|  |
| --- |
| 操作系统课程设计 |
|  |
|  |
| **计算机科学技术 082409 陈骥远** |
| **2011-9-14** |

目录

[一、 课程设计完成任务情况表 3](#_Toc303766765)

[1.1 小组成员名单： 3](#_Toc303766766)

[1.2 小组成员完成情况： 3](#_Toc303766767)

[1.3 个人课设得分认定 3](#_Toc303766768)

[二、 课程设计摘要 4](#_Toc303766769)

[三、 课程设计文档 4](#_Toc303766770)

[3.1 系统结构说明 4](#_Toc303766771)

[3.2 功能演示及代码分析 4](#_Toc303766772)

[3.2.1 显示当前目录 4](#_Toc303766773)

[3.2.2 创建文件夹 4](#_Toc303766774)

[3.2.3 创建文件 4](#_Toc303766775)

[3.2.4 删除文件 4](#_Toc303766776)

[3.2.5 删除文件夹 4](#_Toc303766777)

[3.2.6 进入指定文件夹 4](#_Toc303766778)

[3.2.7 返回上级目录 4](#_Toc303766779)

[3.2.8 打开文件 4](#_Toc303766780)

[3.2.9 关闭文件 4](#_Toc303766781)

[3.2.10 写入文件 4](#_Toc303766782)

[3.2.11 读出文件 4](#_Toc303766783)

[3.2.12 格式化 4](#_Toc303766784)

[3.2.13 退出程序 4](#_Toc303766785)

[四、 课程设计感言 4](#_Toc303766786)

# 课程设计完成任务情况表

## 1.1 小组成员名单：

082409陈骥远

## 1.2 小组成员完成情况：

|  |  |
| --- | --- |
| 课程设计项目 | 完成者 |
| 系统设计 | 陈骥远 100% |
| 程序编码 | 陈骥远 100% |
| 文档撰写 | 陈骥远 100% |
| 搜集资料 | 陈骥远 100% |

## 1.3 个人课设得分认定

由于数据库课程设计截止时间提前一周及我的个人原因，我没找到打算一起写文件系统的小组成员。本程序中除部分设计思路及文件存储格式参考《UNIX操作系统教程》（尤晋元，1985年）外，其他均是我个人设计及编码的结果。由于时间仓促，本程序在跨平台兼容性、字符集、可移植性、并发等情况存在很多尚未解决的问题，同时还有许多零碎的bug尚未解决。希望老师能在给分比例允许的情况下给我优，最好不要挂我，谢了。

**小组成员签名：**

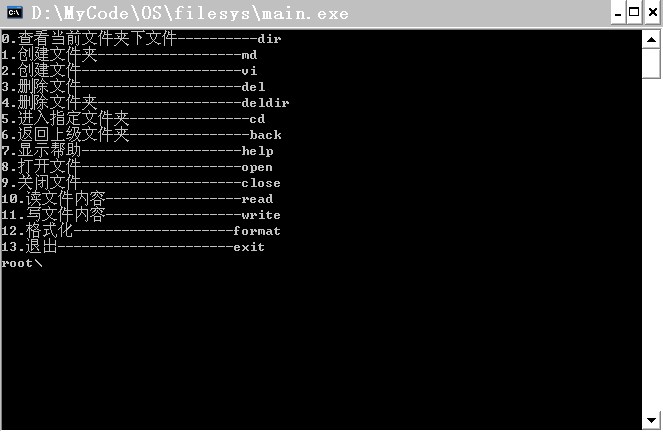
# 课程设计摘要

本课程设计项目是编写一个二级文件系统，本项目编写的二级文件系统实现了查看当前目录文件、创建文件夹、创建文件、删除文件夹、删除文件、进入指定文件夹、返回上级文件夹、打开文件、关闭文件、读文件、写文件、格式化文件系统等功能。

本程序使用C++语言实现，调用了stdio.h和string.h库，使用gnu c++4进行编译，使用fseek和fread/fwrite模拟磁盘读写操作。

本系统除filsys、file和inode节点使用UNIX v6代码中相关部分的简化版以外，其余部分均为我个人编写，未参考课本外其他教材。

下面是命令行的程序界面：



# 课程设计文档

## 3.1 系统结构说明

对于文件系统文件filevolum.sys文件，我把它按照104Byte划分成一个删去，删去第一块是superblock区，存放文件系统的标识数据，之后跟着200块是inode去，然后跟着若干块是数据文件。下图是文件系统示意图：

……

Datablock1

Inode200

……

Inode1

superblock

Superblock记录文件系统的所有信息，其结构如下：

struct filsys

{

unsigned short s\_isize; /\* inode数 \*/

unsigned short s\_fsize; /\* 最大容量 \*/

unsigned short s\_nfree; /\* 空闲块数 \*/

unsigned short s\_free[200]; /\* 空闲块（0~200） \*/

unsigned short s\_ninode; /\* 空闲inode数 \*/

unsigned short s\_inode[200]; /\* 空闲inode(0~200) \*/

};

该结构中的空闲盘块s\_free通过索引盘块进行勾连。在这里我简单介绍一下本程序的数据块情况。本程序数据块分为三种：索引块、目录块和数据块。数据块就是以字节形式存放的数据文件，1024byte全部都是数据；目录块每块最多包含32个目录记录，目录记录为32字节，前30字节为文件名或文件夹名，后2个字节为该文件或文件夹对应的inode编号。索引盘块是这样的：索引盘块最多包含100个盘块编号。当前剩余盘块数<=100的时候就存在当前索引盘块中，否则将索引盘块数据中的0#数据指针指向一个空的盘块，然后将剩余盘块数减100，存满当前盘块后将0#所指盘块和更新过的当前盘块数做为参数传递下去进入下一级盘块进行处理。Superblock的空闲盘块处理其实就是模拟了一个索引盘块把空闲盘块勾连起来。

Inode结构也是对Unix V6的inode结构的一个简化版，只保留本系统中有用信息。其具体结构如下。

struct inode

{

unsigned short i\_mode;

unsigned short i\_size;

unsigned short i\_addr[8]; /\* device addresses constituting file \*/

};

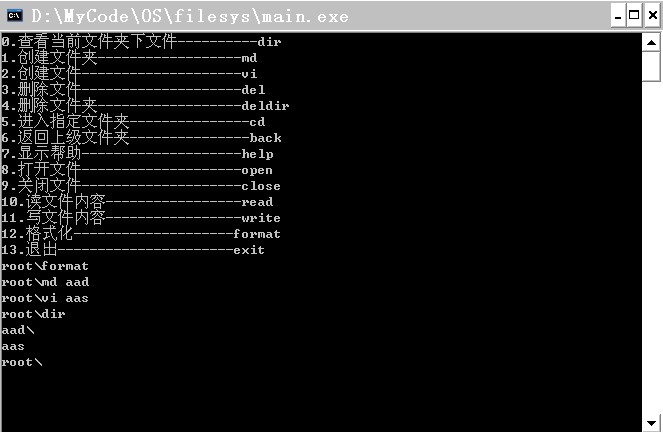
其中，i\_addr也是用之前讲的索引盘块的方法把文件勾连在一起的。文件分为目录文件和数据文件两种。在imode中表示。

## 3.2 功能演示及代码分析

### 3.2.1 显示当前目录

实现该功能的函数的实际操作是遍历当前所在目录文件，并将目录文件中所有在此之中的文件名、文件夹名输出。其具体命令为：dir

下图为该功能运行情况：



下面为实现该功能的代码：

void dir() //察看文件夹目录

{

int item=0,all=node[dirway[no\_dirway-1]].i\_size;

all=all/32;

int count=0;

while (item<all)

{

int u=32;

if (item+32>=all) {u=all-item;}

char buf[1024];

fseek(stream,(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024,0);

int offset=(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024;

fread(buf,1,sizeof(buf),stream);

for (int i=0;i<u;i++)

{

char record[32];

fseek(stream,offset,0);

fread(record,1,32,stream);

int node1=record[30]\*64+record[31];

int uu=(node[node1].i\_mode & 16);

if (uu==0)

{

printf("%s",record);

if ((node[node1].i\_mode & IDIR)!=0) {printf("\\");}

printf("\n");

offset+=32;

}

}

item+=u;

count++;

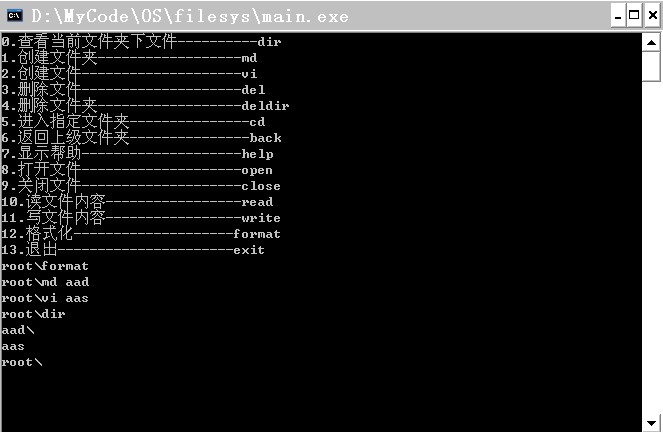
}

}

### 3.2.2 创建文件夹

创建文件夹函数实际是在当前目录结尾处增加一个文件夹（既是目录）的文件，然后写入删去。其格式为：md [name]。

下图为该功能运行情况：



下面为实现该功能的代码：

int md(char fname[]) //增加文件夹

{

if (namei(fname)!=4096) {return -1;}

superblock.s\_ninode--;

int unode=superblock.s\_inode[superblock.s\_ninode];

for (int i=0;i<8;i++) node[unode].i\_addr[i]=0;

node[unode].i\_size=0;

node[unode].i\_mode=IDIR;

ins\_node(dirway[no\_dirway-1],unode,fname);

}

其中namei为在目录文件中查找是否有该名字的文件，没有返回-1，其代码为：

unsigned short namei(char fname[]) //在当前目录下查找文件/文件夹

{

int item=0,all=node[dirway[no\_dirway-1]].i\_size;

all=all/32;

int count=0;

while (item<all)

{

int u=32;

if (item+32>=all) {u=all-item;}

char buf[1024];

fseek(stream,(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024,0);

int offset=(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024;

fread(buf,1,1024,stream);

for (int i=0;i<u;i++)

{

char record[32];

fseek(stream,offset,0);

fread(record,1,32,stream);

int len1,len2;

len1=strlen(fname);

len2=strlen(record);

if (len1==len2)

{

bool check=true;

for (int i=0;i<len1;i++)

if (record[i]!=fname[i]) {check=false;break;}

if (check) {return record[30]\*64+record[31];}

}

offset+=32;

}

item+=u;

count++;

}

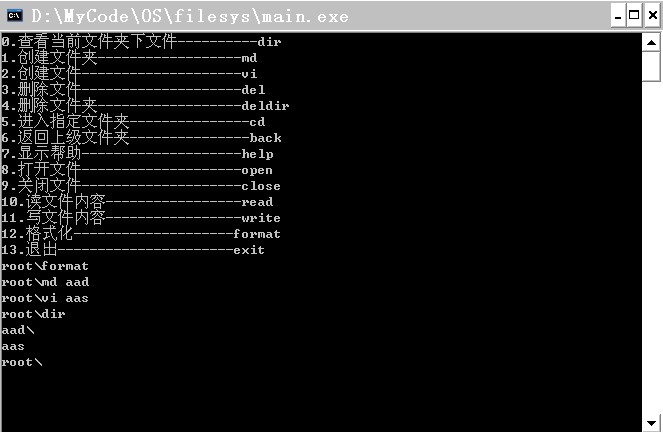
return 4096;

}

### 3.2.3 创建文件

创建文件与创建文件夹类似，只是其附加了文件权限的设置。其具体的格式如下：vi [name] [mode]。

下图为该功能运行情况：



下面为具体实现代码：

int vi(char\* fname,int mode) //增加文件

{

if (namei(fname)!=4096) {return -1;}

superblock.s\_ninode--;

int unode=superblock.s\_inode[superblock.s\_ninode];

for (int i=0;i<8;i++) node[unode].i\_addr[i]=0;

node[unode].i\_size=0;

node[unode].i\_mode = ( IFILE | mode );

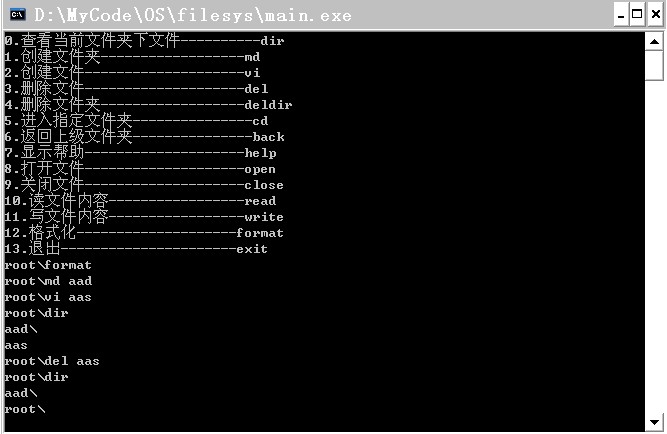
ins\_node(dirway[no\_dirway-1],unode,fname);

}

### 3.2.4 删除文件

删除文件其实是将当前目录中最后一项有效项记录填入被删除项记录，然后将被删除项的inode放入空闲inode表，并将其盘块挂到空闲盘块上去。具体命令格式为：del [name]。

下面为该功能运行情况：



下面为该功能运行代码：

void del(char\* fname) //删除文件

{

int node\_id=namei(fname);

if (node\_id<0) {printf("该目录下没有此文件！\n");}

superblock.s\_inode[superblock.s\_ninode]=node\_id;

superblock.s\_ninode++;

for (int i=0;i<8;i++)

if (node[node\_id].i\_addr[i]!=0) {set\_free\_block(node[node\_id].i\_addr[i]);}

node[node\_id].i\_mode+=IDELETE;

}

### 3.2.5 删除文件夹

该功能工作原理与删除文件类似，只是增加了一个遍历文件夹，将子文件夹递归处理，将子文件删除的功能。其具体实现代码如下：

void deldir(char\* deldir1) //删除文件夹

{

int node\_id=namei(deldir1);

if (node\_id==4096) {printf("该目录下没有此文件夹！\n");return;}

int item=0,all=node[dirway[no\_dirway-1]].i\_size;

all=all/32;

int count=0;

while (item<all)

{

int u=32;

if (item+32>=all) {u=all-item;}

char buf[1024];

fseek(stream,(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024,0);

int offset=(201+node[dirway[no\_dirway-1]].i\_addr[count])\*1024;

fread(buf,1,1024,stream);

for (int i=0;i<u;i++)

{

char record[32];

fseek(stream,offset,0);

fread(record,1,32,stream);

int len2=strlen(record);

char name[30];

for (int j=0;j<len2;j++) {name[j]=record[j];}

int id=record[30]\*64+record[31];

dirway[no\_dirway]=id;

no\_dirway++;

if ((node[id].i\_mode & IFILE)!=0) {del(name);}

else {deldir(name);}

no\_dirway--;

offset+=32;

}

item+=u;

count++;

}

for (int i=0;i<8;i++)

if (node[node\_id].i\_addr[i]!=0) {set\_free\_block(node[node\_id].i\_addr[i]);}

superblock.s\_inode[superblock.s\_ninode]=node\_id;

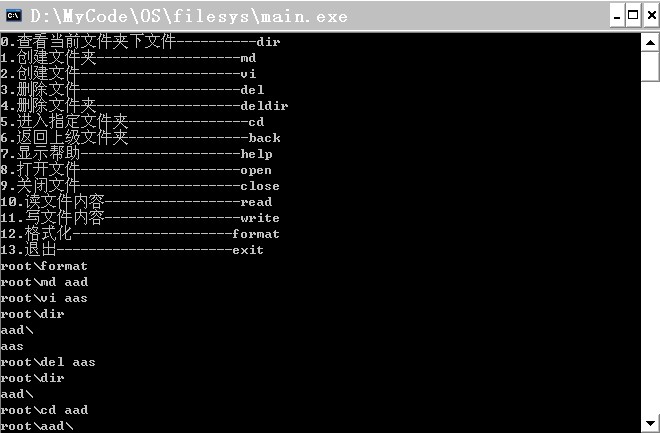
superblock.s\_ninode++;

}

### 3.2.6 进入指定文件夹

该功能其实就是从当前文件夹中进入下级文件夹，维护这个路径我们利用栈dirway和路径显示direction字符串。其具体命令格式为：cd [name]。

下图为其运行情况：



下面为其具体实现代码：

void cd(char fname[]) //进入下一层文件

{

int node\_id=namei(fname);

if (node\_id==4096) {printf("该目录下没有此文件夹！\n");return;}

int len1,len2;len1=strlen(direction);len2=strlen(fname);

for (int i=0;i<len2;i++) {direction[len1+i]=fname[i];}

direction[strlen(direction)]='\\';

dirway[no\_dirway]=node\_id;

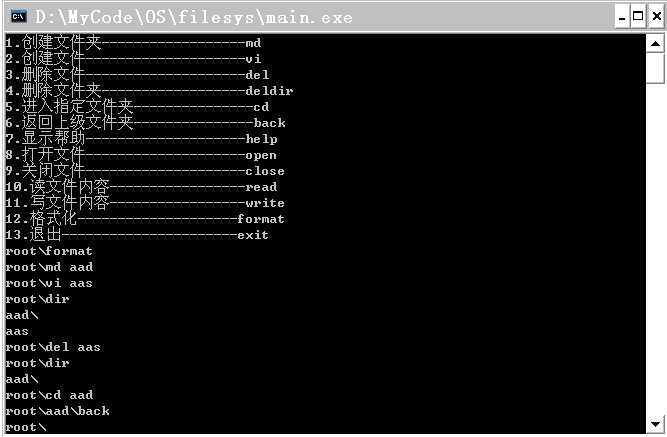
no\_dirway++;

}

### 3.2.7 返回上级目录

其具体维护还是使用了direction和dirway来实现。具体命令为：back。

下图为其具体实现情况：



下面为具体实现代码：

void back() //返回上一层

{

if (no\_dirway==0) {return;}

no\_dirway--;

int i=strlen(direction)-1;

i--;

while (direction[i]!='\\') {i--;}

direction[i+1]=0;

}

### 3.2.8 打开文件

该功能其实就是将一个文件载入打开文件结构并返回其编号。打开文件结构file如下：

struct file

{

unsigned short f\_inode; /\* pointer to inode structure \*/

unsigned short f\_offset[2]; /\* read/write character pointer \*/

};

该功能具体命令为：open [name]。

下图为其具体实现情况：



下面为其具体实现代码：

int open(char\* fname) //打开文件

{

int id=namei(fname);

if (id<0) {printf("该目录下没有此文件！\n");return -1;}

bool bo=false;int i;

for (i=0;i<no\_ofile;i++)

if (ofile[i].f\_inode==4096) {bo=true;break;}

if (!bo) {printf("打开文件数过多！\n");return -1;}

ofile[i].f\_inode=id;

return i;

}

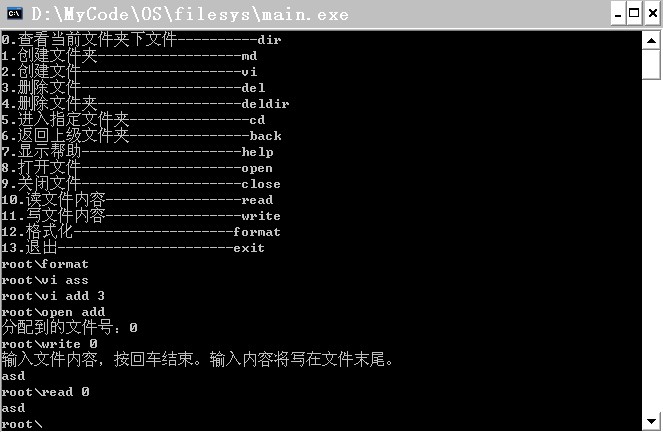
### 3.2.9 关闭文件

该功能只是将打开文件结构中对应编号文件关闭，其命令格式为：close [no]。

### 3.2.10 写入文件

该功能将新的内容写在文件末尾，当前盘块不够的话申请空闲或新的盘块进来使用。其命令具体格式为：write [no]。

下图为其具体运行情况：



下面为其具体实现代码：

void write(int id) //写入文件

{

if (ofile[id].f\_inode==4096) {printf("该编号没有打开文件\n");return;}

if ((node[ofile[id].f\_inode].i\_mode & IWRITE)==0) {printf("没有写权限\n");return;}

printf("输入文件内容，按回车结束。输入内容将写在文件末尾。\n");

char buf[1000];

gets(buf);

int len=strlen(buf);

int nodei=ofile[id].f\_inode;

int point=0;

if (node[nodei].i\_size+len>=8\*1024) {printf("文件过大!\n");}

{

int u=node[nodei].i\_size/1024;

if (node[nodei].i\_addr[u]==4096) {node[nodei].i\_addr[u]=get\_free\_block();}

int v=node[nodei].i\_size%1024;

while ((point<len)&&(u<7))

{

fseek(stream,(201+node[nodei].i\_addr[u])\*1024+v,0);

fwrite(&buf[point],1,1,stream);

v++;

point++;

if (v==1024)

{

u++;v=0;

if ((u!=8)&&(point<len)) node[nodei].i\_addr[u]=get\_free\_block();

}

}

}

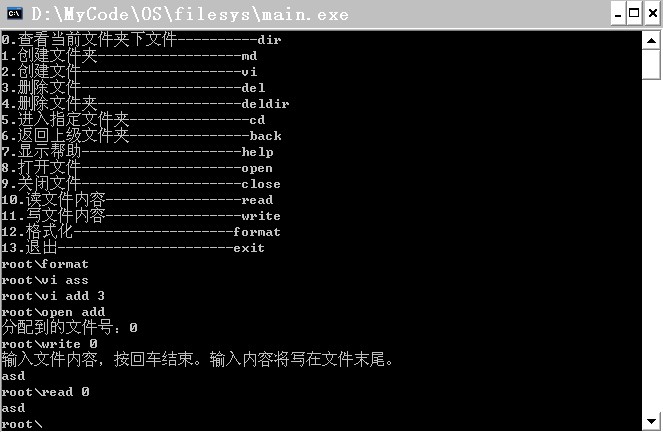
node[nodei].i\_size+=len;

}

### 3.2.11 读出文件

同样，读出文件就是按照文件大小将之前保存的数据文件读出并显示在屏幕上。其具体命令格式为：read [no]。

下图为其具体运行情况：



下面为其具体实现代码：

void read(int id) //读取文件

{

if (ofile[id].f\_inode<0) {printf("该编号没有打开文件\n");return;}

if ((node[ofile[id].f\_inode].i\_mode & IREAD)==0) {printf("没有读权限\n");return;}

int nodei=ofile[id].f\_inode;

int all=node[nodei].i\_size;

for (int i=0;i<7;i++)

{

int u=1024;

if (u>all) {u=all;}

if (u==0) {break;}

fseek(stream,(201+node[nodei].i\_addr[i])\*1024,0);

char buf[1024];

memset(buf,0,sizeof(0));

fread(buf,1,u,stream);

printf("%s",buf);

all-=u;

}

if (node[nodei].i\_size>8\*1024)

{

int cur\_block=node[nodei].i\_addr[7];

int u=node[nodei].i\_size-8\*1024;

while (true)

{

if (u<=0) {break;}

char buf[1024];

fseek(stream,(201+cur\_block)\*1024,0);

fread(buf,1,1024,stream);

if (u>99\*1024)

{

for (int i=0;i<99;i++)

{

char buf1[1024];

fseek(stream,(201+buf[(i+2)\*2]\*64+buf[(i+2)\*2+1])\*1024,0);

fread(buf1,1,1024,stream);

printf("%s",buf1);

}

} else

{

int i=0;

while (u>0)

{

int f=1024;

if (u<1024) {u=f;}

char buf1[1024];

fseek(stream,(201+buf[(i+2)\*2]\*64+buf[(i+2)\*2+1])\*1024,0);

fread(buf1,1,1024,stream);

printf("%s",buf1);

i++;

}

}

cur\_block=buf[2]\*64+buf[3];

u-=99\*1024;

}

}

printf("\n");

}

### 3.2.12 格式化

初始化文件系统。其命令具体格式为：format。

下图为其具体实现情况：



下面为其具体实现代码：

void format() //格式化

{

//fclose(stream);

stream=fopen("filevolume.sys","w");

superblock.s\_fsize=0;

superblock.s\_nfree=0;

for (int i=0;i<200;i++) {superblock.s\_free[i]=0;}

superblock.s\_isize=200;

superblock.s\_ninode=199;

for (int i=0;i<199;i++) {superblock.s\_inode[i]=i+1;}

char buf[1024];

memset(buf,0,sizeof(buf));

//memcpy(buf,&superblock,sizeof(filsys));

int uu=strlen(buf);

//for (int i=0;i<1024;i++) {fprintf(stream,"%c",buf[i]);}

fseek(stream,0,0);

fwrite(&superblock,2,sizeof(filsys)/2,stream);

node[0].i\_size=0;node[0].i\_mode=IDIR;

for (int i=0;i<8;i++) {node[0].i\_addr[i]=4096;}

memset(buf,0,sizeof(buf));

memcpy(buf,&node[0],sizeof(inode));

// for (int i=0;i<1024;i++) {fprintf(stream,"%c",buf[i]);}

fseek(stream,1024,0);

fwrite(buf,1,1024,stream);

node[0].i\_mode=0;

for (int i=0;i<8;i++) {node[0].i\_addr[i]=-1;}

memset(buf,0,sizeof(buf));

memcpy(buf,&node[0],sizeof(inode));

for (int i=0;i<199;i++)

{

//for (int i=0;i<1024;i++) {fprintf(stream,"%c",buf[i]);}

fseek(stream,(2+i)\*1024,0);

fwrite(buf,1,1024,stream);

node[i]=node[0];

}

node[199]=node[0];

fclose(stream);

memset(direction,0,sizeof(direction));

direction[0]='r';

direction[1]='o';

direction[2]='o';

direction[3]='t';

direction[4]='\\';

}

### 3.2.13 退出程序

其功能就是退出程序。

## 课程设计感言

这次课程设计我个人觉得最大的收获就是自己设计并实现了一个二级文件系统。我之前在暑假打算做一个U盘的文件系统，但是因为时间和个人水平问题最终没能做出来。目前的课程设计成果由于数据库课设时间提前和保研的事使得我做的比较仓促，并发控制部分完全没有设计，跨平台的东西也基本不能运行。若有时间我会对并行处理方面进行更深一步的探索。