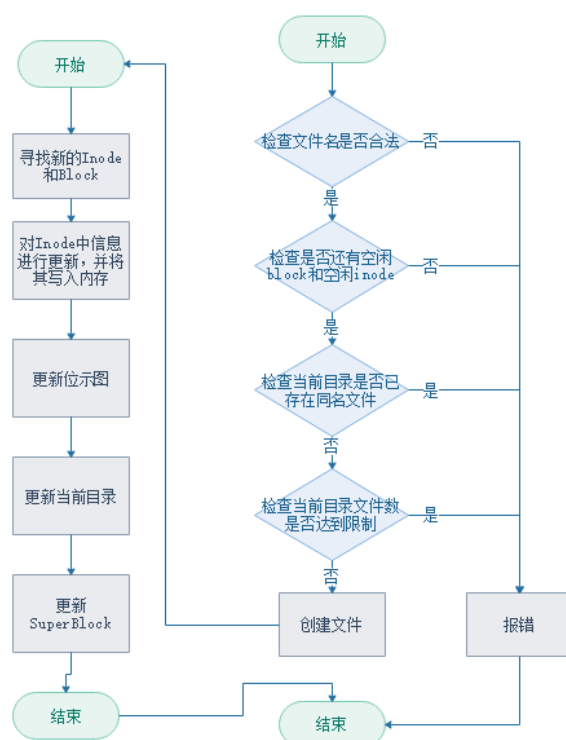


图 36: 创建文件流程图

### 4.3.2 更改文件指针

在保证当前用户为文件打开用户、用户具有写权限的前提下更改 File 结构体中的 `file->f_offset` 即可

### 4.3.3 读文件

在保证当前用户为文件打开用户、用户具有写权限的前提下，从 `file->f_offset` 指定位置开始依次取出 Block 块，直至读出文件长度等于 `length` 或读到文件末尾，在字符串末尾补尾零并返回读到字符串的长度。

### 4.3.4 更改文件权限

每个文件的权限共有三组，分别为主用户、同组用户、和其他用户，每一组用户权限又分为读权限、写权限以及执行权限，在相应的 Inode 中修改 `i_permission` 属性即可对该权限进行修改。

### 4.3.5 删除文件

删除文件的流程图如图所示，删除目录的前提是文件名合法、文件未被打开。

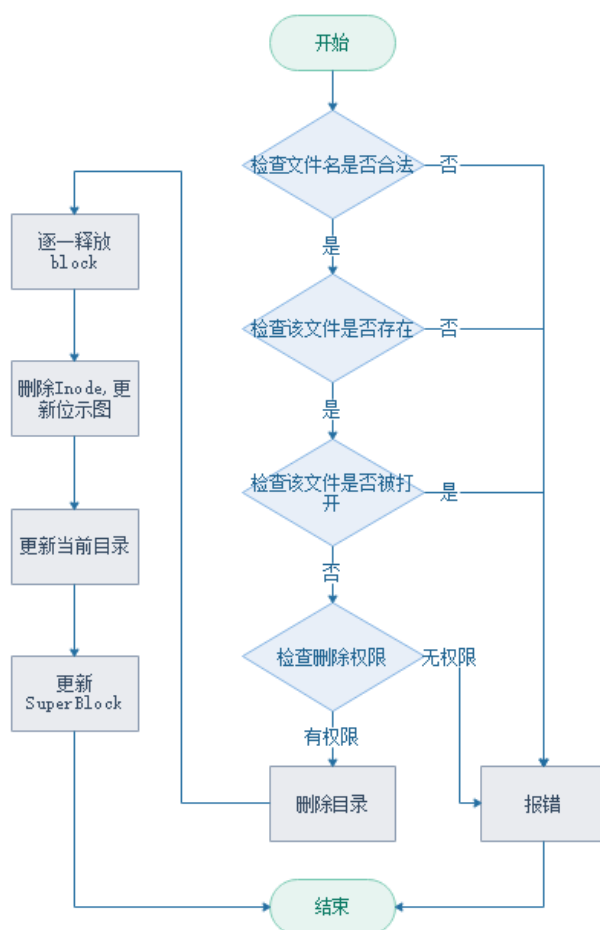


图 37: 删除文件流程图

#### 4.3.6 打开文件

首先检查文件名是否合法，如果合法则在当前目录下寻找这一文件，找到后新建 File 结构体，在其中赋值 `f_inodeid` 为找到的 `inode_num`，`f_offset` 为 0，`f_uid` 为打开文件的用户 id，并返回 file 结构体。

#### 4.3.7 写文件

写文件的流程如图所示，写文件需要保证当前用户为文件打开用户，用户具有写权限。

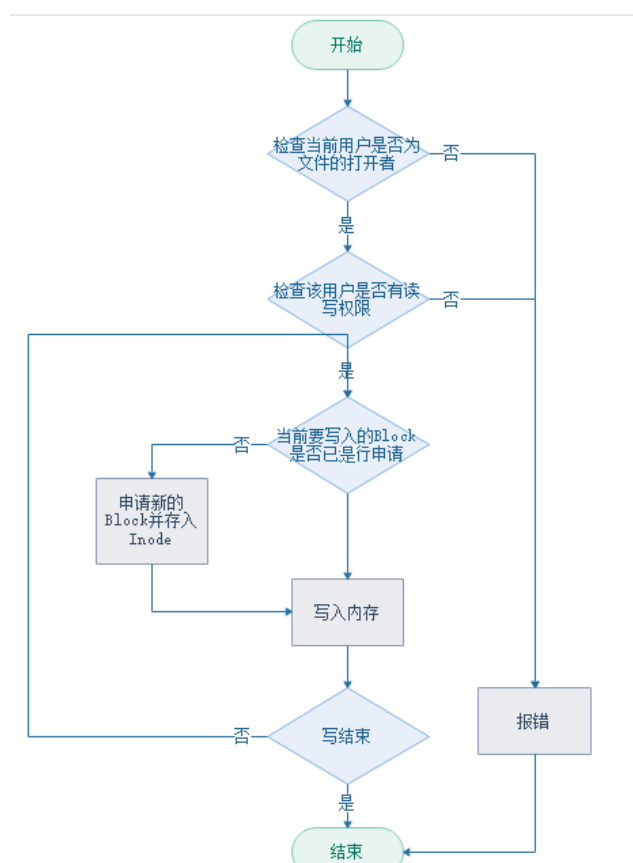


图 38: 写文件流程图

## 4.4 用户管理模块

### 4.4.1 登录用户

检查用户名和密码，若均正确，即可进行登录，若输入用户名错误，则提示此用户不存在，若输入密码错误，则输出密码错误

### 4.4.2 退出登录

将全局的 `user_id` 改为-1，即可完成用户登录的退出

### 4.4.3 创建用户

只有 `root` 用户可以进行用户的注册，注册前需要先检查该用户名是否已经存在，若存在，则给出报错，否则进行用户的新建。

#### 4.4.4 删除用户

只有 root 用户可以进行用户的删除，删除前需要先检查该用户是否存在，若存在则进行用户的删除，否则给出报错。

#### 4.4.5 改变用户所属组

只有 root 用户可以进行用户所属组的更改，更改前需要先检查该用户是否存在，若存在则进行用户的所属组更改，否则给出报错。

## 5 运行结果分析

### 5.1 运行结果

在初次打开 exe，文件系统不存在的情况下，会提示用户进行文件系统的初始化

```

=====
仿UNIX V6++文件系统
  鞠璇  1851846
=====
有关某个命令的详细信息，请键入 HELP 命令名
HELP      帮助文档。
ATTRIB    显示或更改文件属性。
CD         显示当前目录的名称或将其更改。
DEL        删除至少一个文件。
DIR        显示一个目录中的文件和子目录。
EXIT      退出文件系统。
MKDIR      创建一个目录。
RMDIR      删除目录。
PRINT      打印文件内容。
WRITE      向文件中写入内容。
OPEN       打开一个文件。
CLOSE      关闭一个文件。
CREATE     创建一个文件。
OPENLIST   当前打开文件列表。
FSEEK      更改一个文件的指针。
LOGOUT     用户退出登录。
WHOAMI     显示当前用户信息。
FORMAT     格式化文件卷。
REGISTER   用户注册。
DELETEACCOUNT 删除用户 (root用户下)
SU         改变用户。
CHGRP      改变用户所属组 (root用户下)
USERLIST   显示所有用户信息 (root用户下)

文件系统不存在，按任意键进行初始化...
  
```

图 39: 初始界面

随后提示需要进行登录，并输入用户名和密码，默认情况下存在两个用户：root 用户（拥有最高权限）、juju 用户（普通用户），此时使用 root 用户登录

```

用户未登录，请输入用户名和密码
用户名: root
密码: root
root>
root>
root>
root>
root>
root>
root>
root>dir
root>

```

Edit Time	Type	Size(Byte)	File/Directory Name
2021-06-09 21:00:04	<DIR>		.
2021-06-09 21:00:04	<DIR>		..
2021-06-09 21:00:04	<DIR>		bin
2021-06-09 21:00:04	<DIR>		etc
2021-06-09 21:00:04	<DIR>		home
2021-06-09 21:00:04	<DIR>		dev

图 40: 成功登录

对一些基本命令进行测试（详细测试将在下一部分进行展示）：



```

root>userlist
      UserName      UserId  UserGroupId
      root          0        1
      juju          1        2

root>register new1 new1
成功创建用户

root>register new1 new1
用户名已存在, 无法注册
【错误码】6

root>userlist
      UserName      UserId  UserGroupId
      root          0        1
      juju          1        2
      new1          2        2

root>register new2 new2
成功创建用户

root>register new3 new3
成功创建用户

root>register new4 new4
成功创建用户

root>register new5 new5
用户量已达上限, 无法继续注册
【错误码】6

root>userlist
      UserName      UserId  UserGroupId
      root          0        1
      juju          1        2
      new1          2        2
      new2          3        2
      new3          4        2
      new4          5        2

```

图 41: 用户相关命令

```

root>help
有关某个命令的详细信息, 请键入 HELP 命令名
HELP      帮助文档。
ATTRIB    显示或更改文件属性。
CD        显示当前目录的名称或将其更改。
DEL       删除至少一个文件。
DIR       显示一个目录中的文件和子目录。
EXIT      退出文件系统。
MKDIR     创建一个目录。
RMDIR     删除目录。
PRINT     打印文件内容。
WRITE     向文件中写入内容
OPEN      打开一个文件
CLOSE     关闭一个文件
CREATE    创建一个文件
OPENLIST  当前打开文件列表
FSEEK     更改一个文件的指针
LOGOUT    用户退出登录
WHOAMI    显示当前用户信息
FORMAT    格式化文件卷
REGISTER  用户注册 (root用户下)
DELETEACCOUNT 删除用户 (root用户下)
SU        改变用户
CHGRP     改变用户所属组 (root用户下)
USERLIST  显示所有用户信息 (root用户下)

```

图 42: help 命令

```

root>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:05:57 <DIR>      .
2021-06-09 21:05:57 <DIR>      ..
2021-06-09 21:05:57 <DIR>      bin
2021-06-09 21:05:57 <DIR>      etc
2021-06-09 21:05:57 <DIR>      home
2021-06-09 21:05:57 <DIR>      dev

root>cd home
root\home>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:05:57 <DIR>      .
2021-06-09 21:05:57 <DIR>      ..
2021-06-09 21:05:57 <DIR>      texts
2021-06-09 21:05:57 <DIR>      reports
2021-06-09 21:05:57 <DIR>      photos

root\home>open texts
文件名不合法, 当前目录不存在名为texts的文件
【错误码】2

root\home>cd texts
root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:05:57 <DIR>      .
2021-06-09 21:05:57 <DIR>      ..
2021-06-09 21:05:57      0      test.txt

root\home\texts>open test.txt
已成功打开文件test.txt

root\home\texts>write test.txt -s aaaaaaaaaaaaaa
成功写入文件
    
```

图 43: 文件及目录相关测试

## 5.2 命令测试

### 5.2.1 课设要求测试

- 用 `mkdir` 命令创建子目录，建立如图所示的目录结构；

```

root>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:31:22 <DIR>      .
2021-06-09 21:31:22 <DIR>      ..
2021-06-09 21:31:22 <DIR>      bin
2021-06-09 21:31:22 <DIR>      etc
2021-06-09 21:31:22 <DIR>      home
2021-06-09 21:31:22 <DIR>      dev
    
```

图 44: 目录结构-1

```

root\home>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:31:22 <DIR>      .
2021-06-09 21:31:22 <DIR>      ..
2021-06-09 21:31:22 <DIR>      texts
2021-06-09 21:31:22 <DIR>      reports
2021-06-09 21:31:22 <DIR>      photos
    
```

图 45: 目录结构-2

- 把你的课设报告，关于课程设计报告的 `ReadMe.txt` 和一张图片存进这个文件系统，分别放在 `/home/texts`，`/home/reports` 和 `/home/photos` 文件夹；

使用如下的命令将 `ReadMe.txt` 放入 `/home/texts`

```

1 cd home/texts
2 create readme.txt
3 open readme.txt
    
```

```
root\home\reports>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:31:22 <DIR>      .
2021-06-09 21:31:22 <DIR>      ..
2021-06-09 21:48:37      252      report.pdf
```

图 48: 放入 report.pdf

```
4 write readme.txt -f ReadMe.txt
5 close readme.txt
```

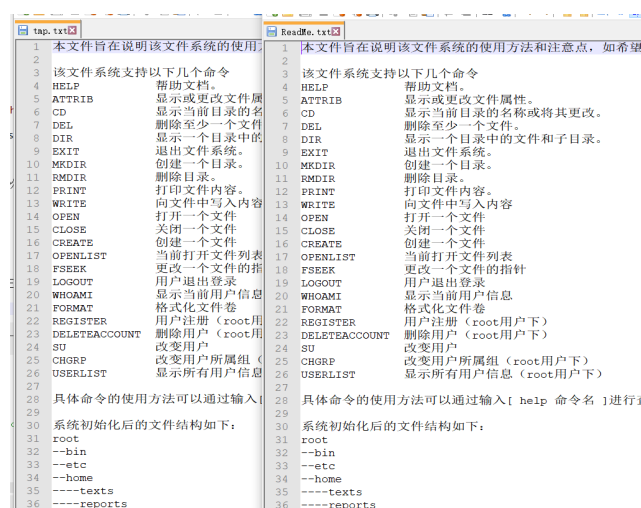
```
root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 21:31:22 <DIR>      .
2021-06-09 21:31:22 <DIR>      ..
2021-06-09 21:32:55      1872      readme.txt
```

图 46: 将 ReadMe.txt 放入/home/texts

使用如下的命令将 readme.txt 中的内容再次写出：

```
1 cd home/texts
2 open readme.txt
3 print readme.txt -p tmp.txt
4 fseek readme.txt 0
5 close readme.txt
```

比对两个文件，完全一致，证明了程序的正确性。



```
1 本文件旨在说明该文件系统的使用方法
2
3 该文件系统支持以下几个命令
4 HELP 帮助文档。
5 ATTRIB 显示或更改文件属性
6 CD 显示当前目录的名称或将其更改。
7 DEL 删除至少一个文件。
8 DIR 显示一个目录中的文件和子目录。
9 EXIT 退出文件系统。
10 MKDIR 创建一个目录。
11 RMDIR 删除目录。
12 PRINT 打印文件内容。
13 WRITE 向文件中写入内容
14 OPEN 打开一个文件
15 CLOSE 关闭一个文件
16 CREATE 创建一个文件
17 OPENLIST 当前打开文件列表
18 FSEEK 更改一个文件的指针
19 LOGOUT 用户退出登录
20 WHOAMI 显示当前用户信息
21 FORMAT 格式化文件卷
22 REGISTER 用户注册 (root用户下)
23 DELETEACCOUNT 删除用户 (root用户下)
24 SU 改变用户
25 CHGRP 改变用户所属组 (root用户下)
26 USERLIST 显示所有用户信息
27
28 具体命令的使用方法可以通过输入[ help 命令名 ]进行查
29
30 系统初始化后的文件结构如下:
31 root
32 --bin
33 --etc
34 --home
35 ----texts
36 ----reports
```

图 47: 文件比对

进一步进行其他两个文件的放入

- 新建文件 `/test/Jerry`，打开该文件，任意写入 800 个字节；

[illegible]

图 49: 创建文件并写入 800 字节

由图49，成功创建文件并进行了写入。

- 将文件读写指针定位到第 500 字节，读出 500 个字节并写回到文件

```
root@test>dir
      Edit Time           Type           Size(Byte)           File/Directory Name
      2021-06-09 21:54:59      <DIR>
      2021-06-09 21:31:22      <DIR>           ..
      2021-06-09 21:57:29              800           jerry

root@test>fseek jerry 500
已将jerry文件指针定位到500

root@test>print jerry -l 500 -p jerryfile.txt
已将jerry文件内容写入jerryfile.txt
```

图 50: 定位读写指针并写回文件

可以看到与预期相同，写入了 300 个字节

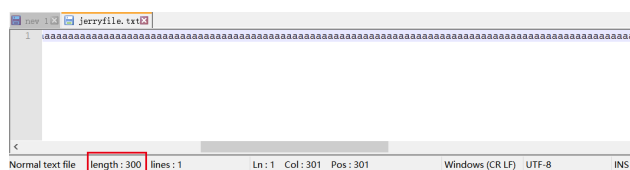


图 51: jerryfile.txt

再将其写回到文件

```
root@test>write_jerry -f jerryfile.txt
已将jerryfile.txt文件内容写入jerry

root@test>dir
```

	Edit Time	Type	Size(Byte)	File/Directory Name
2021-06-09 21:54:59		<DIR>		..
2021-06-09 21:31:22		<DIR>		..
2021-06-09 22:03:37			1100	jerry

图 52: 写回文件

可以看到与预期相同，jerry 目前共由 1100 个字节

## 5.2.2 命令逐条测试

对命令进行逐一测试：

- HELP——帮助文档。

```

root>help
有关某个命令的详细信息，请键入 HELP 命令名
HELP      帮助文档。
ATTRIB    显示或更改文件属性。
CD        显示当前目录的名称或将其更改。
DEL       删除至少一个文件。
DIR       显示一个目录中的文件和子目录。
EXIT      退出文件系统。
MKDIR     创建一个目录。
RMDIR     删除目录。
PRINT     打印文件内容。
WRITE     向文件中写入内容。
OPEN      打开一个文件。
CLOSE     关闭一个文件。
CREATE    创建一个文件。
OPENLIST  当前打开文件列表。
FSEEK     更改一个文件的指针。
LOGOUT    用户退出登录。
WINFO     显示当前用户信息。
FORMAT    格式化文件卷。
REGISTER  用户注册 (root用户下)
DELETEACCOUNT 删除用户 (root用户下)
CHGRP     改变用户所属组 (root用户下)
USERLIST  显示所有用户信息 (root用户下)

root>help attrib
更改文件的读与属性
ATTRIB [+R | -R] [+W | -W] [+E | -E] [O | G | E] [path][filename]

+ 设置属性。
- 清除属性。
    
```

图 53: help 命令测试

- ATTRIB——显示或更改文件属性。

```

root@home/texts>ls -l
Edit Time      Type      Size(Byte)  File/Directory Name  Owner Id  Owner Group  Inode Id  O[RW]G[RW]E[RW]E
2021-06-09 22:28:02 -DIR-      0              .                  0          1          6         111 111 111
2021-06-09 22:28:02 -DIR-      0              ..                 0          1          4         111 111 111
2021-06-09 22:28:02 -FILE-     0              test.txt          0          1          9         111 111 111

root@home/texts>attrib -R E test.txt
root@home/texts>
    
```

图 54: 更改文件读写属性

- CD——显示当前目录的名称或将其更改。

```

root@home/texts>cd ..
root@home>
    
```

图 55: 更改目录

- DEL——删除至少一个文件。

```

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:28:02      0      test.txt
2021-06-09 22:33:52      0      a.txt
2021-06-09 22:33:55      0      b.txt

root\home\texts>del test.txt a.txt b.txt
成功删除文件test.txt
成功删除文件a.txt
成功删除文件b.txt

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..

```

图 56: 删除文件

- DIR———显示一个目录中的文件和子目录。

```

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:28:02      0      test.txt
2021-06-09 22:33:52      0      a.txt
2021-06-09 22:33:55      0      b.txt

```

图 57: 显示目录中内容

- MKDIR———创建一个目录。

```

root\home\texts>mkdir a
成功创建目录a

root\home\texts>mkdir b
成功创建目录b

root\home\texts>mkdir c
成功创建目录c

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:38:30 <DIR>      a
2021-06-09 22:38:32 <DIR>      b
2021-06-09 22:38:34 <DIR>      c

```

图 58: 创建目录

- RMDIR———删除目录。

```

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:38:30 <DIR>      a
2021-06-09 22:38:32 <DIR>      b
2021-06-09 22:38:34 <DIR>      c

root\home\texts>rmdir a
成功删除目录a

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:38:32 <DIR>      b
2021-06-09 22:38:34 <DIR>      c

```

图 59: 删除目录

- PRINT———打印文件内容。

```
root\home\texts>print test.txt
aaaaaaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

图 60: 打印文件内容

- WRITE——向文件中写入内容

```
root\home\texts>write test.txt -s aaaaaaaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
成功写入文件
```

图 61: 向文件写入

- OPEN——打开一个文件

```
root\home\texts>open test.txt
已成功打开文件test.txt
```

图 62: 打开文件

- CLOSE——关闭一个文件

```
root\home\texts>close test.txt
已成功关闭文件test.txt
```

图 63: 关闭文件

- CREATE——创建一个文件

```
root\home\texts>create test.txt
成功创建文件test.txt

root\home\texts>dir
Edit Time      Type      Size(Byte)  File/Directory Name
2021-06-09 22:28:02 <DIR>      .
2021-06-09 22:28:02 <DIR>      ..
2021-06-09 22:40:13 <DIR>      0
2021-06-09 22:38:32 <DIR>      test.txt
2021-06-09 22:38:34 <DIR>      b
                                     c

root\home\texts>
```

图 64: 创建一个文件

- OPENLIST——当前打开文件列表

```
root\home\texts>openlist
当前打开的文件有:
root\home\texts\test.txt
```

图 65: 打开文件列表

- FSEEK——更改一个文件的指针

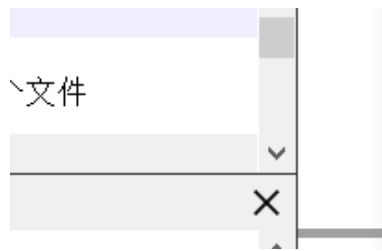


图 66: 更改文件定位指针

- LOGOUT——用户退出登录

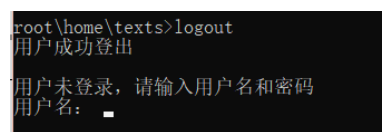


图 67: 用户登出

- WHOAMI——显示当前用户信息

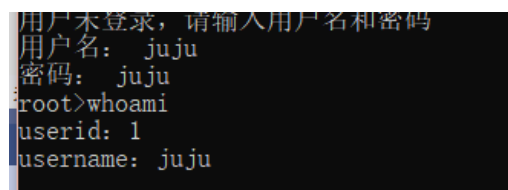


图 68: 显示当前用户信息

- REGISTER——用户注册（root 用户下）

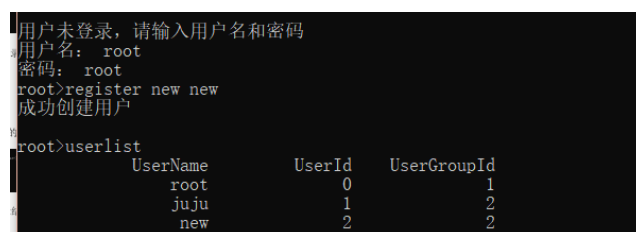


图 69: 用户注册

- DELETEACCOUNT——删除用户（root 用户下）



```

root>userlist
      UserName      UserId      UserGroupId
      root          0          1
      juju          1          2
      new           2          2

root>deleteaccount juju
成功删除用户

root>userlist
      UserName      UserId      UserGroupId
      root          0          1
      new           2          2

```

图 70: 删除用户

- SU———-改变用户

```

root>whoami
userid: 2
username: new

root>su root root
成功更换用户

root>whoami
userid: 0
username: root

```

图 71: 改变用户

- CHGRP———-改变用户所属组（root 用户下）

```

用户名:
root
密码: root
root>userlist
      UserName      UserId      UserGroupId
      root          0          1
      new           2          2

root>chgrp new 3
成功更改用户new到3组

root>userlist
      UserName      UserId      UserGroupId
      root          0          1
      new           2          3

```

图 72: 改变用户所属组

- USERLIST——-显示所有用户信息（root 用户下）

```

root>userlist
      UserName      UserId      UserGroupId
      root          0          1
      new           2          3

```

图 73: 显示用户信息

### 5.2.3 其他测试

- 多用户读写测试

– 首先，用户 juju 在文件 test.txt 中写入 juju

```
root\home\texts>whoami
userid: 1
username: juju

root\home\texts>dir
               Edit Time           Type      Size(Byte)   File/Directory Name
               ~~~~~
               2021-06-09 22:57:04   <DIR>      <DIR>         .
               2021-06-09 22:57:04   <DIR>      <DIR>         ..
               2021-06-09 22:58:08                   0          test.txt

root\home\texts>open test.txt
已成功打开文件test.txt

root\home\texts>write test.txt -s juju
成功写入文件

root\home\texts>fseek test.txt 0
已将test.txt文件指针定位到0

root\home\texts>print test.txt
juju
```

图 74: 多用户读写测试-1

– 随后用户 root 打开同一个文件 test.txt，在其中写入 root

```
root\home\texts>dir
               Edit Time           Type      Size(Byte)   File/Directory Name
               ~~~~~
               2021-06-09 22:57:04   <DIR>      <DIR>         .
               2021-06-09 22:57:04   <DIR>      <DIR>         ..
               2021-06-09 22:58:33                   4          test.txt

root\home\texts>open test.txt
已成功打开文件test.txt

root\home\texts>write test.txt -s root
成功写入文件

root\home\texts>fseek test.txt 0
已将test.txt文件指针定位到0

root\home\texts>print test.txt
root

root\home\texts>whoami
userid: 0
username: root
```

图 75: 多用户读写测试-2

可以看到，root 用户在刚进入文件系统时可以看到 juju 写入的 4 个字节，随后 root 用户写入 root 将 juju 覆盖

– 此时 juju 再次进行文件的读，读到的是 root 用户写入的 root

```
root\home\texts>print test.txt
juju

root\home\texts>fseek test.txt 0
已将test.txt文件指针定位到0

root\home\texts>print test.txt
root

root\home\texts>whoami
userid: 1
username: juju
```

图 76: 多用户读写测试-3

- juju 继续向文件中写入 juju

```
root\home\texts>print test.txt
root

root\home\texts>whoami
userid: 1
username: juju

root\home\texts>write test.txt -s juju
成功写入文件

root\home\texts>fseek test.txt 0
已将test.txt文件指针定位到0

root\home\texts>print test.txt
rootjuju
```

图 77: 多用户读写测试-4

- 此时 root 用户再次读

```
root\home\texts>print test.txt
root

root\home\texts>whoami
userid: 0
username: root

root\home\texts>fseek test.txt 0
已将test.txt文件指针定位到0

root\home\texts>print test.txt
rootjuju
```

图 78: 多用户读写测试-5

由图可知，juju 写入的 juju 四个字符已被 root 用户读到，因此可以得出结论，该文件系统支持多个用户同时读写。

- 大文件读写

## – 从网上下载简爱英文版

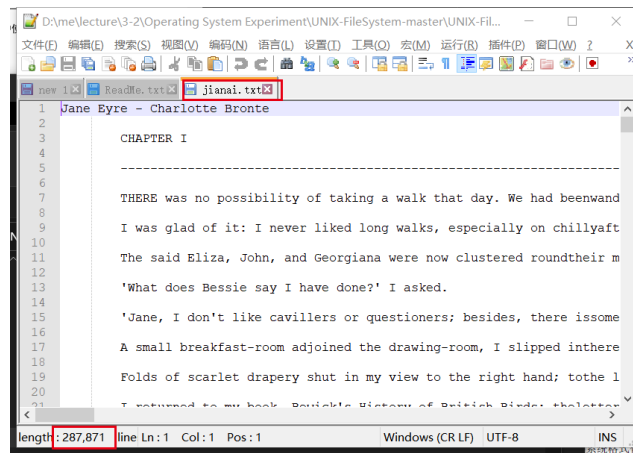


图 79: txt 格式简爱

可以看到其共有 287871 字节

## – 将其写入文件



图 80: 大文件测试-1

## – 将其读出到屏幕

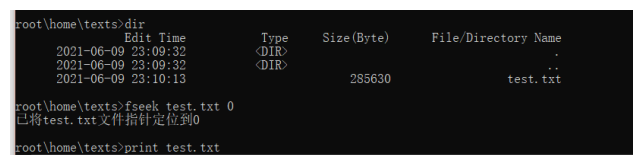


图 81: 大文件测试-2

## – 将输出内容与文件进行比对，两者完全相同

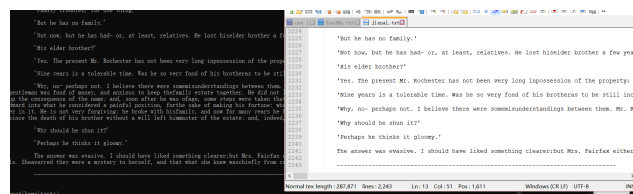


图 82: 大文件测试-3

因此可以得出结论，该系统可以进行大文件的读入

### • 读写权限测试

- root 用户创建文件，并更改其权限为除自己与同组用户外都不可读写

```

用户名: root
密码: root
root>
root>
root>create file.txt
成功创建文件file.txt

root>attrib -w E file.txt
已成功更改文件读写属性

root>attrib -r E file.txt
已成功更改文件读写属性

root>attrib -e E file.txt
已成功更改文件读写属性
    
```

图 83: 读写权限测试-1

```

root:dir /q
Edit Time      Type      Size(Byte)  File/Directory Name  Owner Id  Owner Group  Inode Id  01RWEGIRWEIEIRWEI
2021-06-09 23:17:43 (DIR)      0            .                      0         1          0         111 111 111
2021-06-09 23:17:43 (DIR)      0            bin                    0         1          2         111 111 111
2021-06-09 23:17:43 (DIR)      0            etc                    0         1          3         111 111 111
2021-06-09 23:17:43 (DIR)      0            home                   0         1          4         111 111 111
2021-06-09 23:17:43 (DIR)      0            dev                    0         1          5         111 111 111
2021-06-09 23:18:09 (DIR)      0            file.txt               0         1         10         111 111 000
    
```

图 84: 读写权限测试-2

- 切换到 juju 用户（与 root 非同组）

```

root>su juju juju
成功更换用户

root>open file.txt
已成功打开文件file.txt

root>write file.txt -s aaa
当前用户没有权限对文件进行写操作
【错误码】3
    
```

图 85: 读写权限测试-3

可以看到，系统提示 juju 无权进行写操作

## 5.3 结果分析

由 5.1 及 5.2 可知，程序具有正确的运行结果，并在用户命令输入不当时可以进行正确的报错，同时可以对用户输入命令给予良好的引导，整个程序完整清晰，具有友好性和正确性

## 6 用户使用说明

项目结构：

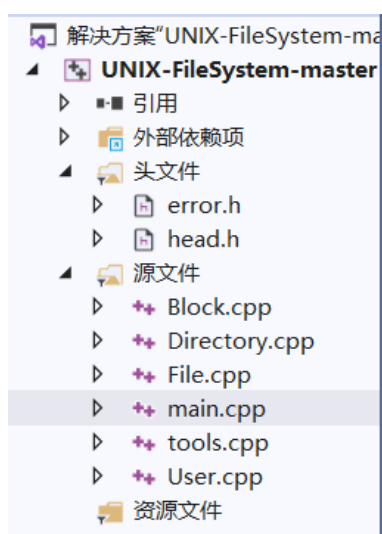


图 86: 项目结构

运行方法：

在 Windows 下直接点击生成的 exe 文件执行，linux 下可以将源程序放在集成环境下编译，也可利 GNU 编译工具，通过写好的 Makefile 进行编译。运行界面为控制台的命令行方式，命令较为简单，通俗易懂，初始界面如下：

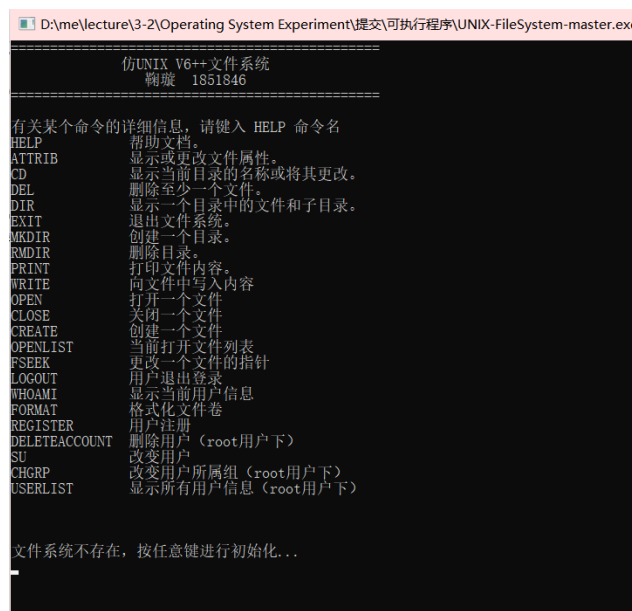


图 87: 初始界面

具体命令的使用方法可以通过输入 [ help 命令名] 进行查看

系统初始化后的文件结构如下: root -bin -etc -home —-texts —-reports —-photos -dev

最初进入系统时包含两个用户: root 用户和 juju 用户, 其中 root 用户为最高权限用户, 可以进行注册用户、删除用户、改变用户所属组、显示用户信息等操作, juju 为普通用户, root 用户密码为 root, juju 用户密码为 juju

该文件系统在节点数量等上具有一定限制, 用户的最大数量为 6, 文件及目录个数最多为 256, 一个目录下的子目录与文件数之和最多为 12, 文件名称的最大长度为 32, 用户名称的最大长度为 16, 用户密码的最大长度为 32

程序正常退出使用 exit 命令, 格式化使用 format 命令。

## 参考文献

- [1] 尤晋元,UNIX 操作系统教程 [M] 西安: 电子科技大学出版社,1985 年 6 月
- [2] Unix v6++ 源码
- [3] 同济大学操作系统课程电子教案