北 京 林 业 大 学

2018 学年— 2019 学年 第 二 学期 操作系统 实习报告书

专　　业：信息管理与信息系统　班　　级：信息 17-2

姓　　名：杨楚峤　学　　号：171001220

实习地点：机房　辅导教师：孟伟

实习内容：文件系统模拟

实习环境：Windows 操作系统相应的开发平台和工具

0 引言

0.1 背景

Y.C.O.S.(Yang Chuqiao Operating System)是一个模拟多用户环境下的磁盘文件系统的软件（类 Linux 文件系统设计），支持在后台程序的调配下同时运行多个客户端。此软件面向个人用户，可以执行创建，整理，管理，删除各类数据以及目录文件等 25 个命令操作。用户可在使用过程中学习计算机文件系统的基本框架，并加深对计算机文件系统的理解。本软件为二次开发人员提供了良好的基础和平台，具有较好的拓展性和鲁棒性。

0.2 运行环境

☆推荐环境：

Windows 10
Visual Studio 2015

☆最低环境：

Windows 7 （以上）
Visual Studio 2013

0.3 相关文档

☆任务文件：

操作系统实习任务书；

☆相关文件：

程序运行说明书；

系统报错说明；
操作系统实习报告；

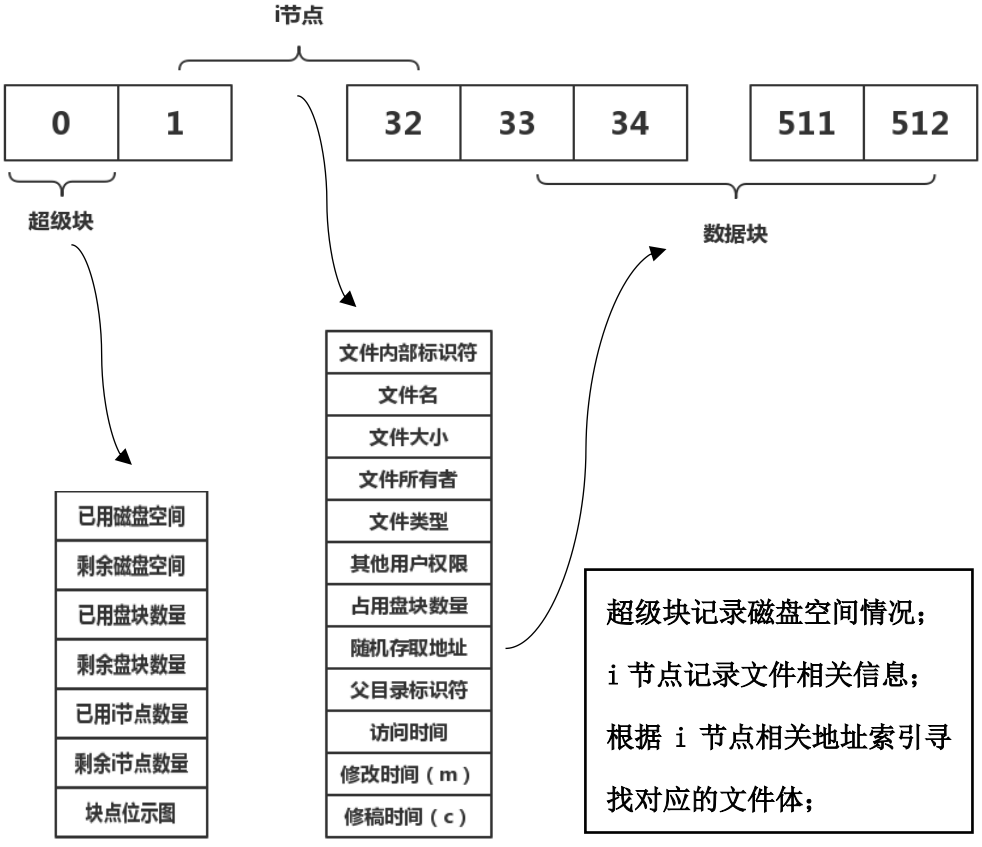
1 文件系统格式设计

1.1 文件系统基础

磁盘驱动器模拟：disk.dat；
文件逻辑结构：流式文件；
文件物理结构：随机文件；
磁盘空间管理：位示图；
文件安全机制：用户权限；
文件类型：二进制文件；

1.2 磁盘划分（单位全部为字节 B）

☆磁盘：0-512 号盘块
磁盘大小 disksize 2400 * 512；
盘块大小 blocksize 2400；
盘块块数 blockcount 512；



☆超级块：占用 0 号盘块

超级块大小 2400；
超级块块数 1；

☆i 节点：占用 1-32 号盘块
i 节点大小 inodesize 600；
i 节点块数 inodecount 32；

☆数据块：占用 33-512 号盘块
数据块大小 2400；
数据块块数 479；
数据块约占磁盘总空间 93.55%；

2 访问软件功能设计

系统目前支持 33 条命令（其中涵盖功能命令 25 条）

命令操作符	命令格式	作用介绍
cls	cls	清除屏幕
help	help	版本信息与帮助界面
login	login	登录一个新的账户
useradd	useradd	添加一个新的账户
exit	exit	结束本次服务
format	format	格式化虚拟磁盘
showsb	showsb	显示磁盘使用情况
time	time	显示当前系统时间
ver	ver	显示当前系统版本
pwd	pwd	显示当前路径
mkdir	mkdir dirname	创建一个新目录
dir\\ls	dir\\ls	显示当前目录下的信息与文件
cd	cd path	切换路径
mkfile	mkfile filename	创建一个新文件
rmdir	rmdir dirname	删除当前路径下一个目录（递归）
rm\\del	rmdir\\del filename	删除当前路径下一个文件
cat\\more	cat\\more filename	显示文件内容
rename	rename name	重命名一个文件或目录
chmod	chmod name mode	修改文件或目录的访问权限
attrib	attrib	显示当前目录下所有文件目录属性
find	find txt	寻找包含指定文本的文件
copy	copy filename path	将一个文件拷贝到另一目录
xcopy	xcopy dirname path	将一个目录拷贝到另一目录（递归）
export	export filename path	将虚拟磁盘内容导出到物理磁盘
import	import path	将物理磁盘导入到虚拟磁盘内容

命令操作符	命令格式	作用介绍
backup	backup	备份虚拟磁盘内容
recover	recover	恢复虚拟磁盘内容
diskupdate	diskupdate	更新虚拟磁盘
move	move filename path	移动文件至某个路径
movedir	movedir dirname path	移动目录至某个路径

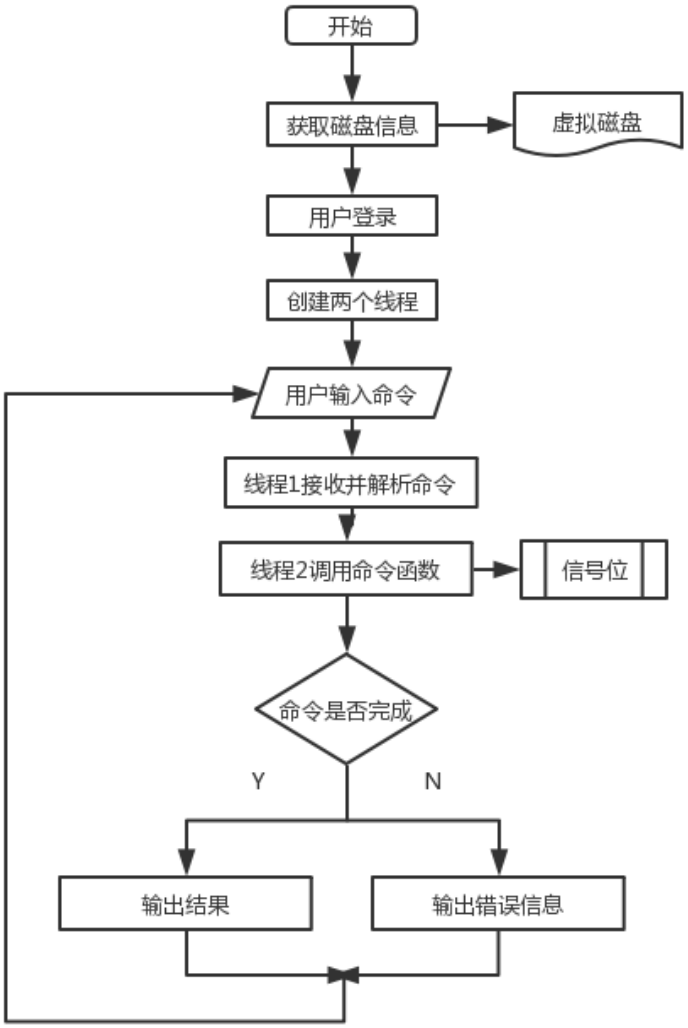
注意：

format 命令默认保留根目录；

backup 命令初始设置 a、b、c 三个目录，其中 a 目录下含子文件；

3 磁盘文件系统总体设计

3.1 整体流程



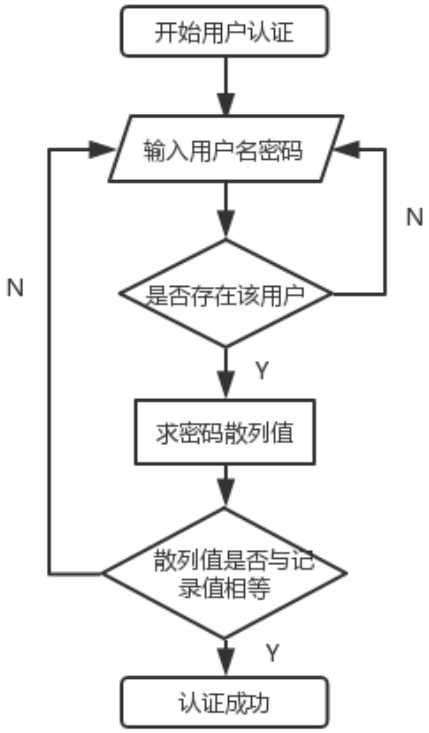
3.2 功能详解（选取典型功能）

3.2.1 用户身份验证 设计

☆算法：

当用户输入用户名和密码后，先将用户名与用户名列表中的每一个用户名进行比对，若未发现相同用户名，则输出错误提示。若找到相应用户，则开始比对密码，将密码做散列函数处理后，与用户列表中的密码散列值进行对比，若散列值相同，则认定登录成功。

☆流程图：



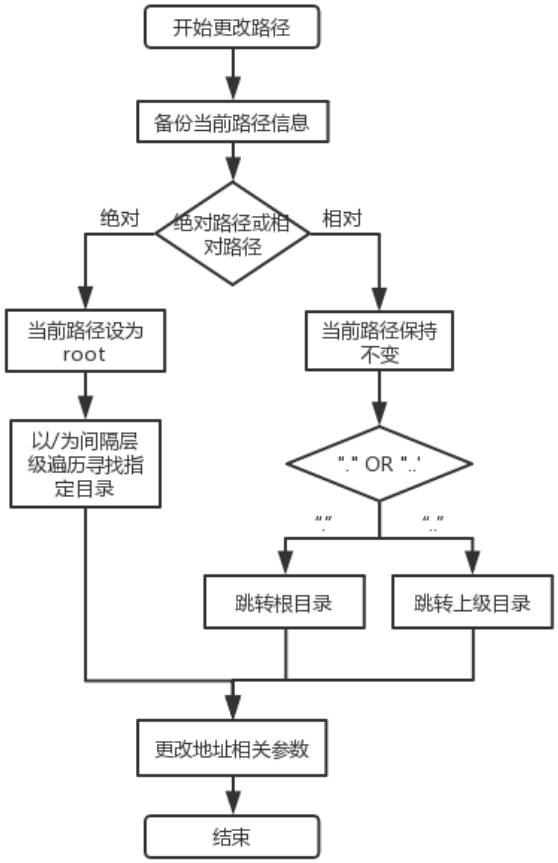
3.2.2 更改当前路径 设计

☆算法：

首先备份当前路径信息。

当用户输入路径后，首先判断绝对路径或相对路径，若为绝对路径则从根节点开始更改，若为相对路径则从当前节点开始更改。其次循环判断是否存在输入的第一部分内容，若存在则更改当前路径，若不存在则将当前路径回滚到之前状态，并输出“路径不存在”返回-1.若存在则更改当前节点路径，并将字符串指针移到下一个‘\’处。

☆流程图：

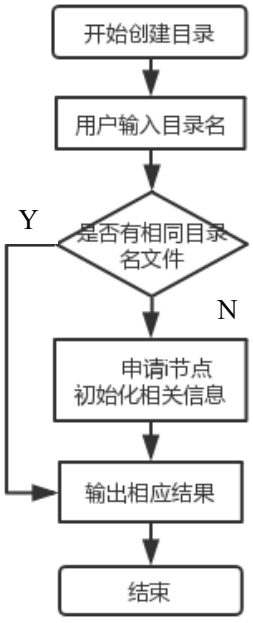


3.2.3 创建一个目录 设计

☆算法：

输入目录名后，检查在当前目录下是否含同名目录。如果没有，则申请 i 节点，初始化相关信息。

☆流程图：

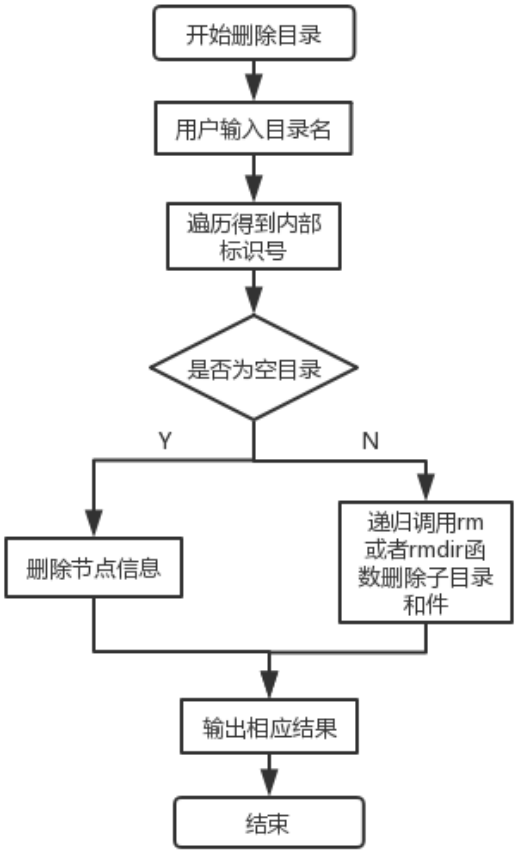


3.2.4 删除一个目录 设计

☆算法：

输入目录名后，遍历寻找内部标识号。查看目录是否为空目录。如果为空目录，则直接删除有关该目录的全部信息。否则，递归调用相应函数删除目录下的子目录和子文件，注意释放相关节点和盘块信息。

☆流程图：

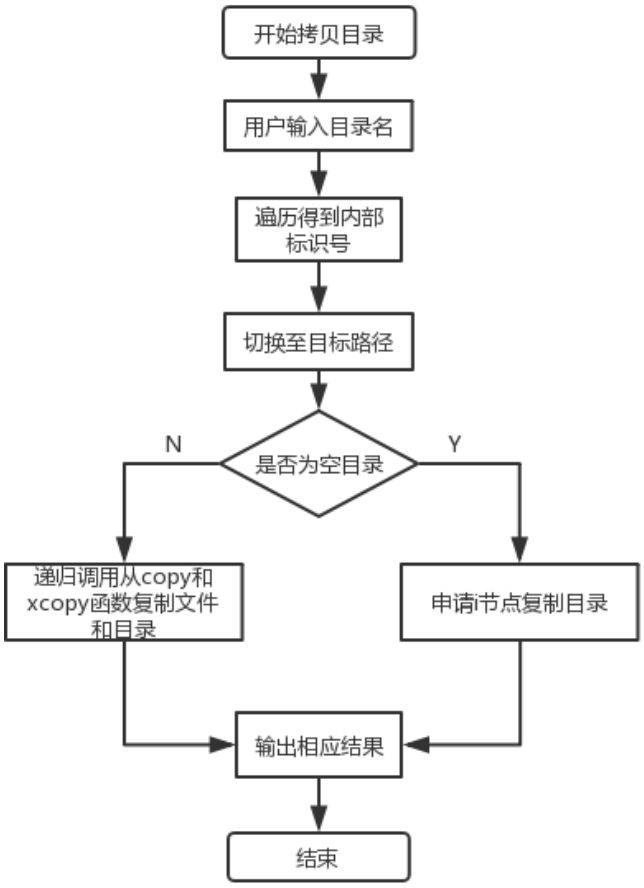


3.2.5 拷贝一个目录 设计

☆算法：

输入目录名后，遍历寻找内部标识号。查看目录是否为空目录。如果为空目录，则直接复制 i 节点即可。否则，递归调用相应函数依次复制目录下的子目录和子文件。

☆流程图：



3.2.6 移动一个目录 设计

☆算法：

先保存当前路径。调用相关函数复制本目录至指定路径，调用相关函数删除原有路径下的此目录。并更改相关路径以及目录信息。

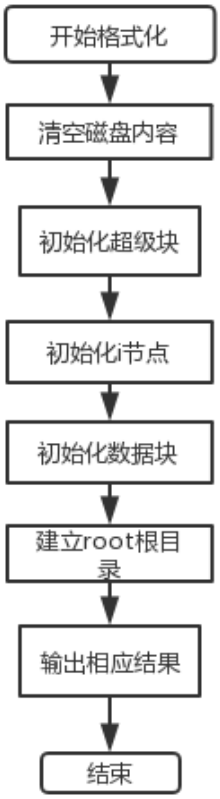
☆流程图：

3.2.7 格式化磁盘 设计

☆算法：

格式化磁盘即恢复磁盘初始化状态。算法设计为在清空磁盘空间后后，依次初始化超级块，i 节点，数据块，并自动建立 root 根目录。

☆流程图：

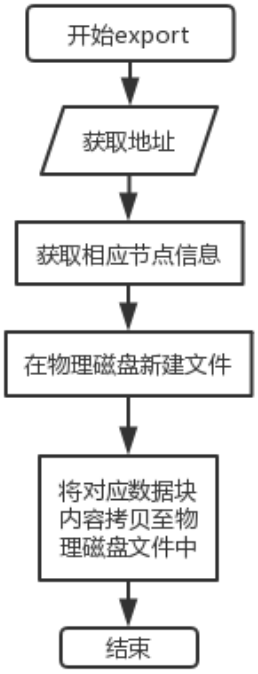


3.2.8 将虚拟磁盘内容导出到物理磁盘 设计

☆算法：

从虚拟磁盘中找出索引节点后将索引节点所连接的所有磁盘块依此读出，并依此写入磁盘中新建的文件中。

☆流程图：

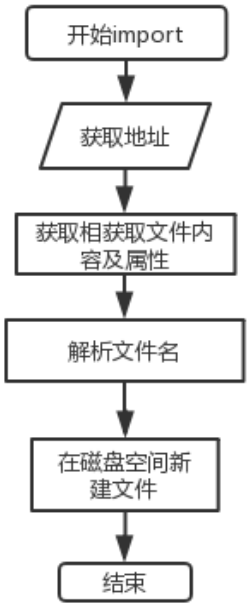


3.2.9 将物理磁盘导入到虚拟磁盘内容 设计

☆算法：

从物理磁盘中找到相应文件，获取文件内容以及大小等属性，将文件名解析后，在虚拟文件系统中新建文件，申请磁盘空间，存入文件内容。

☆流程图：

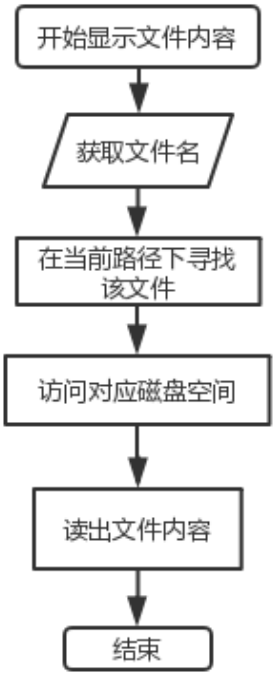


3.2.10 显示文件内容 设计

☆算法：

从缓冲区读取文件名，在当前路径下寻找该文件，并获取其相关节点信息。根据相关地址信息，寻找文件内容存储盘块，输出文件内容。

☆流程图：

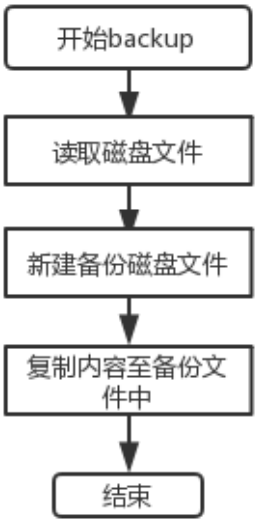


3.2.11 备份虚拟磁盘内容 设计

☆算法：

读取磁盘文件全部内容，新建备份磁盘文件，将保存的全部内容复制到备份文件中。

☆流程图：

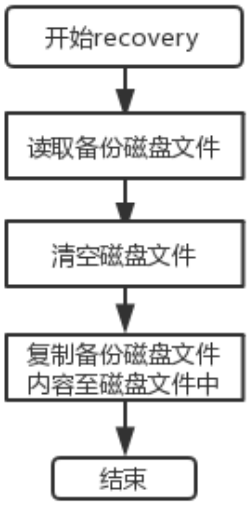


3.2.12 恢复虚拟磁盘内容 设计

☆算法：

读取备份磁盘文件全部内容，清空磁盘文件，将保存的全部内容复制到磁盘文件中。

☆流程图：

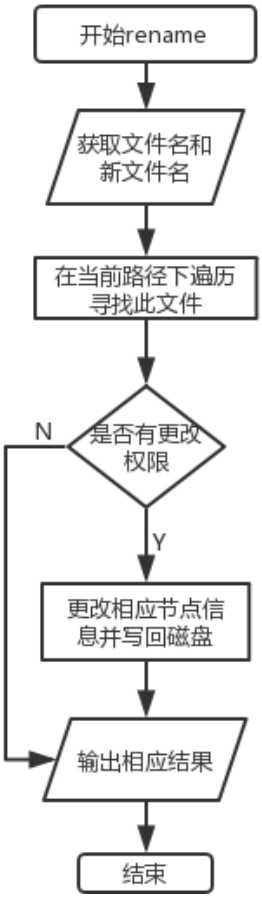


3.2.13 重命名一个文件或目录 设计

☆算法：

在判断权限与文件存在后，将磁盘中的索引节点读入缓冲区，并更改其信息。写回虚拟磁盘。

☆流程图：



3.2.14 显示磁盘使用情况 设计

☆算法：

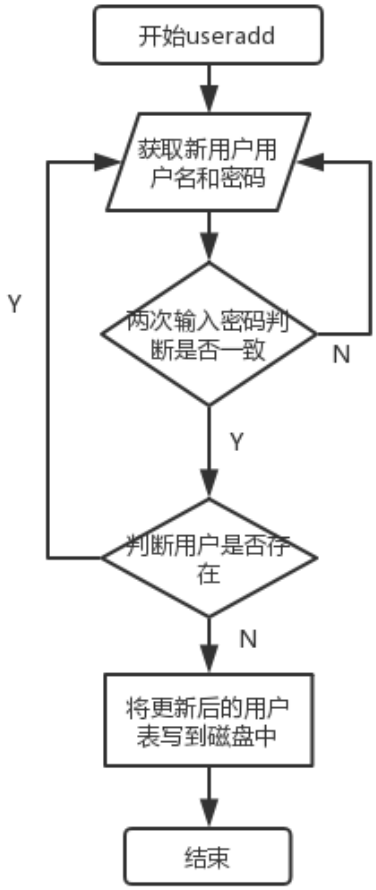
输出超级块属性信息。

3.2.15 添加一个新的账户 设计

☆算法：

首先取出用户表，然后输入用户名判断是否已经存在，若存在重新输入。输入密码，判断两次密码是否相同，相同则写入文件中。

☆流程图：

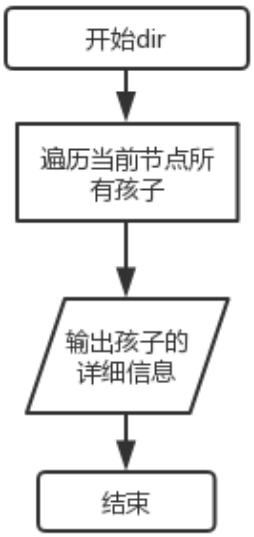


3.2.16 显示当前目录下的信息与文件 设计

☆算法：

遍历当前索引节点的所有子节点，将信息取出并展示在控制台中。

☆流程图：

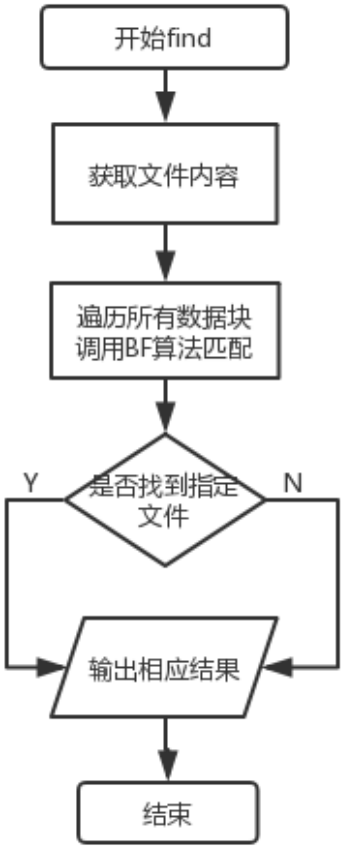


3.2.17 寻找包含指定文本的文件 设计

☆算法：

根据命令行中给出的字符串信息，遍历整个磁盘空间，利用 BF 算法寻找匹配内容，然后返回其盘号，再根据相应盘号寻找指定文件。

☆流程图：



3.2.18 更新虚拟磁盘 设计

☆算法：

遍历当前索引节点的所有子节点，将信息取出并展示在控制台中。

3.2.19 显示当前路径 设计

☆算法：

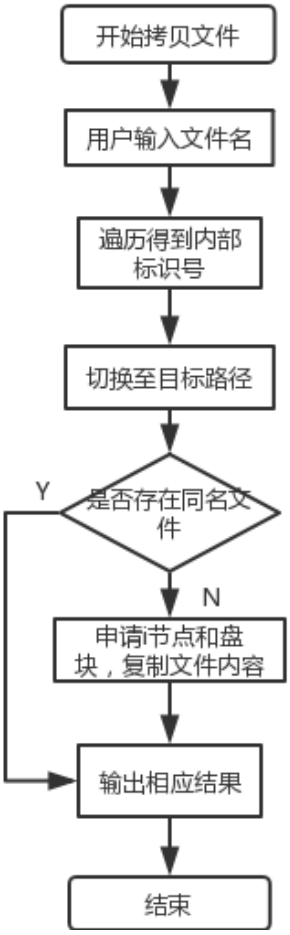
输出当前路径信息。

3.2.20 拷贝一个文件 设计

☆算法：

输入文件名后，遍历寻找内部标识号。切换到目标路径，查看是否存在同名文件。如果没有，则新申请 i 节点，新申请盘块，将文件信息拷贝过去即可。

☆流程图：

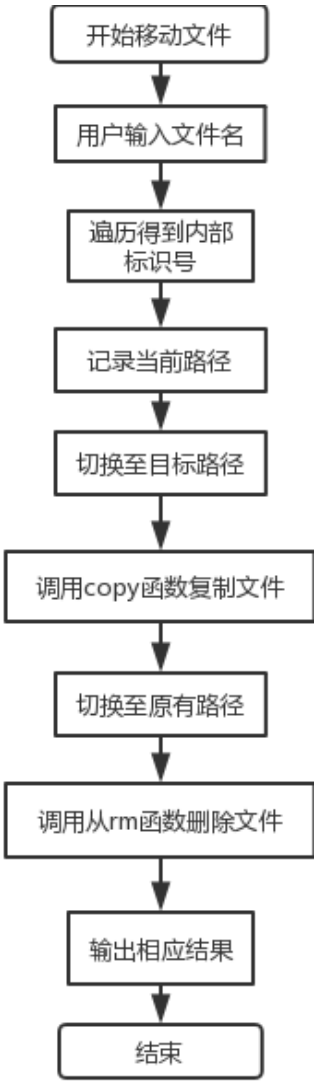


3.2.21 移动一个文件 设计

☆算法：

先保存当前路径。调用相关函数复制文件至指定路径，调用相关函数删除本目录下的文件。并更改相关路径以及文件信息。

☆流程图：

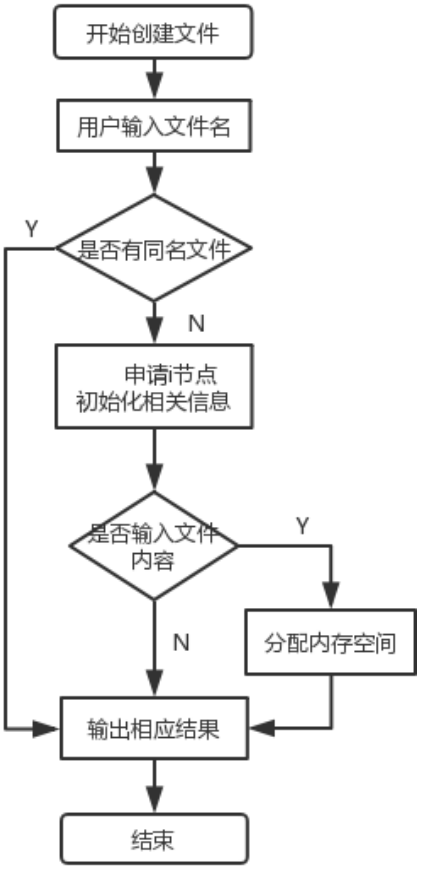


3.2.22 创建一个文件 设计

☆算法：

输入文件名后，检查在当前目录下是否含同名文件。如果没有，则申请 i 节点，初始化相关信息。并向用户询问是否需要输入文件内容，如果需要，则为文件分配内存空间，保存文件内容至数据块中。

☆流程图：

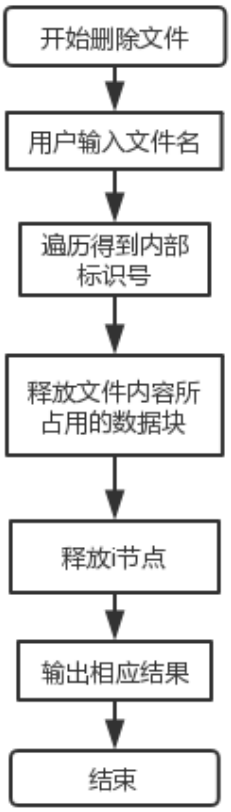


3.2.23 删除一个文件 设计

☆算法：

输入文件名后，遍历寻找内部标识号。如果寻找到相应文件，则释放文件内容所占用的数据块，释放其 i 节点。

☆流程图：

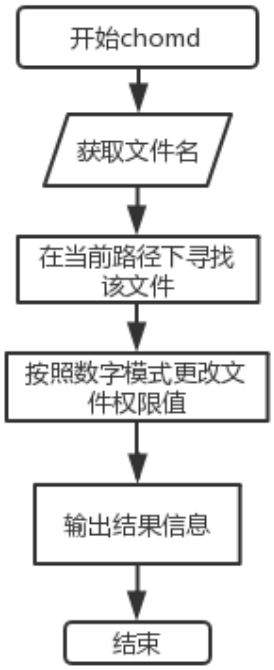


3.2.24 修改文件或目录的访问权限 设计

☆算法：

从地址缓冲区读取路径以及信号量，若无法找到文件，则报“文件不存在”否则读取文件索引节点，对权限进行检查，若权限足够，则修改目录节点块并写回。

☆流程图：



3.2.25 显示当前目录下所有文件目录属性 设计

☆算法：

从缓冲区读取信号量，读取当前目录索引节点，对权限进行检查，若权限足够，则显示当前目录下文件对于其他用户的所有属性信息。

☆流程图：

3.2.26 多线程 设计

（1）DWORD 句柄：（见程序 1）

利用 DWORD WINAPI 技术，在程序内部创建两个线程，一个和用户进行交互，一个用于进入后台维持秩序。在本系统的设计中，进程 1——接收并分析命令，进程 2——调用命令函数。此方式取代了以往 while 轮询方式，使得这两个进程可以独立运行。关键代码部分如下：

```
DWORD WINAPI ThreadFun1(LPVOID para);
DWORD WINAPI ThreadFun2(LPVOID para);


HANDLE mutual;
HANDLE signal1, signal2;
//进程 1--------分析命令
DWORD WINAPI ThreadFun1(LPVOID para)
{
    while (TRUE)
    {
        //WaitForSingleObject(signal1, INFINITE);
        WaitForSingleObject(mutual, INFINITE);
        Sleep(500);
        input();//输入命令
        ReleaseMutex(mutual);
        //ReleaseSemaphore(signal2, 1, NULL);
    }
    return 0;
}

//进程 2--------调用命令
DWORD WINAPI ThreadFun2(LPVOID para)
{
    while (TRUE)
    {
        //WaitForSingleObject(signal2, INFINITE);
        WaitForSingleObject(mutual, INFINITE);
        Sleep(500);
        choice();//命令匹配
        ReleaseMutex(mutual);
        //ReleaseSemaphore(signal1, 0, NULL);
    }
    return 0;
}

//主函数
int _tmain(int argc, _TCHAR* argv[])
{
    head();//系统头部
    login();//登录

    HANDLE thread1;
    HANDLE thread2;
```

```
    mutual = CreateMutex(NULL, 0, NULL);

    thread1 = CreateThread(NULL, 0, ThreadFun1, NULL, 0, NULL);
    thread2 = CreateThread(NULL, 0, ThreadFun2, NULL, 0, NULL);

    WaitForSingleObject(thread1, INFINITE);
    WaitForSingleObject(thread2, INFINITE);

    CloseHandle(thread1);
    CloseHandle(thread2);
    flush();//命令缓存
    system("pause");
}
```

（2）共享内存方式：（见程序 2）

本系统利用共享内存的方式实现了并行处理，这是一种进程间相互通信的机制。利用服务器端开辟内存空间，并为此内存空间设置锁变量（信号量）。在客户端调用 HANDLE 句柄接口，实现对共享区域的访问，但在读写文件的时候，会时刻检查信号量变化。此方式可以支持多个程序同时运行，共享同一片内存区域。

在服务器端创建开辟共享内存空间，并设置控制共享内存空间的指针。

```
// 创建共享文件句柄
HANDLE hMapFile = CreateFileMapping(
    INVALID_HANDLE_VALUE,    // 物理文件句柄
    NULL,    // 默认安全级别
    PAGE_READWRITE,    // 可读可写
    0,    // 高位文件大小
    BUF_SIZE,    // 低位文件大小
    L"ShareMemorySZHC"    // 共享内存名称
);

// 映射缓存区视图 , 得到指向共享内存的指针

int *lpBase = (int*)MapViewOfFile(
    hMapFile,            // 共享内存的句柄
    FILE_MAP_ALL_ACCESS, // 可读写许可
    0,
    0,
    BUF_SIZE
);
*lpBase = 0;
```

在客户端引入该内存区，并设置内存区信号量，用以文件操作时加以限制。

```
//文件映射是一种实现进程间单向或双向通信的机制。
//它允许两个或多个本地进程间相互通信。为了共享文件或内存，
//所有的进程必须使用相同的文件映射的名字或是句柄。
char szBuffer[] = "signal";
HANDLE hMapFile = CreateFileMapping(INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0, SIG_SIZE, (LPCWSTR)szBuffer);
int *signal = (int *)MapViewOfFile(hMapFile, FILE_MAP_ALL_ACCESS, 0, 0, SIG_SIZE);
```

```
//超级块写入文件
void WriteSuperBlock()
{
    while (*signal == 1) {
        Sleep(500);
    }
    fstream myfile("disk.dat", ios::in | ios::out | ios::binary);//打开文件
    if (!myfile.is_open())
    {
        printf("\nCan't open file %s.\n", "user.txt");
        printf("This filesystem not exist!\n");
        system("pause");
        exit(0);
    }

    myfile.write((char*)&superblock, 2400);
    myfile.close();
    *signal = 0;
}
```

注：本部分未介绍 cls、help 等基础命令；
本部分流程图仅限于算法设计思路；
本部分功能流程图省略了部分权限判断以及错误提示输出过程；

4 磁盘文件系统数据结构

4.1 主要数据结构

☆超级块：

SuperBlock 是为了管理磁盘空间而设置的数据结构，存放了磁盘空间的使用情况，实时记录磁盘空间的状况可使整个文件系统的框架和结构更加清晰，此数据结构为文件处理的基础。

```
typedef struct {
                int useddisk;//已用磁盘空间
                int remaindisk;//剩余磁盘空间
                int usedblock;//已用盘块数量
                int remainblock;//剩余盘块数量
                int usedinode;//已用 i 节点数量
                int remaininode;//剩余 i 节点数量
                //块点位示图（将盘块和 i 节点使用情况合为一个数组）
                int block_inodemap[blockcount + inodecount];
  }SuperBlock;
```

☆i 节点：

INode 是为了管理文件而设置的数据结构，存放了为管理文件所需要的有关信息（文件属性），其是文件存在的标志，对任何文件的数据索引以及各种操作都需要经过文件控制块。

```
typedef struct {
                int x;//文件内部标识符
                string name;//文件名
```

```
                int size;//文件大小
                string user_name;//文件所有者
                int type;//文件类型（0 为目录，1 为文件）
                int user_quanxian;//其他用户权限 (r=2,w=1)
                int size_num;//占用盘块数量 (0 为全部共享)
                int address[N];//随机存取存放地址（可不连续存储）
                int father;//父目录内部标识符
                time_t atime;//访问时间
                time_t mtime;//修改时间
                time_t ctime;//修改时间
    }INode;
```

☆用户信息：

User 是为了存储用户信息而设置的数据结构，其中包含用户名、加密密码、用户 UID。此数据结构将用户信息整体化，在用户设定、文件权限等操作中具有一定意义。

```
typedef struct {
                string name; //用户名
                int password; //密码(哈希加密)
                int UID;//用户组号（具有相同组号的具有相同权限）
    }User;
```

4.2 主要函数说明

系统设计的逻辑结构比较强，几乎所有的功能函数全部采用子函数调用的方式，这样大大优化了系统的框架、增强了系统可读性。但也因此，涉及函数较多，以下是实现具体功能的基本函数罗列：

```
void head();//系统头部
void cmdhead();//命令头部
void msg(const char* str);//输出统一格式
unsigned int BKDRHash(char* key, unsigned int len);//哈希处理密码
void star(char *key);//密码加*处理
int login();//用户登录
void useradd();//添加用户
int judge();//判断 yesno
int addfloat(int x, int y);//小数进位
void WriteSuperBlock();//超级块写入文件
void ReadSuperBlock();//文件读取超级块
void Writedata(int Index, char *content);//数据块写入文件
void Readdata(int Index, char *content);//文件读取数据块
void showsb();//输出磁盘信息
void init_sb();//初始化超级块
void WriteINode(int Index, INode &inode);//i 节点写入文件
void ReadINode(int Index, INode &inode);//文件读取 i 节点
void init_id();//初始化 i 节点
```

```
void init_root();//初始化根目录
void diskloading();//载入磁盘
void diskupdating();//更新磁盘
void format();//格式化磁盘
void Backup();//备份磁盘文件
void recovery();//恢复磁盘文件信息
void help();//系统支持命令功能介绍
void pwd();//显示当前路径(绝对路径)
int bianli(string name, int curdir1);//寻找符合条件的 i 节点
int UID(int curdir1);//判断组群用户(返回组群号)
void cd(char *curpath1);//切换当前路径(支持绝对路径和相对路径)
int chmod(string config, int quanxian);//更改当前目录下文件权限
int get_inode();//申请 i 节点
void free_inode(int i);//释放 i 节点
int get_block();//申请盘块
void free_block(int i);// 释放盘块
int mkdir(string name);//创建当前目录下子目录
void createfile(string config);//创建子文件
void time();//显示当前时间
void _time(time_t &m_time);//时间转化
void dir();//显示当前目录下的信息与文件
void exit();//退出系统命令
void more(string name);//显示当前路径下文件内容
int rm(string name, int mulu);//删除当前目录下某个文件
int rmdir(string name, int mulu);//删除当前路径下某个目录
void copy(string name, char *curpath1);//复制当前文件到另一路径
void rename(string name, string name1);//当前路径下文件或者目录重命名
void xcopy(string name, char *curpath1);//复制目录
void import(char *curpath1, int mulu);//从本地导入文件
void find(char *curpath1);//寻找文件中的一段字符
void  attrib(string name);//显示当前目录下文件权限
void input();//输入函数
void charstring(char *config);//char 转 string 类型
void charstring1(char *config);//char 转 string 类型
void intstring(char *config);//char 转 int 类型
void ver();//版本信息
void choice();//命令选择
void flush();//清空命令
int BF(char S[], char T[]);//BF 字符串匹配
void pexport(char *curpath1, char *curpath2);//从本地导出文件
```

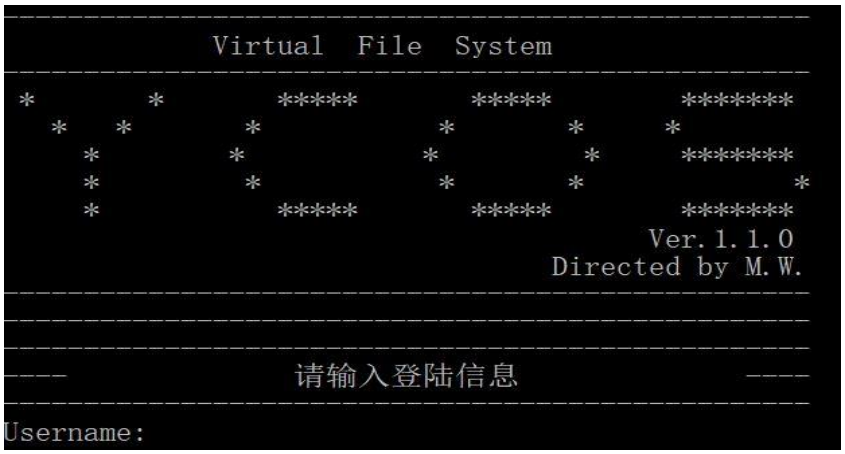
5 测试计划

5.1 测试说明

本部分主要测试功能命令的实现情况，基于不同命令特点制定不同的测试计划。

5.2 测试过程

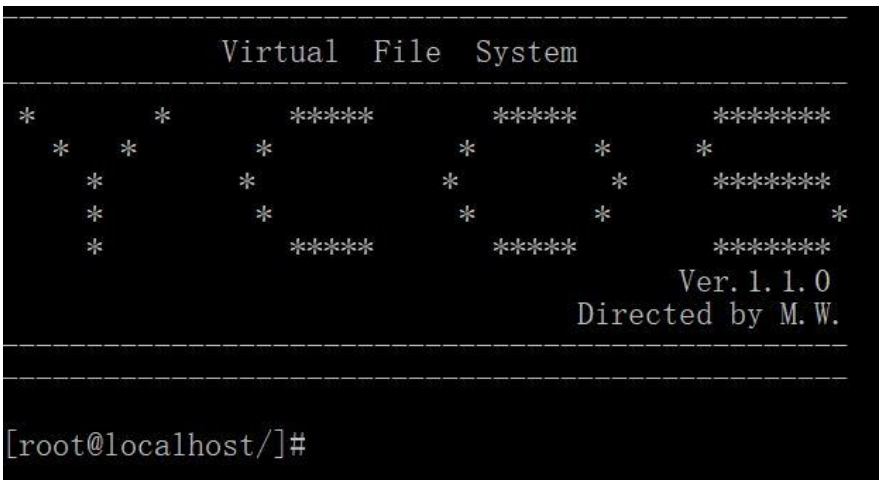
☆主界面☆



☆用户登录☆

☆cls☆

（清空屏幕，保留系统标识）



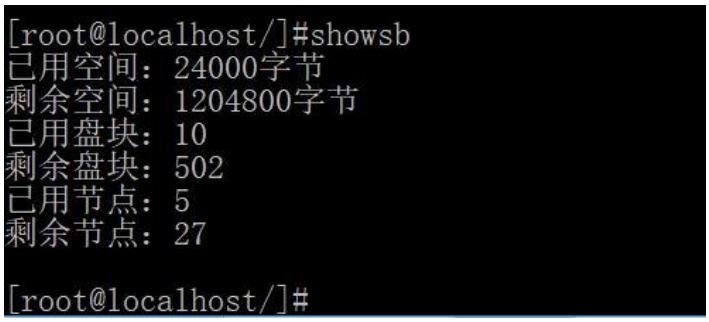
☆help☆

（列举命令操作符、命令格式、作用介绍）

```
[root@localhost/]#help
系统支持命令功能介绍:

(30+3条指令)

命令操作符      命令格式                作用介绍
cls             cls                     清除屏幕
help            help                    版本信息与帮助界面
login           login                   登录一个新的账户
useradd         useradd                 添加一个新的账户
exit            exit                    结束本次服务
format          format                  格式化虚拟磁盘
showsb          showsb                  显示磁盘使用情况
time            time                    显示当前系统时间
ver             ver                     显示当前系统版本
pwd             pwd                     显示当前路径
mkdir           mkdir dirname           创建一个新目录
dir\ls          dir\ls                  显示当前目录下的信息与文件
cd              cd path                 切换路径
mkfile          mkfile filename         创建一个新文件
rmdir           rmdir dirname           删除当前路径下一个目录（递归）
rm\del          rmdir\del filename      删除当前路径下一个文件
cat\more        cat\more filename       显示文件内容
rename          rename name             重命名一个文件或目录
chmod           chmod name mode         修改文件或目录的访问权限
attrib          attrib                  显示当前目录下所有文件目录属性
find            find txt                寻找包含指定文本的文件
copy            copy filename path      将一个文件拷贝到另一目录
xcopy           xcopy dirname path      将一个目录拷贝到另一目录（递归）
export          export filename path    将虚拟磁盘内容导出到物理磁盘
import          import path             将物理磁盘导入到虚拟磁盘内容
backup          backup                  备份虚拟磁盘内容
recover         recover                 恢复虚拟磁盘内容
diskupdate      diskupdate              更新虚拟磁盘
move            move filename path      移动文件至某个路径
movedir         movedir dirname path    移动目录至某个路径

[root@localhost/]#
```

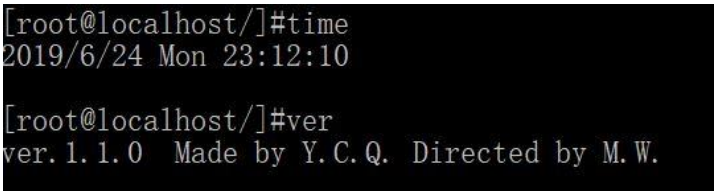
☆showsb☆

（显示磁盘空间使用状况）



☆time||ver☆

（列举系统时间、版本号）

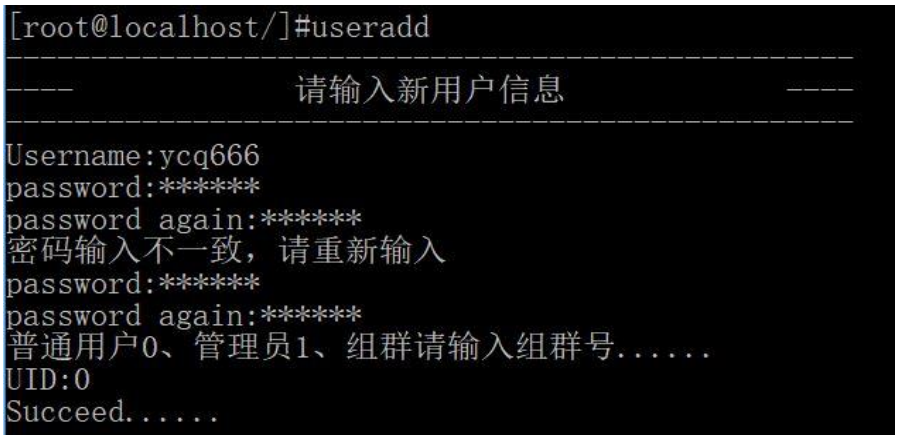


☆format☆

（格式化磁盘，只保存根目录）

☆useradd☆

（添加新用户信息，两次密码不一致报错）



☆mkdir☆

（在根目录下创建三个文件夹 a,b,c）

```
[root@localhost/]#mkdir a
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
-----------------------------------------------

[root@localhost/]#mkdir b
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
-----------------------------------------------
```

```
[root@localhost/]#mkdir c
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
-----------------------------------------------
```

☆cd☆

（进入 a 文件夹//绝对路径，从 a 文件夹返回根目录//相对路径）

```
[root@localhost/a]#cd .

[root@localhost/]#cd /a

[root@localhost/a]#
```

☆pwd☆

（显示当前路径信息）

```
[root@localhost/a]#pwd
当前路径: /a
```

☆mkfile☆

（在 a 文件中创建一个 ycq 文件，包含 9 字节文件信息）

```
[root@localhost/a]#mkfile ycq
Do you want to enter the contents of the file?(Y\N)y
Please enter the contents of the file:I Love OS
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
-----------------------------------------------
```

☆dir☆

（切换路径韦根目录，列举根目录下子目录或子文件信息）

```
[root@localhost/a]#cd .

[root@localhost/]#dir
2019/06/24  23:24:55      root      <DIR>     9        a
2019/06/24  23:24:59      root      <DIR>     0        b
2019/06/24  23:25:08      root      <DIR>     0        c

[root@localhost/]#
```

☆backup☆

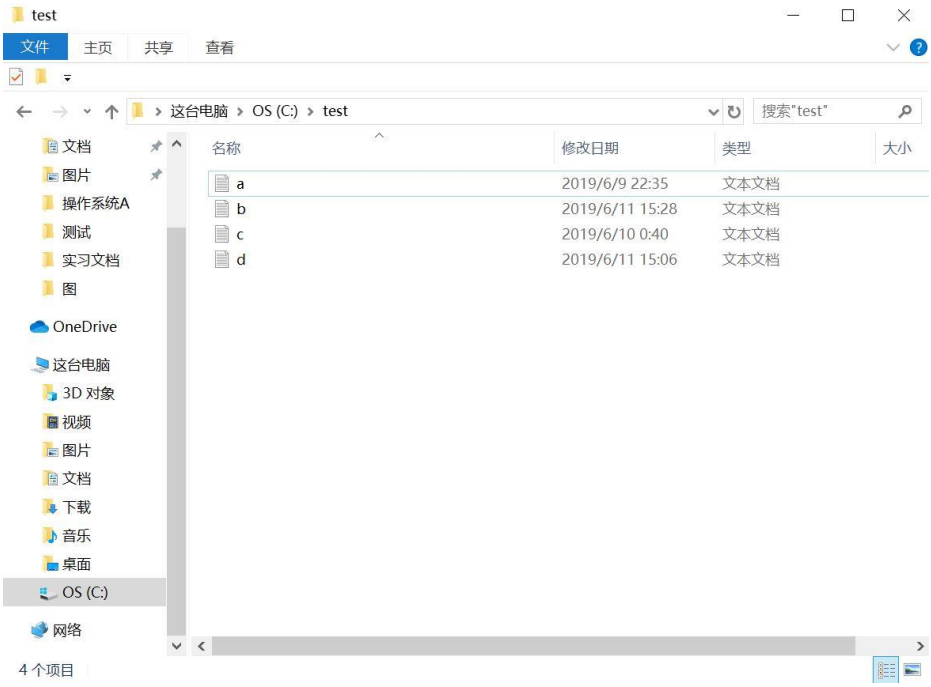
（磁盘备份）

```
[root@localhost/]#backup
备份完成
```

☆export☆

（建立虚拟磁盘文件 ycq1.txt，将其导入 C：\\test）

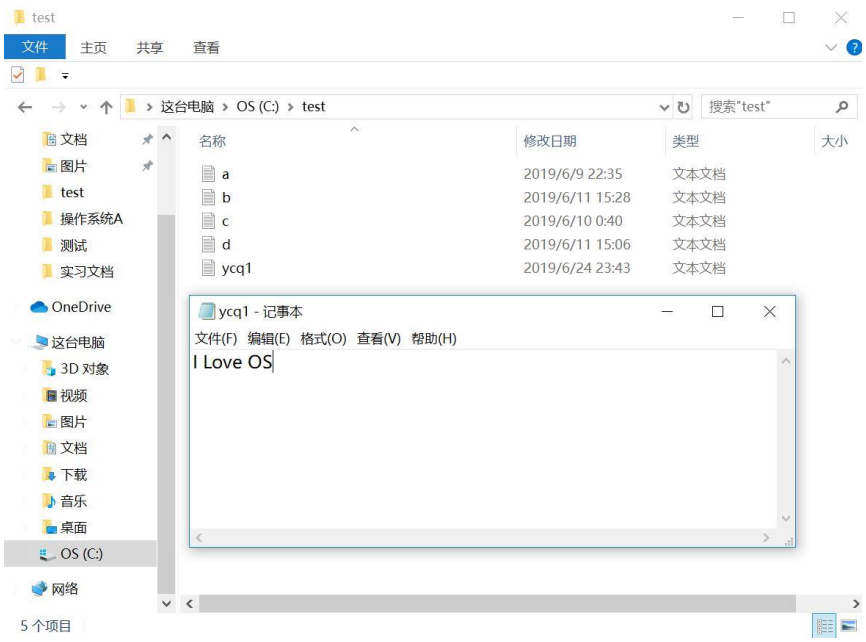
（原 test 目录下文件）



（建立文件 ycq1.txt）

```
[root@localhost/b]#mkfile ycq1.txt
Do you want to enter the contents of the file?(Y\N)y
Please enter the contents of the file:I Love OS
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
------------------------------------------------

[root@localhost/b]#export ycq1.txt C:\\test
```

（导入文件）



( )

☆rm☆

（删除 ycq1.txt 文件）

```
[root@localhost/]#cd /b

[root@localhost/b]#dir
2019/06/24  23:43:12      root              18       ycq1.txt

[root@localhost/b]#rm ycq1.txt
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
------------------------------------------------

[root@localhost/b]#dir

[root@localhost/b]#
```

☆rmdir☆

（删除 a 文件夹）

```
[root@localhost/]#dir
2019/06/10  18:34:00      root      <DIR>     9        a
2019/06/10  18:34:05      root      <DIR>     0        b
2019/06/10  18:34:09      root      <DIR>     0        c

[root@localhost/]#rmdir a
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
------------------------------------------------

Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
------------------------------------------------

[root@localhost/]#dir
2019/06/10  18:34:05      root      <DIR>     0        b
2019/06/10  18:34:09      root      <DIR>     0        c

[root@localhost/]#
```

☆import☆

（将 C:\\test\\a.txt 导入虚拟磁盘目录 b 下）

```
[root@localhost/]#dir
2019/06/10  18:34:05      root      <DIR>     0        b
2019/06/10  18:34:09      root      <DIR>     0        c

[root@localhost/]#cd /b

[root@localhost/b]#import C:\\test\\a.txt
succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
--------------------------------------------------

[root@localhost/b]#dir
2019/06/25  00:03:51      root              6        a.txt

[root@localhost/b]#
```

☆more☆

（显示 a.txt 文件内容）

```
[root@localhost/b]#more a.txt
123456

[root@localhost/b]#
```

☆xcopy☆

（此步骤省略了恢复磁盘步骤，下文会有详细介绍）

（将根目录下 a 目录递归复制到 b 目录下）

```
[root@localhost/a]#cd .

[root@localhost/]#dir
2019/06/25  00:13:38      root      <DIR>     9        a
2019/06/25  00:13:41      root      <DIR>     0        b
2019/06/25  00:13:44      root      <DIR>     0        c
```

```
[root@localhost/]#xcopy a /b
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
--------------------------------------------------

succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
--------------------------------------------------

succeed.....
Please Waiting ,Checking disk integrity...
Please Waiting ,Reading system data...
--------------------------------------------------
```

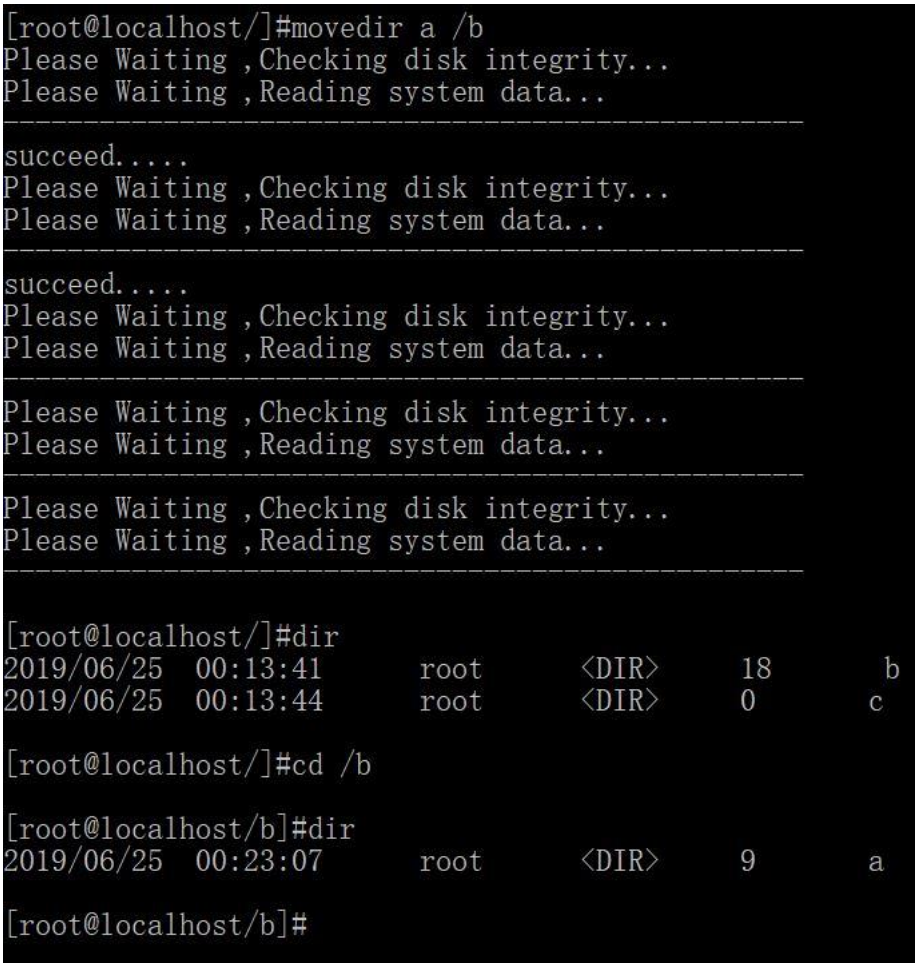
```
[root@localhost/b]#dir
2019/06/25  00:15:00      root      <DIR>     9        a

[root@localhost/b]#
```

☆recover||diskupdate☆

（恢复磁盘并更新）

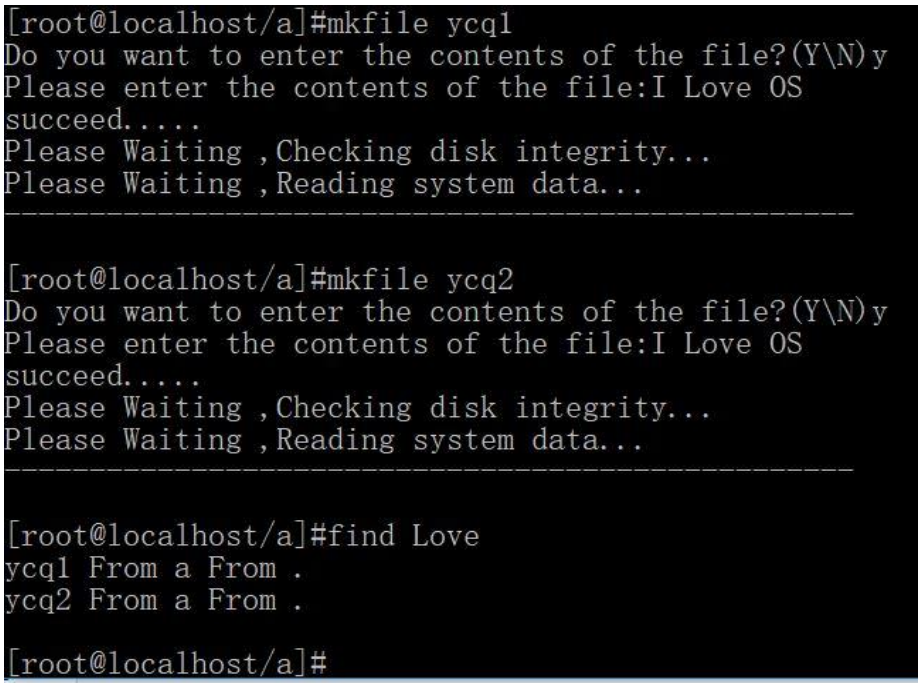
☆movedir☆

（将根目录下 a 目录递归移动到 b 目录下）



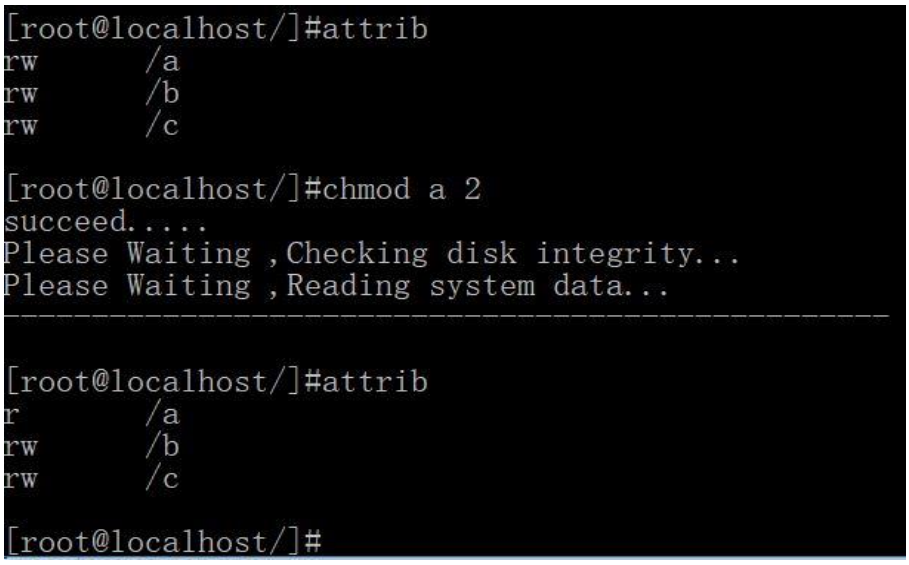
☆find☆

（在整个磁盘区域寻找包含自定内容的文件）



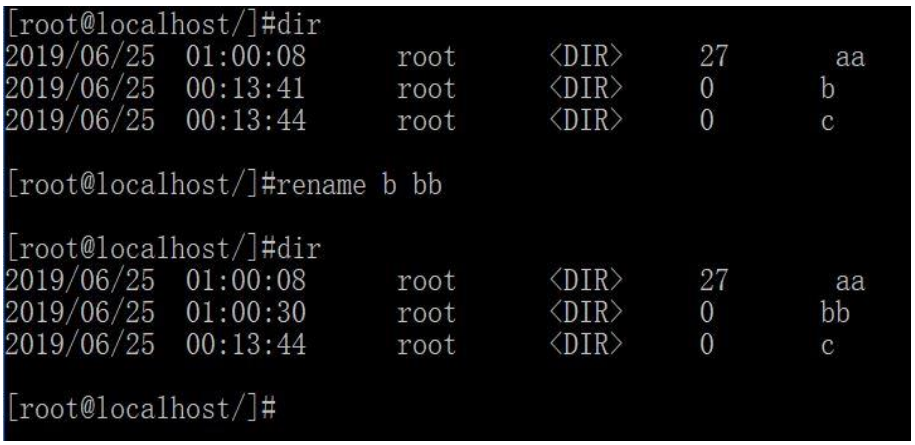
☆attrib|| chmod☆

（显示当前路径下文件属性，并更改文件属性）



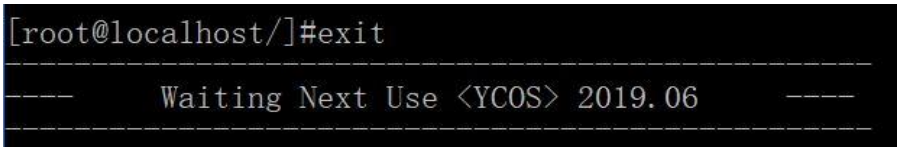
☆rename☆

（将 a，b 文件夹重命名 aa，bb）



☆exit☆

（退出系统）



5.3 测试评价

系统可保证指令的正常执行，实现相关功能。但在多次执行系统操作的同时会出现不稳定的现象，系统的健壮性不足，鲁棒性有待提升。由于这是一个小型实验版交互式系统，在不考虑考虑空间与时间性能的情况下，测试结果较为可靠。

6 亮点分析

☆程序书写规范

系统总计 2400 余行代码，条理清晰，结构合理。所有的功能函数均采用子函数调用方式，既使功能函数更加简单明了，也解决了部分代码重复使用占用内存所产生的冗余情况。代码注释清晰，增加其可读性，为二次开发奠定基础。

☆较完善的界面设计

为强化系统特色，专门设计系统指定 logo，并增加类 Linux 的命令行输入引导，同时嵌入用户名以及当前路径信息。系统设计了大量的指导语句，使得每一步功能的执行都都可得到相应提示与反馈。

☆多用户登陆系统

系统支持多用户登录，将用户相关信息保存至文件中。系统针对用户实现组群管理方式，将系统用户分为系统管理员，同组用户，其他用户三种身份。管理员在创建用户时可指定用户组群。在密码输入时，系统实现密文和退格的处理。

☆密码加密处理

在本系统中，密码并不是直接放至文件中保存，而是经由哈希加密算法处理。同时在进行密码比对时，也需要进行解码处理。这样设计是为了增加系统安全性，即使密码文件被泄露，但由于代码的封闭性，用户密码依旧难以被轻易破解。

☆文件权限系统

每个文件会设置标志位来保存其他用户的访问权限。根据本系统设计，系统管理员身份具有全部权限，同组用户具有相同权限，而其他用户的权限需要自行指定。用户可以利用 chmod 命令使用数据模式加以更改。

☆支持多路径寻址

系统支持相对路径和绝对路径的寻址操作。在本系统中，绝对路径以“\”开头。此功能在切换路径时为用户提供多种选择。

☆增加系统命令数量

为增强系统的可操作性，本系统在原有任务要求基础上增加了若干基础命令。在使用过程中可以有更多体验感。

☆两种线程实现方式

（1）DWORD 句柄：（见程序 1）

利用 DWORD WINAPI 技术，在程序内部创建两个线程，一个和用户进行交互，一个用于进入后台维持秩序。在本系统的设计中，进程 1——接收并分析命令，进程 2——调用命令函数。此方式取代了以往 while 轮询方式，使得这两个进程可以独立运行。

（2）共享内存方式：（见程序 2）

本系统利用共享内存的方式实现了并行处理。利用服务器端开辟内存空间，并为此内存空间设置锁变量（信号量）。在客户端调用 HANDLE 句柄接口，实现对共享区域的访问，但在读写文件的时候，会时刻检查信号量变化。此方式可以支持多个程序同时运行，共享同一片内存区域。

7 结论分析

7.1 问题与解决办法

（1）文件系统结构的设计

俗话说的好：万事开头难。文件系统结构虽然课设的基础。但也是整个课设过程让我感到最艰难的地方。因为最初没有好的想法，自己便搜集了很多文献资料，想参考借鉴目前已成型的文件系统的设计方式。但是已有的可以使用的设计都很复杂，如果全部模仿实现对于我的难度较大。所以最后我选择了在 Linux 的文件系统基础上加以简化来完善我的文件系统结构的设计。

（2）存储空间划分不合理

在最初，经过精心计算，我顺利的完成了磁盘划分。开始的部分命令做的

也很顺利，但是随着命令难度的增加，存储空间划分不合理的问题逐渐暴露了出来。文件内容出现了覆盖叠加，使得整个磁盘空间结构混乱。后来经历了很多调试，增加了盘块大小，调整了超级块，i 节点的内存，修复了这个漏洞。

（3）递归拷贝目录调试

这个命令是困扰我最久的命令，最初调试成功后，便把它搁置一边。但是后来突然验证时发现又出现了堆栈溢出问题，接下来就是反复地调试，最后发现问题出现在 string 和 char 的转换上面，因为我的程序是 string 和 char 混写，而在有些转化上我没有处理好‘\0’的结束符，导致不匹配等一系列问题。

（4）特殊符号问题

VS 中有很多特殊符号不能直接的添加在文本中，这使得我的有些路径选择出现了问题。后来及时的发现，注意到转义字符的合理使用，顺利的解决了此类问题。

（5）find 文件内容匹配问题

这个问题是在答辩的时候老师提出来的，我虽然是全磁盘遍历，但是最初设置为遇到匹配便跳出循环，所以找不到多级相匹配的文件，在多次寻找下出现了混乱。后来加以更改，便可以实现相关功能了。

7.2 尚未解决的问题

①虽然进行了多次调试，但是递归目录拷贝后 dir 显示的文件大小依旧有错误，应该是在递归时文件大小累加的地方出现了逻辑混乱。但是根据我的系统设计以及函数调用方式，这里并不影响系统内存的分配以及相关功能的使用，由于时间问题在多次失败后自己也就没有继续调试。以后在时间充裕的情况下，自己也会再进行深度调试，相信会顺利解决这个问题。

②虽然已经开辟了共享内存空间，并加锁变量限制文件读写。但是共享信号量未能转化为原语操作（原子操作），接下来还需要进一步学习和了解如何使多个进程更好地共享内存空间。

③程序的健壮性还不够，部分命令在执行多次后会出现潜在问题，为增强系统的耐用性，还要对系统进行进一步的细节优化。很多已经实现的命令还有很多提升空间，目前也只是实现了初步功能。在实际的文件系统访问软件中，每一个命令都有具体的选项，不同的选项又会产生不同的结果，本系统未来还有很大的二次开发空间。

7.3 收获与感悟

每一次课设都是一次崭新的历练，在短短的几天时间内，收获要比付出更有意义和价值。即使在上学期已经经历过课设的种种考验，但是在本学期课设开始前，心中还是充满了未知的恐惧。课设开始后，这些恐惧也逐渐转化为前进的动力，希望自己最终可以不辜负这段时间的勤恳付出。

几天的课设，我从一个对文件系统理论都很懵懂的状态，到现在有较为清晰结构认识并可利用编程语言实现相关功能，从能看懂理论知识到能编译再到熟练运用。通过对文件系统的模拟，我加深了对其的理解，很多浮在表面的知识也变得具有实际操作的意义。

本次课设虽然只是模拟了操作系统中的一个小部分，但是也让我们看到操作系统开发的难度。未来的学习道路还很长，我也会更努力的走出舒适圈，尝试做更多有实践价值的事，把知识更灵活的运用。最后，真心感谢这一年来老师的鼓励与帮助，这是我在很多困难的处境下继续坚持下去的力量，自己也会继续脚踏实地的努力下去，相信在未来的日子里定会有更多的成长与收获。