****

操作系统课设报告

|  |  |
| --- | --- |
| 题 目： | Unix文件系统模拟 |
| 学生姓名： | 钟伍全 |
| 指导教师： | 刘嫔 |
| 学 院： | 信息科学与工程学院 |
| 专业班级： | 计科1607 |

本科生院制

2018年7月

目录

[一． 问题与任务描述 3](#_Toc519358071)

[二． 设计的基本概念与原理 3](#_Toc519358072)

[三． 总体设计 4](#_Toc519358073)

[系统总体流程图 4](#_Toc519358074)

[各模块功能说明 5](#_Toc519358075)

[该文件系统的数据结构和申明函数 6](#_Toc519358076)

[主要函数列表 8](#_Toc519358077)

[函数之间的关系 9](#_Toc519358078)

[四． 详细设计 9](#_Toc519358079)

[五． 完成情况 12](#_Toc519358080)

[1. 完成内容 12](#_Toc519358081)

[2. 完成过程 13](#_Toc519358082)

[六． 使用说明 14](#_Toc519358083)

[七． 总结 19](#_Toc519358084)

[1、 过程中遇到的问题 19](#_Toc519358085)

[2、 个人总结与感触 22](#_Toc519358086)

[参考文献： 22](#_Toc519358087)

[附录： 22](#_Toc519358088)

1. 问题与任务描述

模拟UNIX文件系统设计和实现一个多用户多目录的文件系统, 要求如下：

1. 可以模拟多用户，多个用户可以操作不同的文件。
2. 可以实现多目录管理，包括目录的增删改查。
3. 可以实现文件的增删改查、属性管理。
4. 考虑用户、目录、文件的权限管理。
5. 完成文件存储空间管理，包括空闲块管理、磁盘Inode节点、内存Inode节点等内容。
6. 编写测试代码，测试对文件的相关操作：建立，读写等
7. 设计的基本概念与原理

本次课程设计要求我们模拟UNIX文件系统功能设计一个虚拟的文件系统，根据UNIX文件系统的特点，我们组的设计思想是：

首先，申请1M的内存空间来虚拟文件系统，将其以512B每块划分为2048块，采用位示图管理文件系统的方法，利用其原理将第1块作为位图区(512B)，对i节点区和数据块区分别建立位图；每个i节点占16B，每个块有512/16=32个i节点，用2、3共两块作为i节点，总共有64个i节点，在位图区占用8B；其余2045块作为数据区，在位图区占用接近256B(2045位)。

文件系统存储区分配图如下：

|  |  |  |
| --- | --- | --- |
| 位图区 | i节点区 | 数据区 |

位图： 用0表示未使用，1表示使用；

i节点： 文件控制块(FCB)，描述文件信息的一个数据结构；

数据区：存放用户数据，包括目录文件。

位示图用以反映整个存储空间的分配情况，由若干字节构成，每个字节中的每一位对应文件存储器中的一块，“1”状态表示相应块已占用，“0”状态表示该块为空闲。存储块分配时，只要把找到的空闲块所对应的位由 "0" 改为 "1"， 而在释放时，只要把被释放的物理块所对应的位由 "1" 改为 " 0 " 即可。分配和释放都可以在内存的位示图上完成，而且速度较快。磁盘 i 节点利用在内存中生成链表或者数组的方法来生成，并且限制磁盘节点数的最大值。同样内存 i 节点利用链表的方法在内存中生成。同时在往文件中写或者读的时候我们都是对内存的内容进行读写。

在该文件系统中，规定一个文件最多只能占用2个数据块（1024B），这两个数据块可以是不连续的，其块号记录在一个索引块中，该索引块称为inode结构。

读入一个文件时，先根据目录找到相应的i节点号，将i节点读入主存i节点，建立打开文件表指向该主存i节点，再将文件内容读入主存数据区。

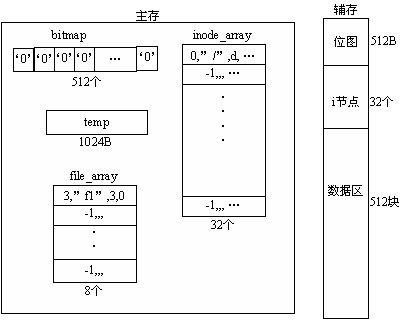


图1：系统总体框架

1. 总体设计

系统总体流程图

开始

文件系统是否建立？

在内存中申请1M内存空间

将磁盘上的文件系统读入内存

Y

N

用户登录模块，输入用户名和密码，可以

在程序中设定，不用保存在文件系统中

用户名、密码对吗？

执行init()，进行初始化

操作，进入用户子目录

执行command()，对用户输入的命令进行解析和

执行。若数据被修改或新建立，则保存主存i节

点和数据块的内容到内存的文件系统，并写回磁

盘，保存修改。若输入错误命令则显示出错信息。

退出用户(logout)吗？

用户输入quit吗？

结束

执行format()，

建立文件系统

N

N

N

Y

Y

Y

图2：系统总体流程图

各模块功能说明

1　format()模块

功能：格式化文件系统，即初始化文件系统，相当于硬盘的格式化。将其中原有的用户及用户下的文件系统全部还原初始状态，即没有任何用户和目录、文件，也就是按设计的文件系统格式重建新的文件系统。

2 get\_blknum()和release\_blk()模块

功能：实现i节点的分配和回收；

算法：当一个新的文件被建立时，在给该文件分配磁盘存储区之前，应为该文件分配存放该文件的说明信息的磁盘i节点。反之，当从文件系统中删除某个文件时，则要首先删除它的i节点项。

3　init()模块

功能：进入文件系统

算法：初始化用户打开文件系表。在内存中申请一个虚拟存储空间，此空间必须大于或等于可格式化时的空间。将文件系统文件读入虚拟磁盘。

4 quit()模块

功能：退出文件系统函数quit()

算法：将虚拟磁盘内容保存到磁盘上。释放虚拟磁盘所占据的空间。撤消用用户打开文件表。

5 touch()模块

功能：创建文件

用法：touch filename或touch Address/filename

算法：分配一空目录项，分配磁盘块。可用位示图。填写该空目录项。如文件长度（0），文件名，类型等。分配一个用户打开文件表项，并填写相关目录，读写指针=0。

6 open()模块

功能：打开文件open()

用法：open(3)

算法：if（该文件已打开or该文件不存在）报错(出错信息)。分配一个用户打开文件表项。

7 close()模块

功能：关闭文件close()

用法：close filename

算法：释放该文件的用户占据的内存空间。清空该文件的用户打开文件表目。

8 vi()模块

功能：写文件vi()

用法：vi filename，需要先打开文件。

算法：指定写入内容长度。buf。修改打开文件表读/写指针。修改目录项中的文件长度。

9 cat()模块

功能：读文件

用法：cat filename。

算法：文件信息.修改读/写指针。

10 delete()模块

功能：删除文件delete()

用法：delete filename

算法：若文件不存在，报错。回收磁盘空间。回收目录项。

该文件系统的数据结构和申明函数

// 用户(20B)

typedef struct

{

char user\_name[10]; // 用户名

char password[10]; // 密码

} User;

// i节点(32B)

typedef struct

{

short inum; // 文件i节点号

char file\_name[10]; // 文件名

char type; // 文件类型

char user\_name[10]; // 文件所有者

short iparent; // 父目录的i节点号

short length; // 文件长度

short address[2]; // 存放文件的地址

} Inode;

// 打开文件表(16B)

typedef struct

{

short inum; // i节点号

char file\_name[10]; // 文件名

short mode; // 读写模式(1:read, 2:write,

// 3:read and write)

short offset; // 偏移量

} File\_table;

// 申明函数

void login(void);

void init(void);

int analyse();

void save\_inode(int);

int get\_blknum(void);

void read\_blk(int);

void write\_blk(int);

void release\_blk(int);

void pathset();

void delet(int innum);

int check(int i);

// 用户命令处理函数

void help(void);

void cd(void);

void dir(void);

void mkdir(void);

void touch(void);

void open(int, string);

void cat(void);

void vi(void);

void close(string);

void su();

void del(void);

void logout(void);

void command(void);

void rd();

void quit();

void df();

主要函数列表

|  |  |  |
| --- | --- | --- |
| 函数原型 | 功能 | 入口、出口参数说明 |
| void login(void) | 用户登陆 |  |
| void init(void)) | 程序初始化 |  |
| int analyse(char \*) | 输入参数分析 | 命令行字符串 |
| void save\_inode(int) | 保存i节点到虚拟磁盘文件 | i节点号 |
| int get\_blknum(void) | 分配一个磁盘块 |  |
| void read\_blk(int) | 读磁盘快内容 | i节点号 |
| void write\_blk(int) | 写磁盘块 | i节点号 |
| void release\_blk(int) | 释放一个磁盘块 | i节点号 |
| void pathset(); | 打印当前路径 |  |
| int check(int); | 检查用户权限 | i节点号 |
| void help(void) | 命令提示帮助函数 |  |
| void dir(void) | 显示当前目录下所有文件和目录 |  |
| void mkdir(void) | 创建目录 |  |
| void touch(void) | 创建文件 |  |
| void open(void) | 打开文件 |  |
| void cat(void) | 读文件 |  |
| void vi(void) | 写文件 |  |
| void close(void) | 关闭文件 |  |
| void del(void) | 删除文件 |  |
| void logout(void) | 用户登出 |  |
| void su() | 切换当前用户 |  |
| void df() | 显示内存和空间使用量 |  |
| void command(void) | 命令管理函数 |  |
| void rmdir(void) | 删除目录 |  |
| void format(void); | 格式化虚拟磁盘 |  |

函数之间的关系

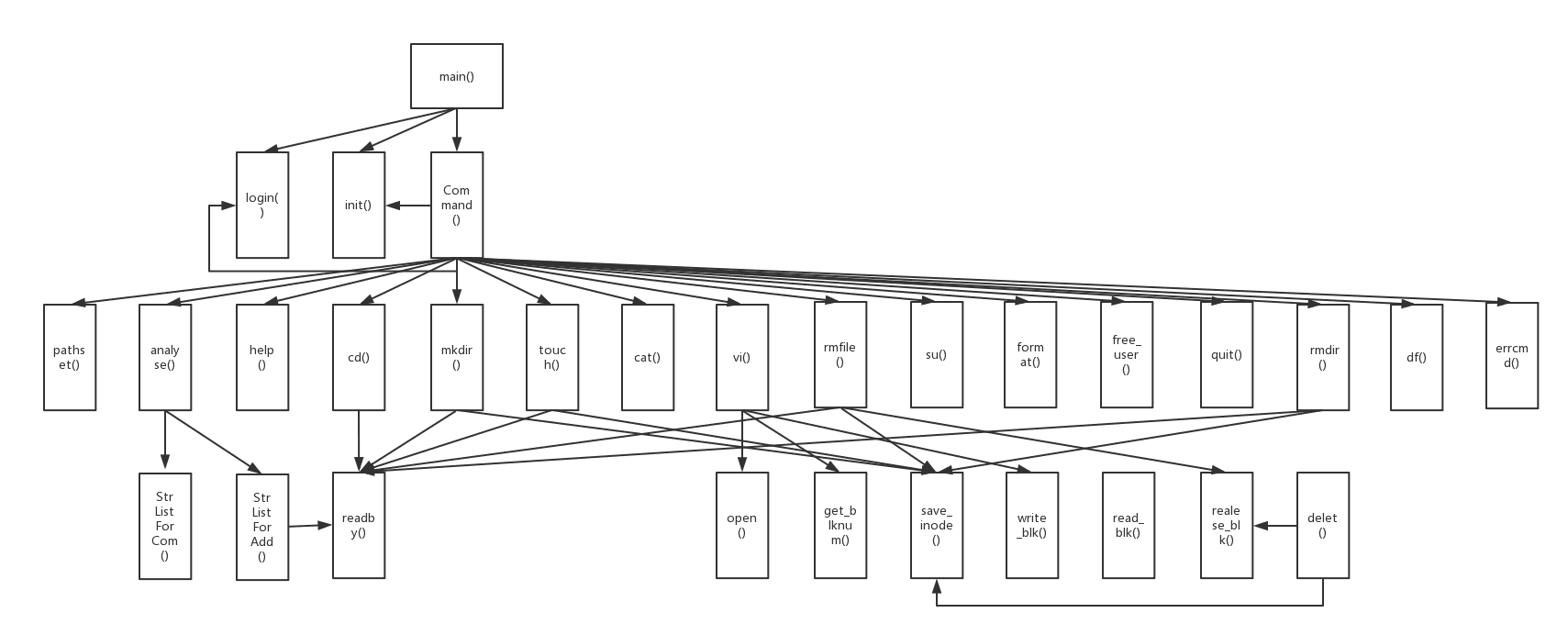


图3：函数关系图

1. 详细设计

因为整个系统是我们组三个人一起分工合作完成的，详细设计部分我就只写自己负责的部分。我们是按照函数文件系统的命令来划分的，涉及的底层函数我们在实现时都是自己完成，最后我们有功能差不多的底层函数，就将这几个函数融合到一起使用。

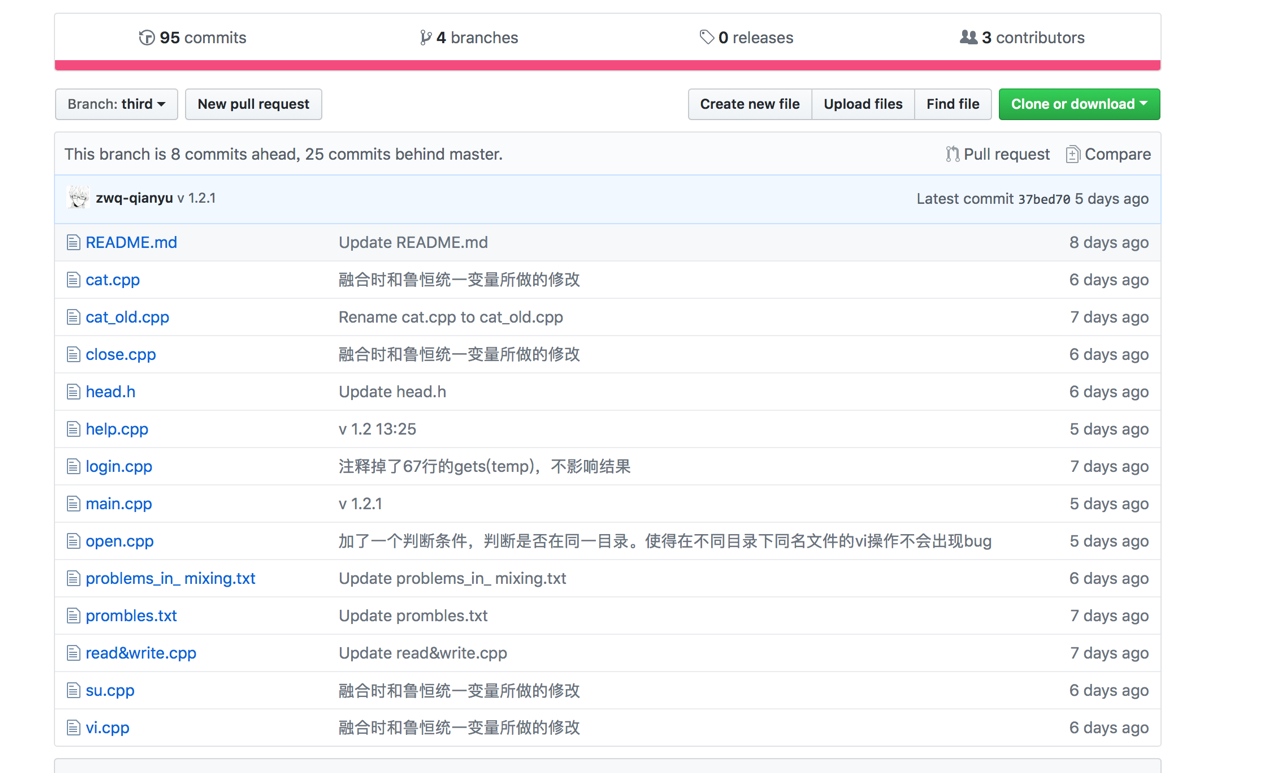
我们所有的功能函数包括：

|  |
| --- |
|  |
|  | cd --- change directory |
|  | clear --- clear the screen |
|  | ls --- show all the files and directories in particular directory |
|  | mkdir --- make a new directory |
|  | touch --- create a new file |
|  | cat --- open and read an exist file |
|  | vi --- open and write something to a particular file |
|  | rm --- delete a exist file |
|  | su --- switch current user |
|  | format --- format a exist filesystem |
|  | exit --- exit this system |
|  | rmdir --- delete a directory |
|  | df --- show the useage of storage  help --- show help menu |

我负责的函数包括：cat、vi、su、exit、help。在实现这几个函数功能时涉及到的函数有：

open()、close()、save\_inode()、get\_blknum()、read\_blk()、write\_blk()等。

我的GitHub上的third分支中的是我写的各个功能函数，具体代码可见：<https://github.com/zwq-qianyu/linux-system-simulation/tree/third>



下面是部分我负责实现的功能模块流程图：

是

否

是

否

是

否

遍历i节点区

要打开的文件是否存在

提示输入打开模式

将打开文件信息写入打开文件信息表file\_array[]

遍历打开文件表file\_array[]

文件是否已经打开

提示输入开始位置和读字节数

调用读块操作，读出块内字符

模式是否为可读

开始

结束

开始

结束

**打开文件open()**

**读文件cat()**

是

否

是

否

遍历打开文件表file\_array[]

文件是否已经打开

提示输入写入字节数

调用写块操作，写入块内字符

模式是否为可写

是

否

遍历打开文件表file\_array[]

文件是否已经打开

将打开文件表项标志位置-1

file\_array[i].inum = -1

开始

结束

**写文件vi()**

开始

结束

**关闭文件close()**

是

否

遍历位示图表查找空闲块bitmap[]

是否有空闲块

分配一块，将位示图位置置’1’。 bitmap[i] = '1'

返回找到的空闲块号

将位示图位置置’0’。 bitmap[i] = '0'

清空块内容

开始

结束

**分配磁盘块get\_blk()**

开始

结束

**释放磁盘块release\_blk(int)**

是

否

输出给定i节点对应的第1块内容

文件是否大于1块

输出给定i节点对应的第2块内容

是

否

将缓冲区buf内容写入第1块

buf内容是否大于512字节

将缓冲区buf的512字节后面的内容写入第2块

开始

结束

**写磁盘块write\_blk(int)**

开始

结束

**读磁盘块read\_blk(int)**

1. 完成情况

1. 完成内容

(1) 实现了多用户，多个用户可以操作不同的文件。

(2) 实现了多目录管理，包括目录的添加与删除。

(3) 实现了文件的添加、删除以及属性管理。

(4) 实现了对不同用户，目录和文件的权限管理，用户只能查看自己的目录和文件，禁止查看其它用户的目录和文件。

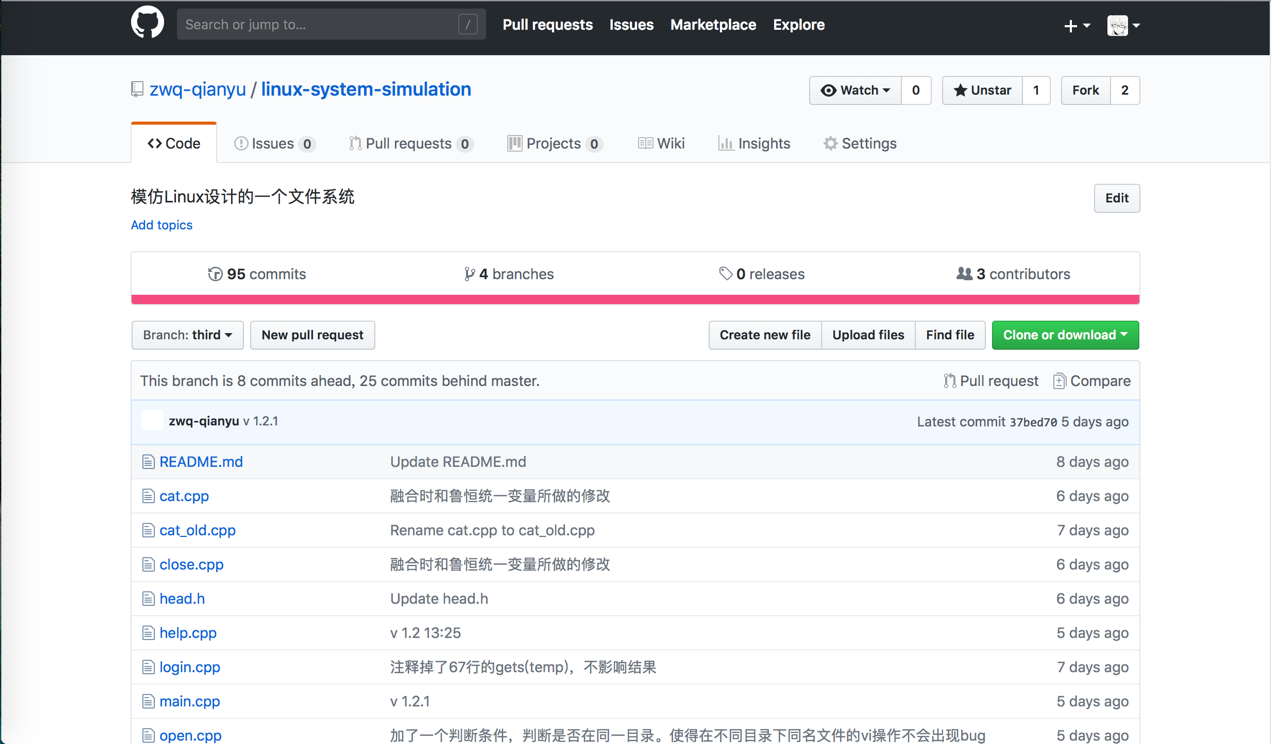
(5) 实现了文件存储空间管理，包括空闲块管理、磁盘Inode节点、内存Inode节点等内容。具体可以查看hd.dat文件。

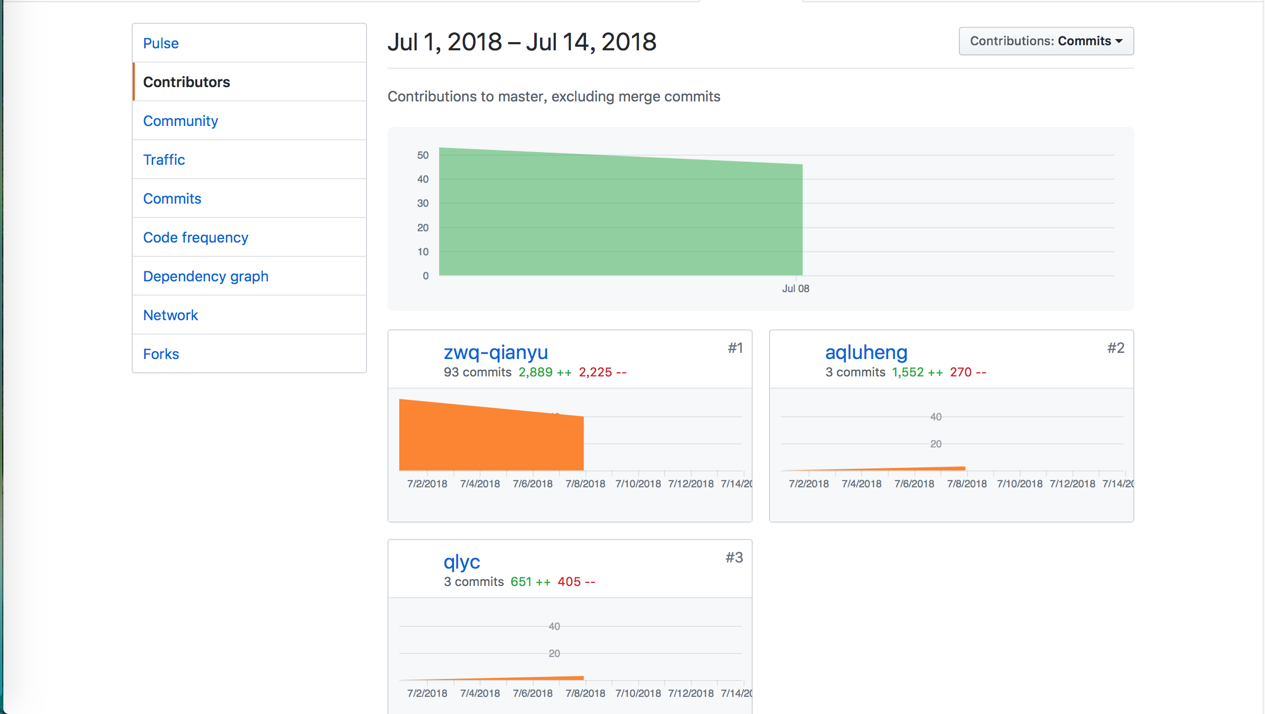
(6) 系统已经经过测试，目前未发现问题。

(7) 多级查看、多级cd等等都有实现。

2. 完成过程

整个系统由我、鲁恒、黄春伟3人合作完成，使用GitHub进行提交和版本控制，下面是GitHub上该项目的截图：





具体内容可到我的GitHub上查看：<https://github.com/zwq-qianyu/linux-system-simulation/tree/third>

1. 使用说明

直接打开可运行文件new\_main.exe；

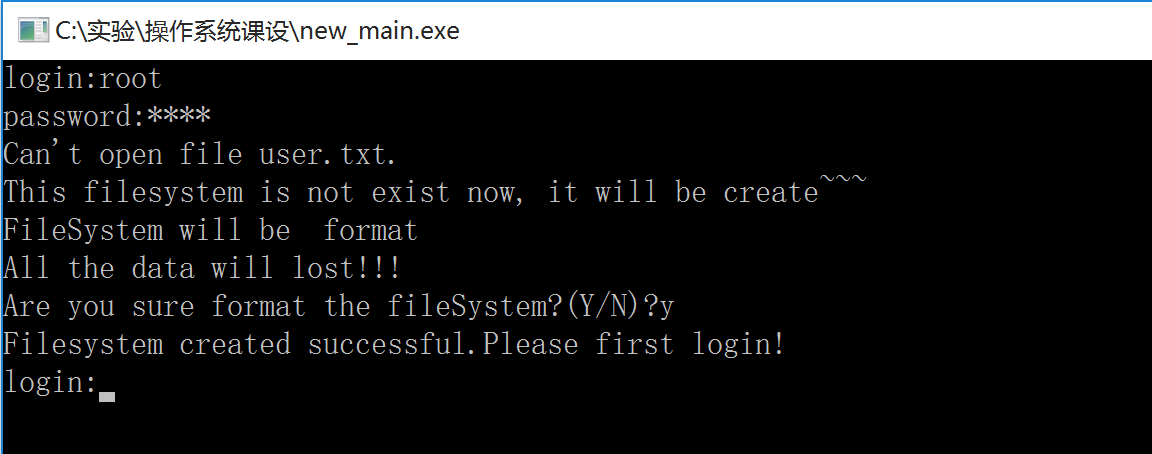
**或**用软件Dev-C++打开**third中的main**.cpp**、head.h**编译运行**；**

**或**用软件**vs2017**打开**master中的main**.cpp**、head.h**编译运行**；**

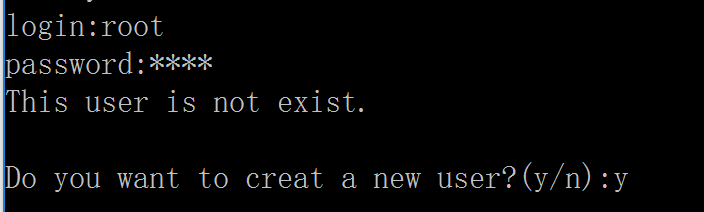
以上提到的 可执行软件与源代码下载地址：<https://github.com/zwq-qianyu/linux-system-simulation>

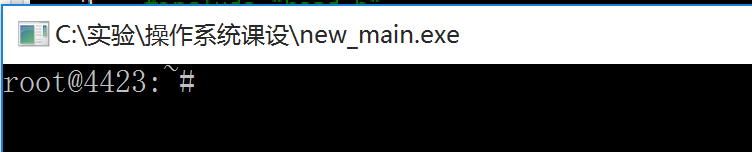
具体使用说明：

1. 初始化系统

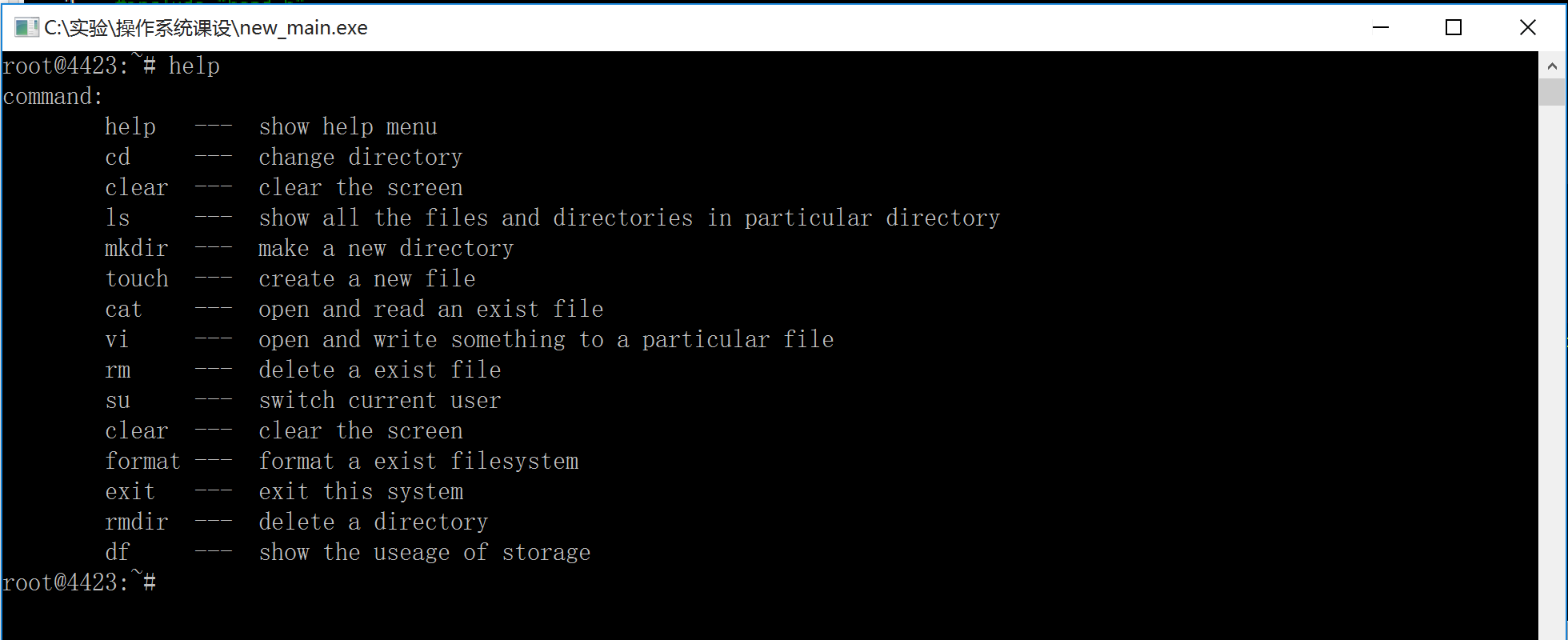


1. 创建新用户并登陆

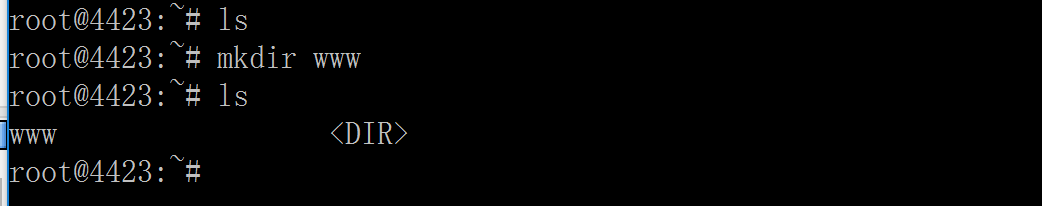




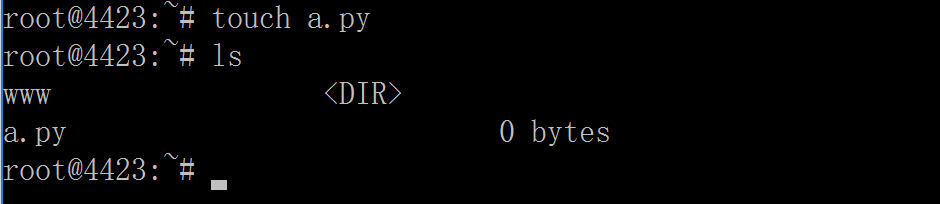
1. 使用help命令查看系统



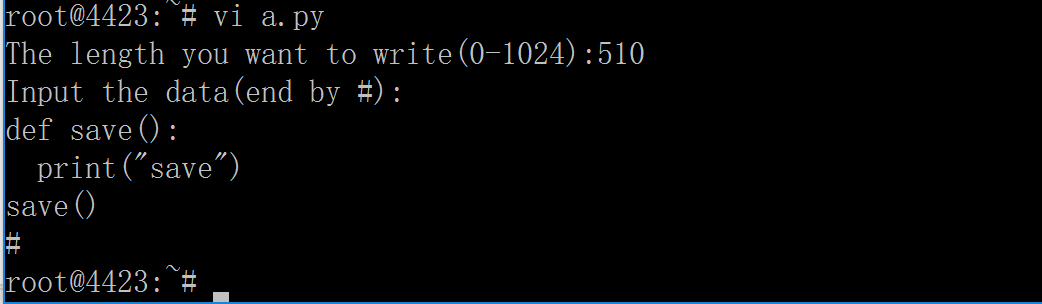
1. 使用mkdir命令创建目录



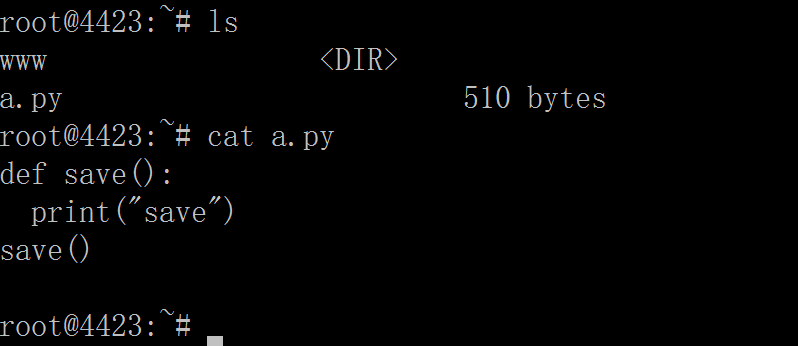
1. 使用touch创建文件



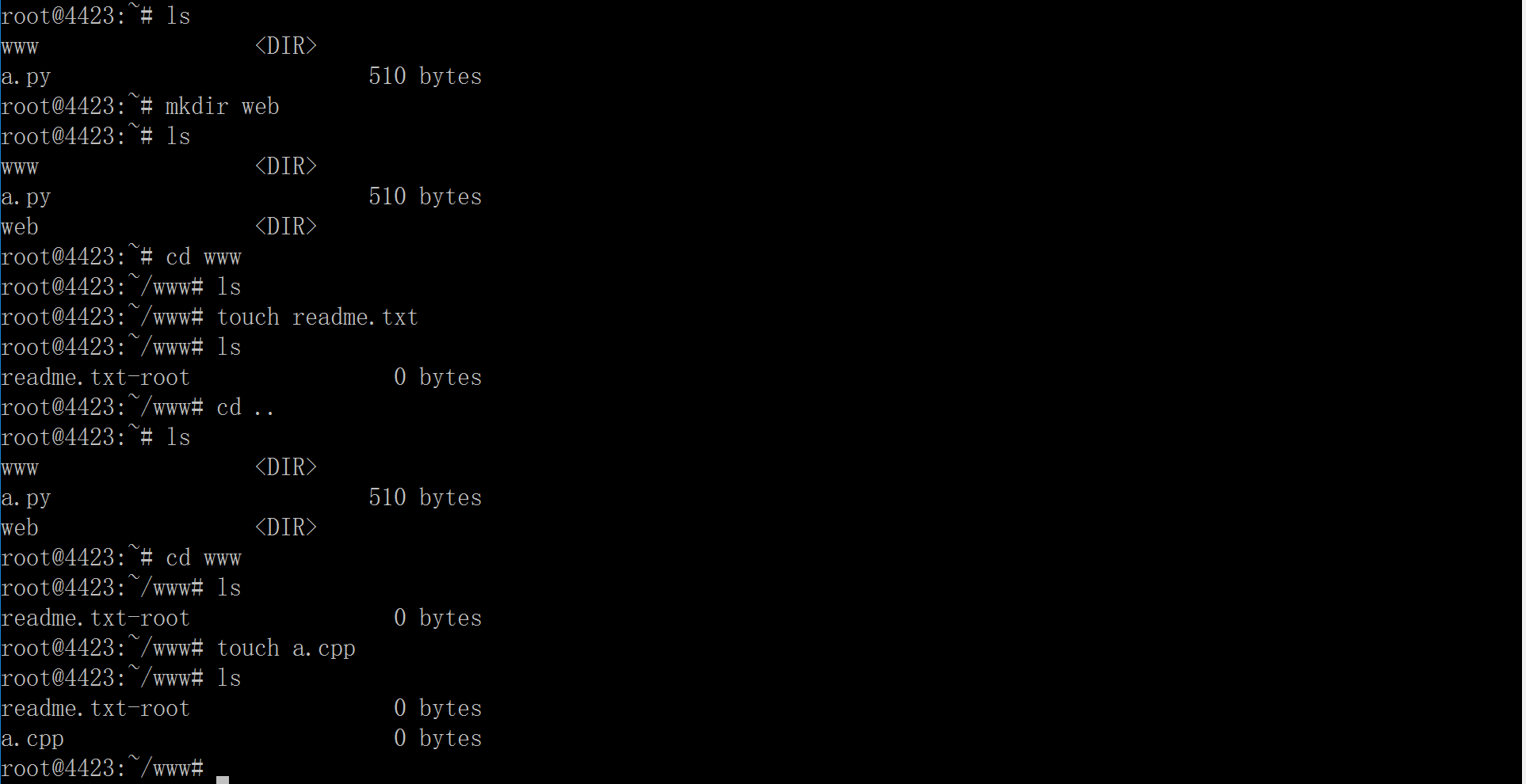
1. 使用vi写文件



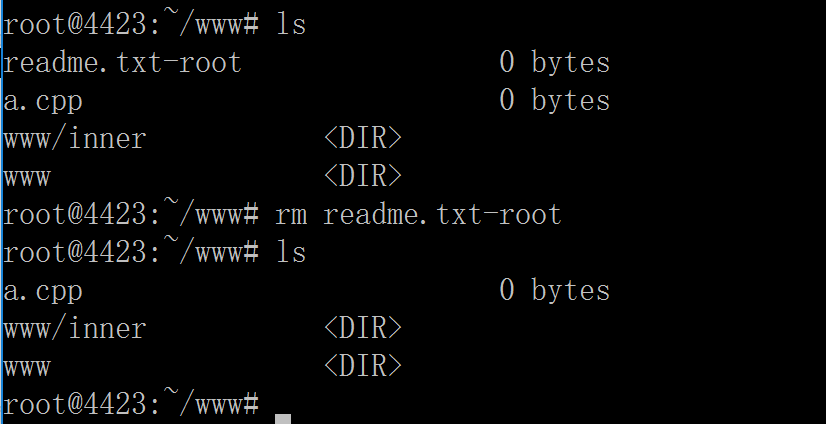
1. 使用cat查看文件



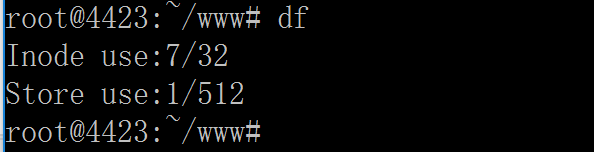
1. 使用cd进入目录



1. 使用rm删除文件



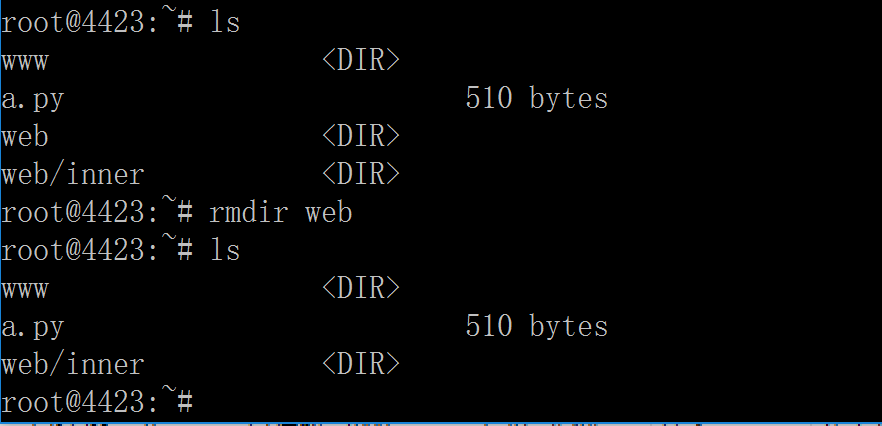
10. 使用df查看磁盘和内存使用情况



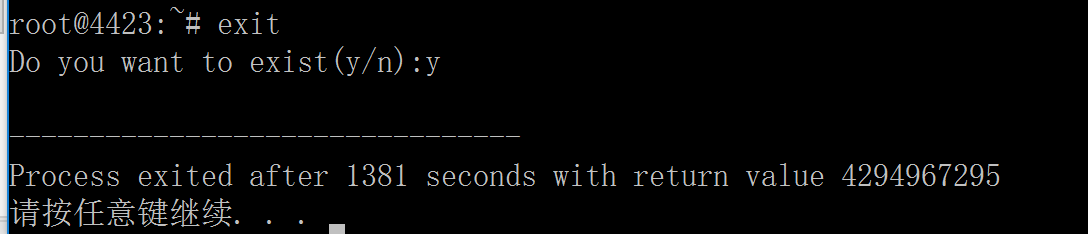
11. 使用ls查看当前目录下的文件与目录



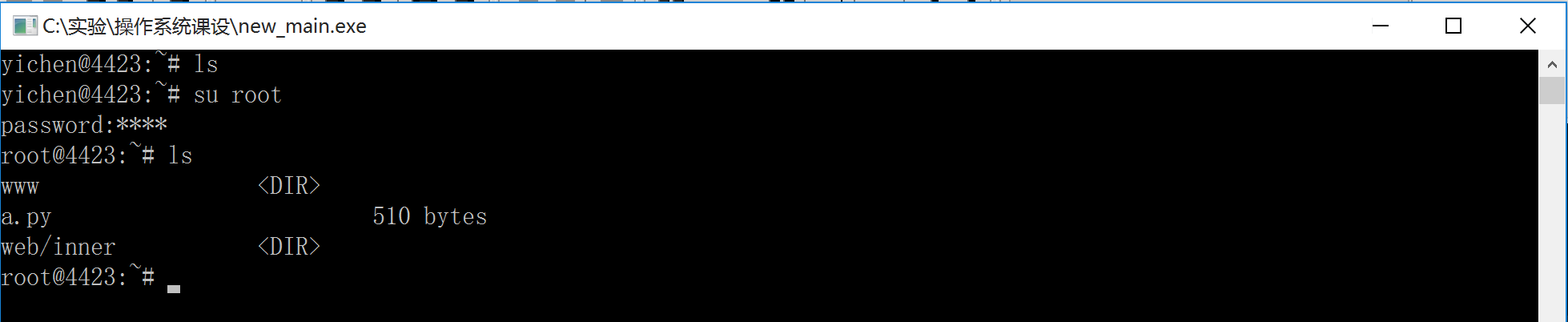
12. 使用rmdir删除目录



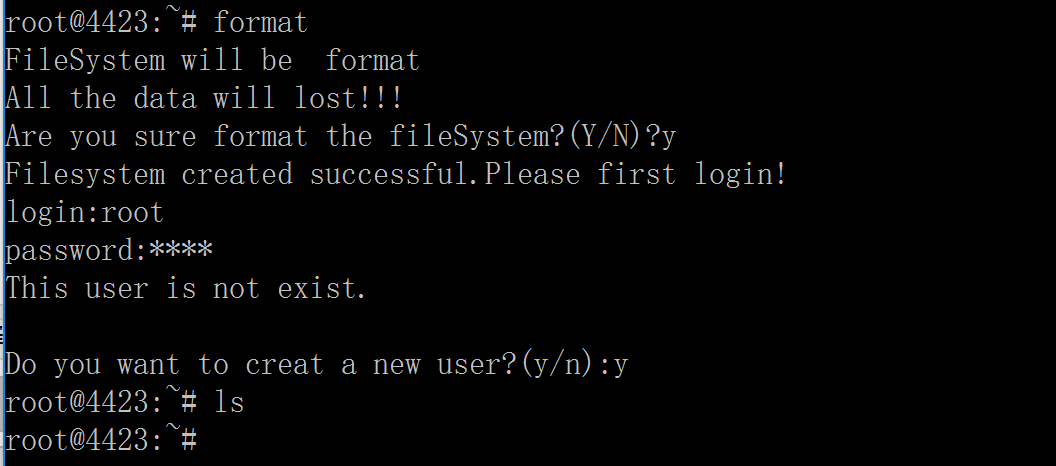
13. 使用exit退出系统



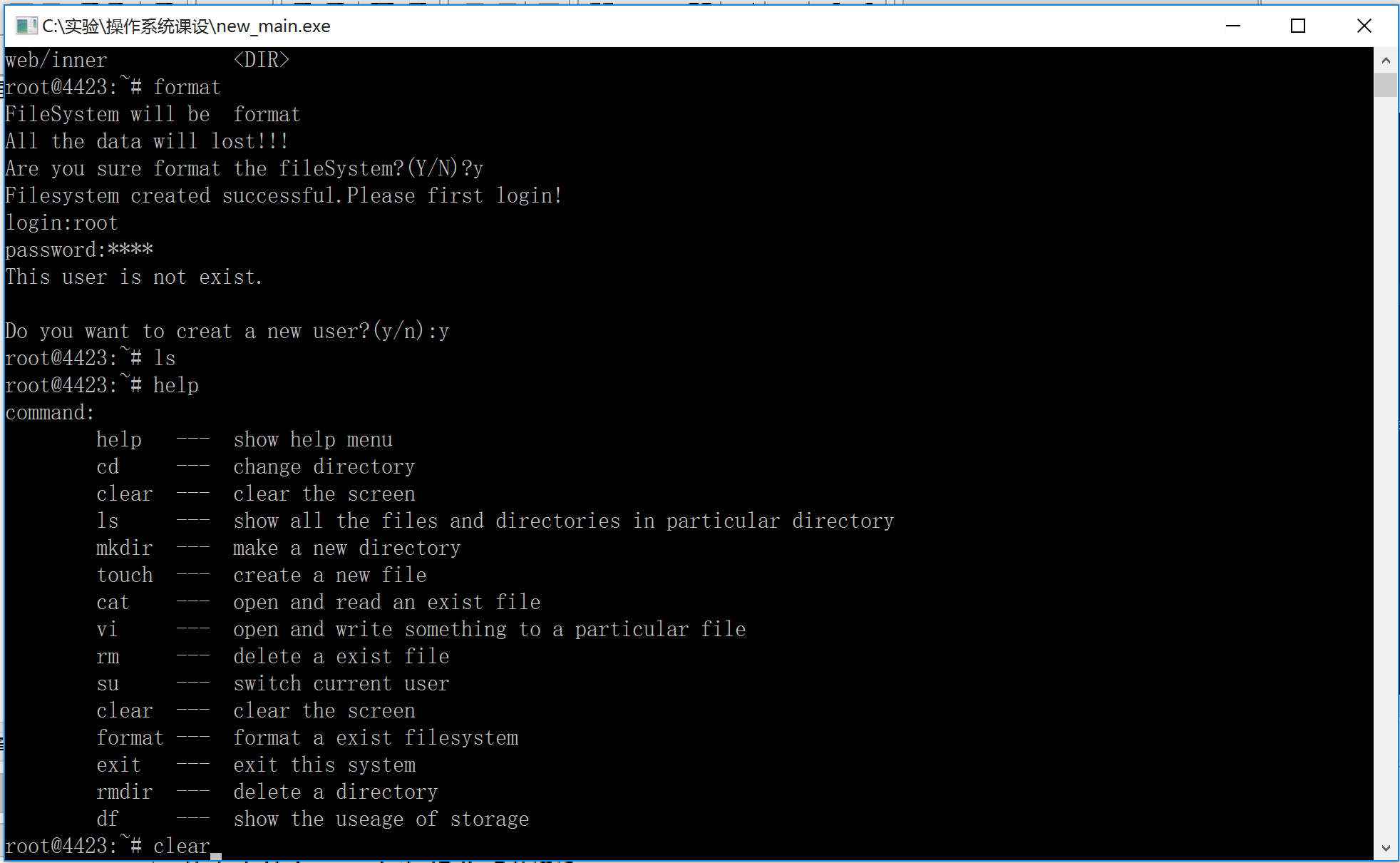
14. 使用su命令切换当前用户



15. 使用format格式化系统磁盘

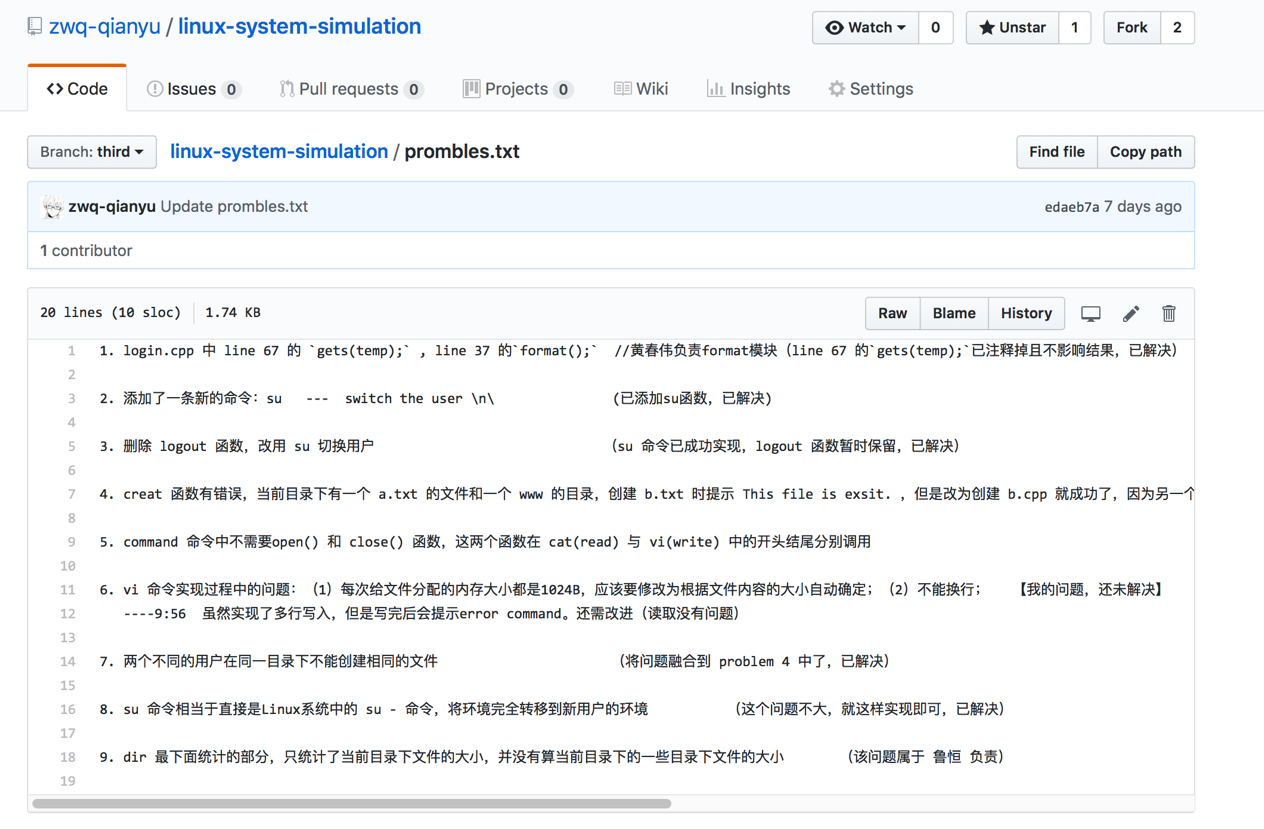


16. 使用clear清屏





1. 总结
2. 过程中遇到的问题



1). login.cpp 中 line 67 的 `gets(temp);` , line 37 的`format();` //黄春伟负责format模块（line 67 的`gets(temp);`已注释掉且不影响结果，已解决）

2). 添加了一条新的命令：su --- switch the user \n\ (已添加su函数，已解决)

3). 删除 logout 函数，改用 su 切换用户 （su 命令已成功实现，logout 函数暂时保留，已解决）

4). creat 函数有错误，当前目录下有一个 a.txt 的文件和一个 www 的目录，创建 b.txt 时提示 This file is exsit. ，但是改为创建 b.cpp 就成功了，因为另一个用户有 b.txt 文件。另外，即使同一用户在不同目录下创建的文件也不能重名，这是有问题的。 （该问题属于 黄春伟 负责）

5). command 命令中不需要open() 和 close() 函数，这两个函数在 cat(read) 与 vi(write) 中的开头结尾分别调用

6). vi 命令实现过程中的问题：（1）每次给文件分配的内存大小都是1024B，应该要修改为根据文件内容的大小自动确定；（2）不能换行； 【我的问题，还未解决】

----9:56 虽然实现了多行写入，但是写完后会提示error command。还需改进（读取没有问题）

7). 两个不同的用户在同一目录下不能创建相同的文件 （将问题融合到 problem 4 中了，已解决）

8). su 命令相当于直接是Linux系统中的 su - 命令，将环境完全转移到新用户的环境 （这个问题不大，就这样实现即可，已解决）

9). dir 最下面统计的部分，只统计了当前目录下文件的大小，并没有算当前目录下的一些目录下文件的大小 （该问题属于 鲁恒 负责）

上面这些问题都是在开发过程中我遇到的，目前都已经解决。下面还有一些我们在对各自的部分进行融合时出现的问题。



在将3个人分别负责的模块合到一起时出现的问题：

1). 定义的变量不同：

（1）用户命令的参数个数----钟伍全：int argc;--------鲁恒：vector<string>vc\_of\_str;

（2）用户命令的参数---------钟伍全：char \*argv[5];--------鲁恒：string s1, s2;

2). 钟伍全使用的纯C，鲁恒代码中含有C++语法等

3). 因为变量定义的不同，导致使用到代表同一个数据的时候使用方式需要修改：

（1）钟伍全负责的open函数中，第7行的`!strcmp(inode\_array[i].file\_name, argv1)`和第10行中的 argv1 无法得到正确的数据，需要修改，第7行可以考虑使用`s2 == inode\_array[i].file\_name`；close函数中也有同样的问题。------猜测：由于是在cat和vi函数中调用open与close函数，并传递参数过去，所以只需对cat和vi函数进行修改即可（待修改实验确定）。

（2）钟伍全负责的cat（read）和vi（write）函数中，因为变量定义不同需要修改一些内容,同时可以将部分C语言的写法换为简介的C++语法。

4). 钟伍全同时写了功能差不多的su函数和logout函数，command函数以及一些其他函数需要修改一下。

处理方式：

对1): 考虑简单简介，采用鲁恒的方式来定义 用户命令的参数个数 和 用户命令的参数个数。

对2)： C 和 C++ 混用，以简单简介为原则。

对3)： 待修改--->钟伍全。

对4)： 待修改--->钟伍全。

这些问题目前也都已经解决了。同时，上面所有问题都记录在GitHub上，可以随时查看。

1. 个人总结与感触

这是我第一次和别人一起合作写一个项目，并且同时使用GitHub进行版本控制。虽然以前也是用GitHub进行版本控制开发，但都是自己一个人做开发。这次是和别人一起，这让我感受到了版本控制在团队开发中的重要性。

同时，这次作为队长，让我提升了不少能力。比如系统总体架构设计、项目任务分工、团队任务意见协调等等。感受到了团队开发相比于个人开发的便利与轻松。

最后，这个操作系统课设。因为是模拟Unix文件系统，这让我更加深刻地了解了Unix系统底层是如何给文件系统分配内存、存储空间等的。对Linux系统的操作也更加熟练了。

参考文献：

1.《计算机操作系统教程（第4版）习题解答与实验指导》 张饶学 清华大学出版社

附录：

head.h

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <string>

#include <iostream>

#include <vector>

using namespace std;

#define BLKSIZE 512 // 数据块的大小

#define BLKNUM 512 // 数据块的块数

#define INODESIZE 32 // i节点的大小

#define INODENUM 32 // i节点的数目

#define FILENUM 8 // 打开文件表的数目

// 用户(20B)

typedef struct

{

char user\_name[10]; // 用户名

char password[10]; // 密码

} User;

// i节点(32B)

typedef struct

{

short inum; // 文件i节点号

char file\_name[10]; // 文件名

char type; // 文件类型

char user\_name[10]; // 文件所有者

short iparent; // 父目录的i节点号

short length; // 文件长度

short address[2]; // 存放文件的地址

} Inode;

// 打开文件表(16B)

typedef struct

{

short inum; // i节点号

char file\_name[10]; // 文件名

short mode; // 读写模式(1:read, 2:write,

// 3:read and write)

short offset; // 偏移量

} File\_table;

// 申明函数

void login(void);

void init(void);

int analyse();

void save\_inode(int);

int get\_blknum(void);

void read\_blk(int);

void write\_blk(int);

void release\_blk(int);

void pathset();

void delet(int innum);

int check(int i);

// 用户命令处理函数

void help(void);

void cd(void);

void dir(void);

void mkdir(void);

void touch(void);

void open(int, string);

void cat(void);

void vi(void);

void close(string);

void su();

void del(void);

void logout(void);

void command(void);

void rd();

void quit();

void df();

main.cpp

//#include "stdafx.h"

#include "head.h"

char choice;

vector<string>vc\_of\_str;

string s1, s2;

int inum\_cur; // 当前目录

char temp[2 \* BLKSIZE]; // 缓冲区

User user; // 当前的用户

char bitmap[BLKNUM]; // 位图数组

Inode inode\_array[INODENUM]; // i节点数组

File\_table file\_array[FILENUM]; // 打开文件表数组

char image\_name[10] = "hd.dat"; // 文件系统名称

FILE \*fp; // 打开文件指针

// "help", "cd", "dir", "mkdir", "touch", "open","read", "write", "close", "delete", "logout", "clear", "format","quit","rd"

const string Commands[] = { "help", "cd", "ls", "mkdir", "touch", "open","cat", "vi", "close", "rm", "su", "clear", "format","exit","rmdir","df" };

// bin/xx 给出进入bin即可

int readby() { //根据当前目录和第二个参数确定转过去的目录

int result\_cur = 0; string s = s2;

if (s.find('/') != -1) {

s = s.substr(0, s.find\_last\_of('/') + 1);

}

else {

s = "";

}

int temp\_cur = inum\_cur;

vector<string>v;

while (s.find('/') != -1) {

v.push\_back(s.substr(0, s.find\_first\_of('/')));

s = s.substr(s.find\_first\_of('/') + 1);

}

if (v.size() == 0) {

return inum\_cur;

}if (v[0].length() == 0) {

temp\_cur = 0;

}

else if (v[0] == "..") {

if (inode\_array[inum\_cur].inum > 0) {

temp\_cur = inode\_array[inum\_cur].iparent;

}

}

else {

int i;

for (i = 0; i < INODENUM; i++) {

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].iparent == inum\_cur) &&

(inode\_array[i].type == 'd') &&

inode\_array[i].file\_name == v[0]) {

break;

}

}

if (i == INODENUM) {

return -1;

}

else {

temp\_cur = i;

}

}

int i;

for (unsigned int count = 1; count < v.size(); count++) {

for (i = 0; i < INODENUM; i++) {

if ((inode\_array[i].inum > 0) &&//是否为空

(inode\_array[i].iparent == temp\_cur) &&

(inode\_array[i].type == 'd') &&

inode\_array[i].file\_name == v[count]) {

break;

}

}

if (i == INODENUM) {

return -1;

}

else {

temp\_cur = i;

}

}

result\_cur = temp\_cur;

return result\_cur;

}

//创建映像hd，并将所有用户和文件清除

void format(void)

{

FILE \*fp;

int i;

Inode inode;

printf("FileSystem will be format \n");

printf("All the data will lost!!!\n");

printf("Are you sure format the fileSystem?(Y/N)?");

scanf("%c", &choice);

gets(temp); // 更改

if ((choice == 'y') || (choice == 'Y'))

{

fp = fopen(image\_name, "w+b");

if (fp == NULL)

{

printf("Can't open file %s\n", image\_name);

getch();

exit(-1);

}

//成功打开后，位图数组置0

for (i = 0; i < BLKSIZE; i++)

fputc('0', fp); //

inode.inum = 0;

inode.iparent = 0;

inode.length = 0;

inode.address[0] = -1;

inode.address[1] = -1;

inode.type = 'd';

strcpy(inode.file\_name, "/");

strcpy(inode.user\_name, "all");

fwrite(&inode, sizeof(Inode), 1, fp);

inode.inum = -1;

//对32个节点进行格式化

for (i = 0; i < 31; i++)

fwrite(&inode, sizeof(Inode), 1, fp);

//数据块进行格式化

for (i = 0; i < BLKNUM\*BLKSIZE; i++)

fputc('\0', fp);

fclose(fp);

//清除用户信息

fp = fopen("user.txt", "w+");

if (fp == NULL)

{

printf("Can't create file %s\n", "user.txt");

exit(-1);

}

fclose(fp);

printf("Filesystem created successful.Please first login!\n");

}

return;

}

void login() {

/\*功能: 用户登陆，如果是新用户则创建用户\*/

char \*p;

int flag;

char user\_name[10];

char password[10];

char file\_name[10] = "user.txt";

char choice; //选择是否（y/n）

do {

printf("login:");

gets(user\_name);

printf("password:");

p = password;

while (\*p = \_getch()) {

if (\*p == 0x0d) //当输入回车键时，0x0d为回车键的ASCII码

{

\*p = '\0'; //将输入的回车键转换成空格

break;

}

printf("\*"); //将输入的密码以"\*"号显示

p++;

}

flag = 0;

if ((fp = fopen(file\_name, "r+")) == NULL) {

printf("\nCan't open file %s.\n", file\_name);

printf("This filesystem is not exist now, it will be create~~~\n");

format(); //黄春伟负责这个模块

login();

}

while (!feof(fp)) {

fread(&user, sizeof(User), 1, fp);

// 已经存在的用户, 且密码正确

if (!strcmp(user.user\_name, user\_name) &&

!strcmp(user.password, password)) {

fclose(fp);

printf("\n");

return; //登陆成功，直接跳出登陆函数

}

// 已经存在的用户, 但密码错误

else if (!strcmp(user.user\_name, user\_name)) {

printf("\nThis user is exist, but password is incorrect.\n");

flag = 1; //设置flag为1，表示密码错误，重新登陆

fclose(fp);

break;

}

}

if (flag == 0) {

printf("\nThis user is not exist.\n");

break; //用户不存在，先跳出循环创建新用户

}

} while (flag);

// 创建新用户

if (flag == 0) {

printf("\nDo you want to creat a new user?(y/n):");

scanf("%c", &choice);

if ((choice == 'y') || (choice == 'Y')) {

strcpy(user.user\_name, user\_name);

strcpy(user.password, password);

fwrite(&user, sizeof(User), 1, fp);

fclose(fp);

return;

}

if ((choice == 'n') || (choice == 'N'))

login();

}

}

// 功能: 将所有i节点读入内存

void init(void)

{

int i;

if ((fp = fopen(image\_name, "r+b")) == NULL)

{

printf("Can't open file %s.\n", image\_name);

exit(-1);

}

// 读入位图

for (i = 0; i < BLKNUM; i++)

bitmap[i] = fgetc(fp);

// 显示位图

// 读入i节点信息

for (i = 0; i < INODENUM; i++)

fread(&inode\_array[i], sizeof(Inode), 1, fp);

// 显示i节点

// 当前目录为根目录

inum\_cur = 0;

// 初始化打开文件表

for (i = 0; i < FILENUM; i++)

file\_array[i].inum = -1;

}

void df() {

int count\_inode = 0, count\_mem = 0;

for (int i = 0; i < INODENUM; i++) {

if (inode\_array[i].inum > 0)count\_inode++;

}

for (int i = 0; i < BLKNUM; i++) {

if (bitmap[i] == '1')count\_mem++;

}

cout << "Inode use:" << count\_inode << "/" << INODENUM << endl;

cout << "Store use:" << count\_mem << "/" << BLKNUM << endl;

}

void StrListForCom() {

vc\_of\_str.clear();

vc\_of\_str.push\_back("cd");

vc\_of\_str.push\_back("mkdir");

vc\_of\_str.push\_back("ls");

vc\_of\_str.push\_back("vi");

vc\_of\_str.push\_back("sudo");

vc\_of\_str.push\_back("rmfile");

vc\_of\_str.push\_back("touch");

vc\_of\_str.push\_back("man");

vc\_of\_str.push\_back("cat");

vc\_of\_str.push\_back("clear");

vc\_of\_str.push\_back("rmdir");

vc\_of\_str.push\_back("df");

}

void StrListForAdd() {

vc\_of\_str.clear();

int temp\_cur;

temp\_cur = readby();

for (int i = 0; i < INODENUM; i++) {

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].iparent == temp\_cur)) {

if (inode\_array[i].type == 'd' && !strcmp(inode\_array[i].user\_name, user.user\_name))

{

string temp = inode\_array[i].file\_name;

temp += '/';

vc\_of\_str.push\_back(temp);

}

if (inode\_array[i].type == '-' && !strcmp(inode\_array[i].user\_name, user.user\_name))

{

vc\_of\_str.push\_back(inode\_array[i].file\_name);

}

}

}

}

// 结果: 0-14为系统命令, 15为命令错误

int analyse()

{

string s = ""; s1 = ""; s2 = "";

int tabcount = 0;

int res = 0;

while (1) {

s1 = s;

if (s.find(' ') == -1)s2 = "";

else {

while (s1.length()>0 && s1[s1.length() - 1] == ' ') {

s1 = s1.substr(0, s1.length() - 1);

}

while (s1.length()>0 && s1[0] == ' ') {

s1 = s1.substr(1);

}

if (s1.find(' ') == -1)s2 = "";

else

s2 = s1.substr(s1.find\_first\_of(' ') + 1);

while (s2.length()>0 && s2[s2.length() - 1] == ' ') {

s2 = s2.substr(0, s2.length() - 1);

}

while (s2.length()>0 && s2[0] == ' ') {

s2 = s2.substr(1);

}

s1 = s1.substr(0, s1.find\_first\_of(' '));

}

int ch = \_getch();

if (ch == 8) { //退格

if (!s.empty()) {

printf("%c", 8);

printf(" ");

printf("%c", 8);

s = s.substr(0, s.length() - 1);

}

}

else if (ch == 13) { //回车

for (res = 0; res < 16; res++) {

if (s1 == Commands[res])break;

}

break;

}

else if (ch == 9) { //tab

int count = 0; vector<int>v;

string tstr;

if (s.find(' ') != -1) {

tstr = s.substr(s.find\_last\_of(' ') + 1);

if (tstr.find('/') != -1) {

tstr = tstr.substr(tstr.find\_last\_of('/') + 1);

}

StrListForAdd();

}

else {

tstr = s;

StrListForCom();

}

for (unsigned int i = 0; i < vc\_of\_str.size(); i++) {

if (vc\_of\_str[i].length() >= tstr.length() && vc\_of\_str[i].substr(0, tstr.length()) == tstr) {

count++; v.push\_back(i);

}

}

//cout << "count:" << count<<endl;

//cout << "tstr:" << tstr<<endl;

if (count < 1) {

if (s.find(' ') == -1) {

s.push\_back(' ');

printf(" ");

}

tabcount = -1;

}

if (count == 1) {

for (unsigned int i = tstr.length(); i < vc\_of\_str[v[0]].length(); i++) {

s.push\_back(vc\_of\_str[v[0]][i]);

printf("%c", vc\_of\_str[v[0]][i]);

}

if (s.find(' ') == -1) {

s.push\_back(' ');

printf(" ");

}

tabcount = -1;

}

if (count > 1 && tabcount) {

cout << "\n";

cout << vc\_of\_str[v[0]];

for (unsigned int i = 1; i < v.size(); i++) {

cout << " " << vc\_of\_str[v[i]];

}

cout << endl;

pathset();

cout << s;

tabcount = -1;

}

}

else if (ch == ' ') {

printf("%c", ch);

s.push\_back(ch);

}

else {

printf("%c", ch);

s.push\_back(ch);

}

//用于处理按两次tab

if (ch == 9) {

tabcount++;

}

else {

tabcount = 0;

}

}

if (s1 == "") {

return -1;

}

printf("\n");

return res;

}

void save\_inode(int num){

/\*功能: 将num号i节点保存到hd.dat\*/

if ((fp = fopen(image\_name, "r+b")) == NULL){

printf("Can't open file %s\n", image\_name);

exit(-1);

}

fseek(fp, 512 + num \* sizeof(Inode), SEEK\_SET);

fwrite(&inode\_array[num], sizeof(Inode), 1, fp);

fclose(fp);

}

int get\_blknum(void){

/\*功能: 申请一个数据块\*/

int i;

for (i = 0; i < BLKNUM; i++)

if (bitmap[i] == '0') break;

// 未找到空闲数据块

if (i == BLKNUM){

printf("Data area is full.\n");

exit(-1);

}

bitmap[i] = '1';

if ((fp = fopen(image\_name, "r+b")) == NULL){

printf("Can't open file %s\n", image\_name);

exit(-1);

}

fseek(fp, i, SEEK\_SET);

fputc('1', fp);

fclose(fp);

return i;

}

void read\_blk(int num){

/\*功能: 将i节点号为num的文件读入temp \*/

int i, len;

char ch;

int add0, add1;

len = inode\_array[num].length;

add0 = inode\_array[num].address[0];

if (len > 512)

add1 = inode\_array[num].address[1];

if ((fp = fopen(image\_name, "r+b")) == NULL){

printf("Can't open file %s.\n", image\_name);

exit(-1);

}

fseek(fp, 1536 + add0 \* BLKSIZE, SEEK\_SET);

ch = fgetc(fp);

for (i = 0; (i < len) && (ch != '\0') && (i < 512); i++){

temp[i] = ch;

ch = fgetc(fp);

}

if (i > 512){

fseek(fp, 1536 + add0 \* BLKSIZE, SEEK\_SET);

ch = fgetc(fp);

for (; (i < len) && (ch != '\0'); i++){

temp[i] = ch;

ch = fgetc(fp);

}

}

temp[i] = '\0';

fclose(fp);

}

void write\_blk(int num){

/\*功能: 将temp的内容输入hd的数据区\*/

int i, len;

int add0, add1;

add0 = inode\_array[num].address[0];

len = inode\_array[num].length;

if ((fp = fopen(image\_name, "r+b")) == NULL){

printf("Can't open file %s.\n", image\_name);

exit(-1);

}

fseek(fp, 1536 + add0 \* BLKSIZE, SEEK\_SET);

for (i = 0; (i<len) && (temp[i] != '\0') && (i < 512); i++)

fputc(temp[i], fp);

if (i == 512){

add1 = inode\_array[num].address[1];

fseek(fp, 1536 + add1 \* BLKSIZE, SEEK\_SET);

for (; (i < len) && (temp[i] != '\0'); i++)

fputc(temp[i], fp);

}

fputc('\0', fp);

fclose(fp);

}

// 功能: 释放文件块号为num的文件占用的空间

void release\_blk(int num)

{

FILE \*fp;

if ((fp = fopen(image\_name, "r+b")) == NULL)

{

printf("Can't open file %s\n", image\_name);

exit(-1);

}

bitmap[num] = '0';

fseek(fp, num, SEEK\_SET);

fputc('0', fp);

fclose(fp);

}

void help(){

/\*功能: 显示帮助命令\*/

printf("command: \n\

help --- show help menu \n\

cd --- change directory \n\

clear --- clear the screen \n\

ls --- show all the files and directories in particular directory \n\

mkdir --- make a new directory \n\

touch --- create a new file \n\

cat --- open and read an exist file \n\

vi --- open and write something to a particular file \n\

rm --- delete a exist file \n\

su --- switch current user \n\

clear --- clear the screen \n\

format --- format a exist filesystem \n\

exit --- exit this system \n\

rmdir --- delete a directory \n\

df --- show the useage of storage \n");

}

//设置文件路径

void pathset()

{

string s;

if (inode\_array[inum\_cur].inum == 0)s = "";

else {

int temp = inum\_cur;

while (inode\_array[temp].inum != 0) {

s = inode\_array[temp].file\_name + s;

s = '/' + s;

temp = inode\_array[temp].iparent;

}

}

cout << user.user\_name << "@" << "4423" << ":~" << s << "# ";

}

// 功能: 切换目录(cd .. 或者 cd dir1)

void cd()

{

int temp\_cur;

if (s2.length() == 0) {

temp\_cur = 0;

}

else {

if (s2[s2.length() - 1] != '/')s2 += '/';

temp\_cur = readby();

}

if (temp\_cur != -1) {

inum\_cur = temp\_cur;

}

else {

cout << "No Such Directory" << endl;

}

}

// 功能: 显示当前目录下的子目录和文件(dir)

void dir(void)

{

int temp\_cur;

int i = 0;

if (s2.length() == 0) {

temp\_cur = inum\_cur;

}

else {

if (s2[s2.length() - 1] != '/')s2 += '/';

temp\_cur = readby();

if (temp\_cur == -1 || inode\_array[temp\_cur].type != 'd') {

cout << "No Such Directory" << endl;

return;

}

}

if (temp\_cur != -1 && inode\_array[temp\_cur].type == 'd')

for (i = 0; i < INODENUM; i++)

{

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].iparent == temp\_cur)

&& !strcmp(inode\_array[i].user\_name, user.user\_name))

{

if (inode\_array[i].type == 'd')

{

printf("%-20s<DIR>\n", inode\_array[i].file\_name);

}

if (inode\_array[i].type == '-')

{

printf("%-20s%12d bytes\n", inode\_array[i].file\_name, inode\_array[i].length);

}

}

}

}

void rm(int inode) {

int i;

for (i = 0; i < INODENUM; i++) {

if (inode\_array[i].iparent == inode && !strcmp(inode\_array[i].user\_name, user.user\_name)) {

if (inode\_array[i].type == 'd')rm(i);

else delet(i);

}

}

delet(inode);

}

//rmdir dir或rmdir dir/a/b

// 功能: 删除目录树(rd dir1)

void rmdir()

{

if (s2.length() == 0) {

printf("No Such Directory\n");

return;

}

if (s2[s2.length() - 1] != '/')s2 += '/';

int temp\_cur = readby();

if (temp\_cur == -1)printf("No Such Directory\n");

else {

if (inode\_array[temp\_cur].type != 'd') {

printf("That's A File!\n");

}

else {

int temp = inode\_array[inum\_cur].iparent;

while (temp != 0) {

if (temp == temp\_cur || temp\_cur == 0) {

printf("Can't delete your father directory\n");

return;

}

temp = inode\_array[temp].iparent;

}

rm(temp\_cur);

}

}

return;

}

// 功能: 在当前目录下创建子目录(mkdir dir1)

void mkdir(void)

{

int i;

if (s2.length() == 0) {

cout << "Please input name" << endl;

return;

}

// 遍历i节点数组, 查找未用的i节点

for (i = 0; i < INODENUM; i++) {

if (inode\_array[i].iparent == inum\_cur && inode\_array[i].type == 'd'

&&inode\_array[i].file\_name == s2 && inode\_array[i].inum > 0

&& !strcmp(inode\_array[i].user\_name, user.user\_name)) {

break;

}

}

if (i != INODENUM) {

printf("There is directory having same name.\n");

return;

}

for (i = 0; i < INODENUM; i++)

if (inode\_array[i].inum < 0) break;

if (i == INODENUM)

{

printf("Inode is full.\n");

exit(-1);

}

inode\_array[i].inum = i;

strcpy(inode\_array[i].file\_name, s2.data());

inode\_array[i].type = 'd';

strcpy(inode\_array[i].user\_name, user.user\_name);

inode\_array[i].iparent = inum\_cur;

inode\_array[i].length = 0;

save\_inode(i);

}

// 功能: 在当前目录下创建文件(creat file1)

void touch(void)

{

if (s2.length() == 0) {

printf("Please input filename.\n");

return;

}

int i, temp\_cur; string temps1, temps2;

if (s2.find('/') != -1) {

temps1 = s2.substr(0, s2.find\_last\_of('/') + 1);

temps2 = s2.substr(s2.find\_last\_of('/') + 1);

s2 = temps1;

temp\_cur = readby();

if (temp\_cur == -1) {

printf("No Such Directory\n");

return;

}

}

else {

temps2 = s2;

temp\_cur = inum\_cur;

}

for (i = 0; i < INODENUM; i++)

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].type == '-') &&

temps2 == inode\_array[i].file\_name&&

inode\_array[i].iparent == temp\_cur &&

!strcmp(inode\_array[i].user\_name, user.user\_name)) break;

if(i!=INODENUM){

printf("There is same file\n");

return;

}

for (i = 0; i < INODENUM; i++)

if (inode\_array[i].inum < 0) break;

if (i == INODENUM)

{

printf("Inode is full.\n");

exit(-1);

}

inode\_array[i].inum = i;

strcpy( inode\_array[i].file\_name, temps2.data());

inode\_array[i].type = '-';

strcpy(inode\_array[i].user\_name, user.user\_name);

inode\_array[i].iparent = temp\_cur;

inode\_array[i].length = 0;

save\_inode(i);

}

//检查当前I节点的文件是否属于当前用户

int check(int i)

{

int j;

char \*uuser, \*fuser;

uuser = user.user\_name;

fuser = inode\_array[i].user\_name;

j = strcmp(fuser, uuser);

if (j == 0) return 1;

else return 0;

}

void open(int mymode, string argv1) {

/\*功能: 打开当前目录下的文件(open file1)\*/

int i, inum, mode, filenum, chk;

for (i = 0; i < INODENUM; i++)

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].type == '-') &&

argv1 == inode\_array[i].file\_name &&

!strcmp(inode\_array[i].user\_name, user.user\_name) &&

inode\_array[i].iparent == inum\_cur) //判断是否在当前目录下

break;

if (i == INODENUM) {

cout << "This is no " + argv1 + " file...\n";

return;

}

inum = i;

chk = check(i); //检查该文件是否为当前用户的文件

mode = mymode;

if (chk != 1) {

printf("This file is not yours !\n");

return;

}

if ((mode < 1) || (mode > 3)) {

printf("Open mode is wrong.\n");

return;

}

filenum = i;

file\_array[filenum].inum = inum;

strcpy(file\_array[filenum].file\_name, inode\_array[inum].file\_name);

file\_array[filenum].mode = mode;

file\_array[filenum].offset = 0;

}

void cat() {

/\*功能: 从文件中读出字符(read file1)\*/

int i, start, num, inum;

/\*if (argc != 2){

printf("command cat must have one object. \n");

return;

}\*/

open(1, s2);

for (i = 0; i < FILENUM; i++)

if ((file\_array[i].inum > 0) &&

(s2 == file\_array[i].file\_name))

break;

inum = file\_array[i].inum;

//printf("The length of %s:%d.\n", argv[1], inode\_array[inum].length);

if (inode\_array[inum].length > 0) {

read\_blk(inum);

for (i = 0; temp[i] != '\0'; i++)

printf("%c", temp[i]);

printf("\n");

}

close(s2);

}

void vi() {

/\*功能: 向文件中写入字符(write file1)\*/

int i, inum, length;

//length = 0;

/\*if (argc != 2){

printf("Command write must have one args. \n");

return;

}\*/

open(3, s2);

for (i = 0; i < FILENUM; i++)

if ((file\_array[i].inum>0) &&

s2==file\_array[i].file\_name) break;

if (i == FILENUM){

cout << "Open " << s2 << " false.\n";

return;

}

inum = file\_array[i].inum;

//printf("The length of %s:%d\n", inode\_array[inum].file\_name, inode\_array[inum].length);

if (inode\_array[inum].length == 0){

printf("The length you want to write(0-1024):");

scanf("%d", &length);

gets(temp);

if ((length < 0) && (length >1024)){

printf("You can't creat a file which data less than 0 byte or more than 1024 bytes.'.\n");

return;

}

inode\_array[inum].length = length;

inode\_array[inum].address[0] = get\_blknum();

if (length > 512)

inode\_array[inum].address[1] = get\_blknum();

save\_inode(inum);

printf("Input the data(end by #):\n");

//gets(temp);

char c;

string str;

while((c=getchar())!='#'){

if (c == 0x0d){ //当输入回车键时，0x0d为回车键的ASCII码

c = '\0'; //将输入的回车键转换成换行符

}

str = str + c;

}

for(int i=0;i<length;i++){

temp[i] = str[i];

}

write\_blk(inum);

}

else{

printf("Are you sure to write this file? Data in it will all be deleted.(y/n):");

scanf("%c", &choice);

if ((choice == 'y') || (choice == 'Y')){

printf("The length you want to write(0-1024):");

scanf("%d", &length);

gets(temp);

if ((length < 0) && (length >1024)){

printf("You can't creat a file which data less than 0 byte or more than 1024 bytes.'.\n");

return;

}

inode\_array[inum].length = length;

inode\_array[inum].address[0] = get\_blknum();

if (length > 512)

inode\_array[inum].address[1] = get\_blknum();

save\_inode(inum);

printf("Input the data(end by #):\n");

//gets(temp);

char c;

string str;

while((c=getchar())!='#'){

if (c == 0x0d){ //当输入回车键时，0x0d为回车键的ASCII码

c = '\0'; //将输入的回车键转换成换行符

}

str = str + c;

}

for(int i=0;i<length;i++){

temp[i] = str[i];

}

write\_blk(inum);

}

}

close(s2);

init(); //需要初始化一下，否则文件有一个文件名会出错

}

void close(string argv1) {

/\*功能: 关闭已经打开的文件(close file1)\*/

int i;

for (i = 0; i < FILENUM; i++)

if ((file\_array[i].inum > 0) &&

(argv1 == file\_array[i].file\_name)) break;

if (i == FILENUM) {

printf("This file didn't be opened.\n");

return;

}

else

file\_array[i].inum = -1;

}

void su() {

/\*功能: 切换当前用户(logout)\*/

char \*p;

int flag;

string user\_name;

char password[10];

char file\_name[10] = "user.txt";

fp = fopen(file\_name, "r"); //初始化指针，将文件系统的指针指向文件系统的首端(以只读方式打开文件)

/\*if (argc != 2){

printf("command su must have one object. \n");

return;

}\*/

do {

user\_name = s2;

printf("password:");

p = password;

while (\*p = \_getch()) {

if (\*p == 0x0d) { //当输入回车键时，0x0d为回车键的ASCII码

\*p = '\0'; //将输入的回车键转换成空格

break;

}

printf("\*"); //将输入的密码以"\*"号显示

p++;

}

flag = 0;

while (!feof(fp)) {

fread(&user, sizeof(User), 1, fp);

// 已经存在的用户, 且密码正确

if ((user.user\_name == user\_name) &&

!strcmp(user.password, password)) {

fclose(fp);

printf("\n");

return; //登陆成功，直接跳出登陆函数

}

// 已经存在的用户, 但密码错误

else if ((user.user\_name == user\_name)) {

printf("\nThis user is exist, but password is incorrect.\n");

flag = 1; //设置flag为1，表示密码错误，重新登陆

fclose(fp);

break;

}

}

if (flag == 0) {

printf("\nThis user is not exist.\n");

break; //用户不存在，直接跳出循环，进行下一条指令的输入

}

} while (flag);

}

//根据Inode节点号删存储

void delet(int innum)

{

inode\_array[innum].inum = -1;

if (inode\_array[innum].length >= 0)

{

release\_blk(inode\_array[innum].address[0]);

if (inode\_array[innum].length >= 512)

release\_blk(inode\_array[innum].address[1]);

}

save\_inode(innum);

}

// 功能: 删除文件

void rmfile(void)

{

if (s2.length() == 0) {

printf("This file doesn't exist.\n");

return;

}

int i, temp\_cur; string temps1, temps2;

if (s2.find('/') != -1) {

temps1 = s2.substr(0, s2.find\_last\_of('/') + 1);

temps2 = s2.substr(s2.find\_last\_of('/') + 1);

s2 = temps1;

temp\_cur = readby();

}

else {

temps2 = s2;

temp\_cur = inum\_cur;

}

for (i = 0; i < INODENUM; i++)

if ((inode\_array[i].inum > 0) &&

(inode\_array[i].type == '-') &&

temps2 == inode\_array[i].file\_name&&

inode\_array[i].iparent == temp\_cur &&

!strcmp(inode\_array[i].user\_name, user.user\_name)) break;

if (i == INODENUM)

{

printf("This file doesn't exist.\n");

return;

}

delet(i);

}

// 功能: 退出当前用户(logout)

void logout()

{

login();

}

// 功能: 退出文件系统(quit)

void quit()

{

char choice;

printf("Do you want to exist(y/n):");

scanf("%c", &choice);

gets(temp);

if ((choice == 'y') || (choice == 'Y'))

exit(-1);

}

// 功能: 显示错误

void errcmd()

{

printf("Command Error!!!\n");

}

//清空内存中存在的用户名

void free\_user()

{

int i;

for (i = 0; i<10; i++)

user.user\_name[i] = '\0';

}

// 功能: 循环执行用户输入的命令, 直到logout

// "help", "cd", "dir", "mkdir", "touch", "open","read", "write", "close", "delete", "logout", "clear", "format","quit","rd"

void command(void)

{

system("cls");

do

{

pathset();

switch (analyse())

{

case -1:

printf("\n");

break;

case 0:

help();

break;

case 1:

cd();

break;

case 2:

dir();

break;

case 3:

mkdir();

break;

case 4:

touch();

break;

case 5:

//open();

break;

case 6:

cat();

break;

case 7:

vi();

break;

case 8:

//close();

break;

case 9:

rmfile();

break;

case 10:

su();

break;

case 11:

system("cls");

break;

case 12:

format();

init();

free\_user();

login();

break;

case 13:

quit();

break;

case 14:

rmdir();

break;

case 15:

df();

break;

case 16:

errcmd();

break;

default:

break;

}

} while (1);

}

// 主函数

int main(void)

{

login();

init();

command();

return 0;

}