FAT12 文件系统的仿真

学院名称：	数据科学与计算机学院
专业（班级）：	18计科教学3班
学生姓名:	夏一溥
学号:	18340178
时间:	2020 年 5 月 4 日
网课实验：	FAT12文件系统仿真

FAT12 简述

FAT12是DOS时代早期的文件系统，结构非常简单。

FAT12的基本组织单位:

1. 字节（Byte）： 基本数据单位
2. 扇区 （Sector）： 磁盘中最小的数据单元
3. 簇（Cluster）： 一个或多个扇区，由BPB表决定，在FAT12中为1簇=512字节=1扇区

FAT12文件系统由引导区BPB、FAT表、根目录项表和文件数据区组成:

扇区位置	长度（扇区）	内容
0	1	主引导记录
1	9	FAT1
10	9	FAT2
19	14	目录文件项
33	---	文件数据

主引导记录:

名称	开始字节	长度	内容	参考值
BS_jmpBOOT	0	3	一个短跳转指令	jmp short LABEL_STARTnop
BS_OEMName	3	8	厂商名	'ZGH'
BPB_BytesPerSec	11	2	每扇区字节数（Bytes/Sector）	0x200
BPB_SecPerClus	13	1	每簇扇区数（Sector/Cluster）	0x1
BPB_ResvdSecCnt	14	2	Boot记录占用多少扇区	ox1
BPB_NumFATs	16	1	共有多少FAT表	0x2
BPB_RootEntCnt	17	2	根目录区文件最大数	0xE0
BPB_TotSec16	19	2	扇区总数	0xB40
BPB_Media	21	1	介质描述符	0xF0
BPB_FATSz16	22	2	每个FAT表所占扇区数	0x9
BPB_SecPerTrk	24	2	每磁道扇区数（Sector/track）	0x12
BPB_NumHeads	26	2	磁头数（面数）	0x2
BPB_HiddSec	28	4	隐藏扇区数	0
BPB_TotSec32	32	4	如果BPB_TotSec16=0,则由这里给出扇区数	0
BS_DrvNum	36	1	INT 13H的驱动器号	0
BS_Reserved1	37	1	保留，未使用	0
BS_BootSig	38	1	扩展引导标记(29h)	0x29
BS_VolID	39	4	卷序列号	0
BS_VolLab	43	11	卷标	'ZGH'
BS_FileSysType	54	8	文件系统类型	'FAT12'

名称	开始字节	长度	内容	参考值
引导代码及其他内容	62	448	引导代码及其他数据	引导代码（剩余空间用0填充）
结束标志0xAA55	510	2	第510字节为0x55，第511字节为0xAA	0xAA55

目录项：

数据成员	偏移	长度	描述
DIR_Name	0x00	0x0B	文件名 8 字节，扩展名 3 字节
DIR_Attr	0x0B	0x01	文件属性
Reserve	0x0C	0x0A	保留位
DIR_WrtTime	0x16	0x02	最后一次写入时间
DIR_WrtDate	0x18	0x02	最后一次写入日期
DIR_FstClus	0x1A	0x02	文件开始的簇号
DIR_FileSize	0x1C	0x04	文件大小

实验内容

- [x] 模拟读取软盘映像，并展示其中文件内容
- [x] 打开文本文件
- [x] 创建新文本文件并编辑内容
- [x] 编辑文本文件内容，或重命名
- [x] 展示软盘映像目录结构

实验过程

代码实现：

数据结构：

BPB：//主引导记录

```
struct BPB
{
    u16 BPB_BytsPerSec; //每扇区字节数
    u8 BPB_SecPerClus;  //每簇扇区数
    u16 BPB_RsvdSecCnt; //Boot记录占用的扇区数
    u8 BPB_NumFATs;     //FAT表个数
    u16 BPB_RootEntCnt; //根目录最大文件数
    u16 BPB_TotSec16;
    u8 BPB_Media;
    u16 BPB_FATSz16; //FAT扇区数
    u16 BPB_SecPerTrk;
    u16 BPB_NumHeads;
```

```
    u32 BPB_HiddSec;
    u32 BPB_TotSec32; //如果BPB_FATSz16为0，该值为FAT扇区数
}; // size = 25Bytes
```

RootEntry：//根目录项

```
struct RootEntry
{
    char DIR_Name[11];
    u8 DIR_Attr; //文件属性
    char reserved[10];
    u16 DIR_WrtTime;
    u16 DIR_WrtDate;
    u16 DIR_FstClus; //开始簇号
    u32 DIR_FileSize;
}; // size = 32Bytes
```

fnode： //存储节点

```
struct fnode
{
    char rname[40] = { 0 };
    char fname[12] = { 0 };
    RootEntry fentry;
    void append(const char *ch) { strcat(fname, ch); }
    void rappend(const char *ch) { strcat(rname, ch); }
};
```

重点问题与解决方法：

- **如何通过根目录项访问，存储盘里所有文件信息**

考虑的方向是用数组将根目录项中的目录项储存，并用深度搜索遍历根目录项中所有的文件夹，并将遍历到的文件存储于另一个数组，于是在之后的数据处理中，可以通过顺序访问数组来遍历软盘映像中的全部文件。

处理根目录项：

```
void getRootFiles(FILE *fat12, fptr rootEntry_ptr) {
    int frbase = fileRootBase;
    for (int i = 0; i < RootEntCnt; i++)
    {
        fseek(fat12, frbase, SEEK_SET);
        fread(rootEntry_ptr, 1, 32, fat12);
        frbase += 32;
        if ((rootEntry_ptr->DIR_Name[0] == '\0') ||
(checkFile(rootEntry_ptr->DIR_Name, 0) == 0)) //过滤非法条目
            continue;
        fnode f;
        if ((rootEntry_ptr->DIR_Attr & 0x10) == 0) //此条目是文件
        {
            getFname(rootEntry_ptr->DIR_Name);
        }
        else //目录    则放进队列
        {
            getRname(rootEntry_ptr->DIR_Name);
```

```
            f.rappend("/");
        }
        f.append(fname_tmp);
        f.fentry = *rootEntry_ptr;
        froot.push_back(f);
    }
}
```

DFS：

```
    getRootFiles(fat12, rootEntry_ptr);
    for (auto ele : froot) {
        if ((ele.fentry.DIR_Attr & 0x10) == 0)
            continue;
        char tmp[50];
        memset(tmp, 0, sizeof(tmp));
        strcat(tmp, ele.rname);
        strcat(tmp, ele.fname);
        ftree.push_back(ele);
        dfs(fat12, ele, tmp);
    }
```

- 读取FAT表中的值：

由于FAT中的数据为12位，并不能通过常规的方法读取，需要对取得的2字节数据进行一定的处理，通过小尾顺序和FAT项结构可以的知，当 i 为偶数时——去掉高四位，i 为奇数——去掉低四位。

```
int getFATValue(FILE *fat12, int num) //3072
{
    u16 bytes;
    u16 *bytes_ptr = &bytes;
    fseek(fat12, fatBase + num * 3 / 2, SEEK_SET);
    fread(bytes_ptr, 1, 2, fat12);
    return (num & 1) ? (bytes >> 4) : (bytes & ((1 << 12) - 1));
}
```

- 判断数据区的结束：

FAT表中，当值等于0xFF7时，表示坏簇，当值大于0xFF7时，表示文件结束。于是可以用循环来判断是否读取文件完成：

```
while (curClus < 0xFF8)
    {
        if (curClus == 0xFF7)
        {
            printf("bad cluster,read failed\n");
            break;
        }
        //  operation should continue
    }
```

- 获取系统时间（日期）/计算文件时间（日期）

FAT12文件系统中，时间和日期由2字节（16位）存储：

```
u16 DIR_WrtTime;
u16 DIR_WrtDate;
```

其中，时间： 时/分/秒/ —— 5b/6b/保留5b/ ； 日期： 年/月/日/ —— 7b（+1980）/4b/5b/

于是，对时间于日期的操作仅仅就是位操作：

Time：

```
void showTime(unsigned short time) {
    u16 tm_min = 0b11111100000;
    u16 tm_hour = 0b1111100000000000;
    tm_min = ((tm_min&time) >> 5);
    tm_hour = ((tm_hour&time) >> 11);
    cout << setw(2) << tm_hour << ":" << setw(2) << tm_min;
}
```

Date：

```
void showDate(unsigned short date) {
    u16 year = 0b1111111000000000;
    u16 month = 0b111100000;
    u16 day = 0b11111;
    year = ((year&date) >> 9) + 1980;
    month = ((month&date) >> 5);
    day = (day & date);
    cout << year << "/" << right << setw(2) << setfill('0') << month << "/"
<< setw(2) << setfill('0') << day << "    ";
}
```

- 新文件的写入：

文件写入需要通过目录项-->数据区来完成操作，而在新创建文件的时候，我们是无法预先知道此文件的目录项信息的，所以需要改变下顺序。即 读入-->创建新目录项-->写入。

```
RootEntry WirteIntext(FILE *fat12); //读取输入内容，并返回由此内容创建的目录项
void getWritrEntry(FILE *fat12, string tname); //通过WirteIntext创建的目录项完成对软盘的写入
```

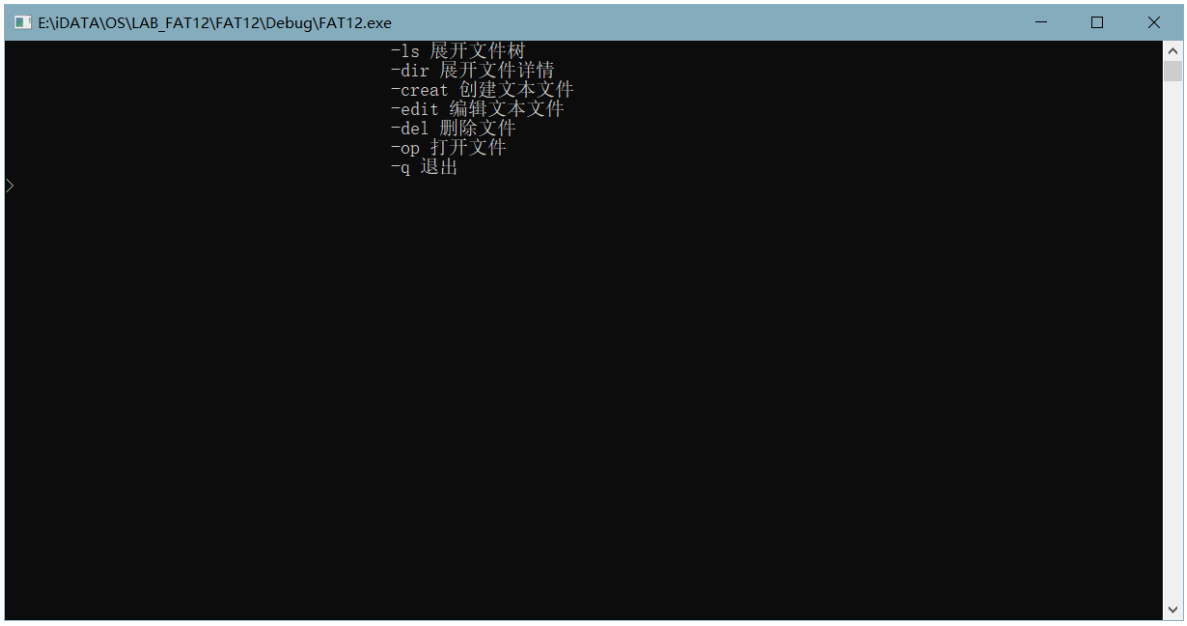
实验环境与配置

Windows 10家庭版

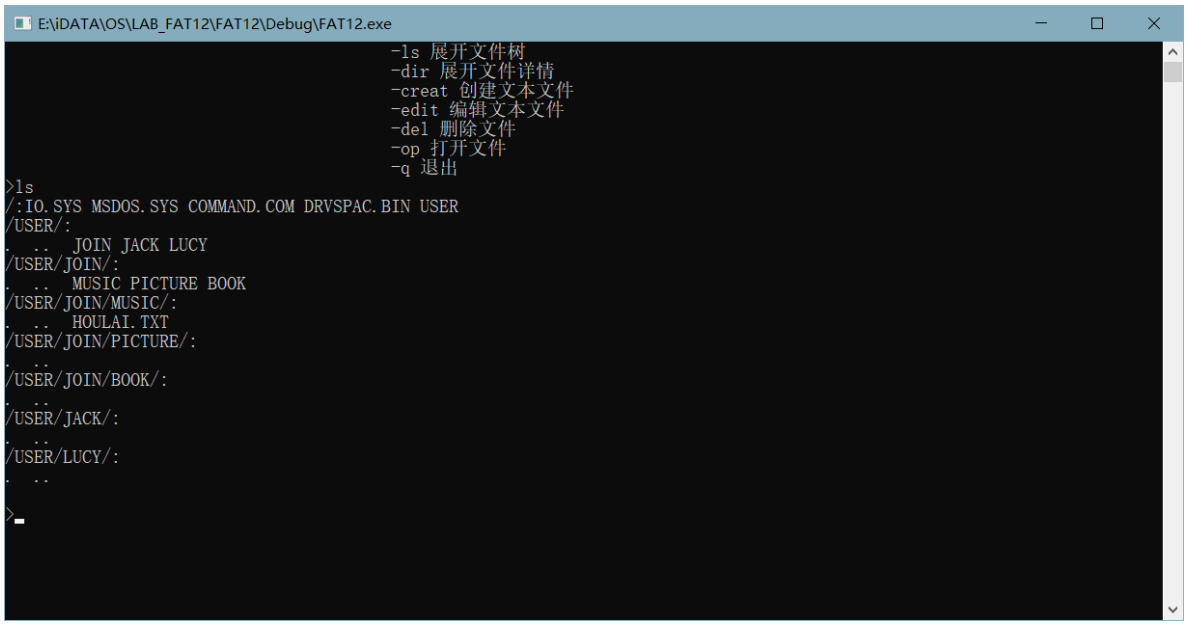
g++ -std=11

实验效果：

开始界面：



ls:



dir:

```
>dir /
name                          size                                date
IO.SYS                          40774                       1994/05/31    12:52
MSDOS.SYS                       38138                       1994/05/31    12:52
COMMAND.COM                     54645                       1994/05/31    12:52
DRVSPAC.BIN                     66294                       1994/05/31    12:52
USER                            0                           2020/03/10    20:17
```

creat:

```
>creat /HELLO.TXT
hello
world
^Z
```

```
>dir /
name                          size                                date
IO.SYS                          40774                       1994/05/31    12:52
MSDOS.SYS                       38138                       1994/05/31    12:52
COMMAND.COM                     54645                       1994/05/31    12:52
DRVSPAC.BIN                     66294                       1994/05/31    12:52
USER                            0                           2020/03/10    20:17
HELLO.TXT                       12                          2020/05/04    17:14
```

op:

```
>op HELLO.TXT
hello
world
```

del:

```
>del /HELLO.TXT

>dir /
name                        size                              date
IO.SYS                        40774                     1994/05/31    12:52
MSDOS.SYS                     38138                     1994/05/31    12:52
COMMAND.COM                   54645                     1994/05/31    12:52
DRVSPAC.BIN                   66294                     1994/05/31    12:52
USER                          0                         2020/03/10    20:17
```

```
>dir /
name                        size                              date
IO.SYS                        40774                     1994/05/31    12:52
MSDOS.SYS                     38138                     1994/05/31    12:52
COMMAND.COM                   54645                     1994/05/31    12:52
DRVSPAC.BIN                   66294                     1994/05/31    12:52
USER                          0                         2020/03/10    20:17
HELLO.TXT                     12                        2020/05/04    17:14

>del HELLO.TXT
The root rount is not found, please retype it.

>del /HELLO.TXT

>dir /
name                        size                              date
IO.SYS                        40774                     1994/05/31    12:52
MSDOS.SYS                     38138                     1994/05/31    12:52
COMMAND.COM                   54645                     1994/05/31    12:52
DRVSPAC.BIN                   66294                     1994/05/31    12:52
USER                          0                         2020/03/10    20:17

>_
```

edit:

```
>creat /TEST.TXT
18340178XYP
^Z

>dir /
name                        size                              date
IO.SYS                        40774                     1994/05/31    12:52
MSDOS.SYS                     38138                     1994/05/31    12:52
COMMAND.COM                   54645                     1994/05/31    12:52
DRVSPAC.BIN                   66294                     1994/05/31    12:52
USER                          0                         2020/03/10    20:17
TEST.TXT                      12                        2020/05/04    17:21

>edit TEST.TXT
HELLO COMPUTER OPERATION SYSTEM
^Z
Would you rename the file ? (EOF to giv up)
Enter the new name:
TEST2.TXT

>dir /
name                        size                              date
IO.SYS                        40774                     1994/05/31    12:52
MSDOS.SYS                     38138                     1994/05/31    12:52
COMMAND.COM                   54645                     1994/05/31    12:52
DRVSPAC.BIN                   66294                     1994/05/31    12:52
USER                          0                         2020/03/10    20:17
TEST2.TXT                     32                        2020/05/04    17:22

>op TEST2.TXT
HELLO COMPUTER OPERATION SYSTEM
>
```

总结：

这次实验给我最大的收获是清楚的理解掌握了FAT12文件系统存储文件的格式，并掌握了其中主引导记录BPB和目录项中各字节对应的内容。同时还复习了C语言对二进制文件的读写操作，文件流操作。

实验中我感到最困难的是开是构建整个文件系统的时候无从下手，因为没有方向所以一直在原地打转，不过经过老师的讲解和互联网上优质的博客内容，我一步一步建立了各个数据结构，确立了存储方式。当这些基础夯实后，后续的算法如DFS，经过上学期的训练则并不困难。

该项目还有许多不成熟的地方，比如

1. 许多地方未作错误输入的处理，即鲁棒性很低，因为输入错误过于百花齐放，我由于时间关系与不想让类似这般“补丁”的东西喧宾夺主，所以没有做得非常完善，于是当面向实用的时候还需要很大修改。

2. 文件名只能为11字节，超过后会出现显示问题。这一问题其实可以通过将超出长度存储于数据区解决，但还没来的及实现。
3. 使用数组预先存储所有文件，当文件数目很多的时候效率低下。

Reference：

1. https://blog.csdn.net/yxc135/article/details/8769086
2. https://blog.csdn.net/judyge/article/details/52373751
3. https://blog.csdn.net/qq_39654127/article/details/88429461#main-toc

附录：

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <stdlib.h>
#include <vector>
#include <string.h>
#include <ctime>
#include <cstdio>
#include <iomanip>
using namespace std;
typedef unsigned char u8;   //1字节
typedef unsigned short u16; //2字节
typedef unsigned int u32;   //4字节
int BytsPerSec;             //每扇区字节数
int SecPerClus;             //每簇扇区数
int RsvdSecCnt;             //Boot记录占用的扇区数
int NumFATs;                //FAT表个数
int RootEntCnt;             //根目录最大文件数
int FATSz;                  //FAT扇区数
int fatBase;                //Boot记录占用的扇区数*每扇区字节数=boot记录占用字节数
int fileRootBase;           //（Boot记录占用的扇区数+FAT表个数* FAT扇区数）*每扇区字节数
int dataBase;               //（Boot记录占用的扇区数+FAT表个数* FAT扇区数+（根目录最大文件数* 32+每扇区字节数-1）/每扇区字节数）*每扇区字节数
int BytsPerClus;
#pragma pack(1) /*指定按1字节对齐*/
struct BPB
{
    u16 BPB_BytsPerSec; //每扇区字节数
    u8 BPB_SecPerClus;  //每簇扇区数
    u16 BPB_RsvdSecCnt; //Boot记录占用的扇区数
    u8 BPB_NumFATs;     //FAT表个数
    u16 BPB_RootEntCnt; //根目录最大文件数
    u16 BPB_TotSec16;
    u8 BPB_Media;
    u16 BPB_FATSz16; //FAT扇区数
    u16 BPB_SecPerTrk;
    u16 BPB_NumHeads;
    u32 BPB_HiddSec;
```

```
    u32 BPB_TotSec32; //如果BPB_FATSz16为0，该值为FAT扇区数
}; // size = 25Bytes
//根目录条目
struct RootEntry
{
    char DIR_Name[11];
    u8 DIR_Attr; //文件属性
    char reserved[10];
    u16 DIR_WrtTime;
    u16 DIR_WrtDate;
    u16 DIR_FstClus; //开始簇号
    u32 DIR_FileSize;
}; // size = 32Bytes
#pragma pack() /*取消指定对齐，恢复缺省对齐*/
typedef struct RootEntry *fptr;
struct fnode
{
    char rname[40] = { 0 };
    char fname[12] = { 0 };
    RootEntry fentry;
    void append(const char *ch) { strcat(fname, ch); }
    void rappend(const char *ch) { strcat(rname, ch); }
};
int checkFile(char *, int);
void getFname(const char *);
void getRname(const char *);
unsigned short getTime();
unsigned short getDate();
void showTime(unsigned short);
void showDate(unsigned short);
// 写入生成文本文件目录项内容
RootEntry WirteIntext(FILE *fat12);
// 写入生成文本文件的目录项并返回此目录项地址
void getWritrEntry(FILE *fat12, string tname);
// 初始化根目录项
void getRootFiles(FILE *fat12, struct RootEntry *rootEntry_ptr);
int getFATValue(FILE *fat12, int num);
void dfs(FILE *fat12, fnode cur, char *rname);
void op(FILE *fat12, string tname);
void edit(FILE *fat12, string fname);
void del(FILE *fat12, string fname);
void dir(FILE *fat12, string rname);
void print(char *, int);
void ls(FILE *fat12);
char transletter(char x);
// 从BPB表初始化FAT信息
void ini(FILE* fat12);
char dot1[10] = ".  ..  ";
char dot2[10] = ".\n..\n";
char colon[2] = ":";
char tdata[100000];
char fname_tmp[13];
// 根目录文件队列
vector<fnode> froot;
// 文件树
vector<fnode> ftree;
void print(char *x, int a) {
    x[a] = '\0';
```

```
    printf("%s", x);
}
unsigned short getTime() {
    time_t nowtime;
    struct tm* p;;
    time(&nowtime);
    p = localtime(&nowtime);
    unsigned short ans = (p->tm_hour << 11) + (p->tm_min << 5);
    return ans;
}
void showTime(unsigned short time) {
    u16 tm_min = 0b11111100000;
    u16 tm_hour = 0b1111100000000000;
    tm_min = ((tm_min&time) >> 5);
    tm_hour = ((tm_hour&time) >> 11);
    cout << setw(2) << tm_hour << ":" << setw(2) << tm_min;
}
unsigned short getDate() {
    time_t nowtime;
    struct tm* p;;
    time(&nowtime);
    p = localtime(&nowtime);
    unsigned short year = p->tm_year - 80;
    unsigned short mon = p->tm_mon + 1;
    unsigned short ans = (year << 9) + (mon << 5) + p->tm_mday;
    return ans;
}
void showDate(unsigned short date) {
    u16 year = 0b1111111000000000;
    u16 month = 0b111100000;
    u16 day = 0b11111;
    year = ((year&date) >> 9) + 1980;
    month = ((month&date) >> 5);
    day = (day & date);
    cout << year << "/" << right << setw(2) << setfill('0') << month <<
"/" << setw(2) << setfill('0') << day << "    ";
}
void getTextData(FILE *fat12, RootEntry re)
{
    int curClus = re.DIR_FstClus;
    int startByte;
    int fsize = re.DIR_FileSize;
    while (curClus < 0xFF8)
    {
        if (curClus == 0xFF7)
        {

            printf("bad cluster,read failed\n");
            break;
        }
        startByte = dataBase + (curClus - 2) * BytsPerClus;
        fseek(fat12, startByte, SEEK_SET);
        if (fsize >= BytsPerClus) {
            fread(tdata, 1, BytsPerClus, fat12);
            fsize -= BytsPerClus;
        }
        else
            fread(tdata, 1, fsize, fat12);
```

```
        tdata[fsize - 1] = '\0';
        cout << tdata;
        curClus = getFATValue(fat12, curClus); //获取fat项的内容
    }
}
void getTextEntry(FILE *fat12, RootEntry re, string tname)
{
    int curClus = re.DIR_FstClus;
    int startByte;
    while (curClus < 0xFF8)
    {
        if (curClus == 0xFF7)
        {

            printf("bad cluster,read failed\n");
            break;
        }
        startByte = dataBase + (curClus - 2) * BytsPerClus;
        for (int loop = 0; loop < BytsPerClus; loop += 32)
        {
            RootEntry roottmp;
            fptr rootptr = &roottmp;
            fseek(fat12, startByte + loop, SEEK_SET);
            fread(rootptr, 1, 32, fat12);
            if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr-
>DIR_Name, 0) == 0) || (rootptr->DIR_Attr & 0x10) != 0)
                continue;
            getFname(rootptr->DIR_Name);
            string tmp = fname_tmp;
            if (tmp == tname)
            {
                getTextData(fat12, *rootptr);
                break;
            }
        }
        curClus = getFATValue(fat12, curClus); //获取fat项的内容
    }
}
int getFATValue(FILE *fat12, int num) //3072
{
    u16 bytes;
    u16 *bytes_ptr = &bytes;
    fseek(fat12, fatBase + num * 3 / 2, SEEK_SET);
    fread(bytes_ptr, 1, 2, fat12);
    return (num & 1) ? (bytes >> 4) : (bytes & ((1 << 12) - 1));
}
int checkFile(char *fname_tmp, int pos)
{
    for (int j = pos; j < pos + 11; j++)
    {
        if (!(((fname_tmp[j] >= 48) && (fname_tmp[j] <= 57)) ||
            ((fname_tmp[j] >= 65) && (fname_tmp[j] <= 90)) ||
            ((fname_tmp[j] >= 97) && (fname_tmp[j] <= 122)) ||
            (fname_tmp[j] == ' ')))
            return 0;
    }
    return 1;
}
```

```
void getRname(const char *dirname)
{
    int tmplen = 0;
    for (int k = 0; k < 12 && dirname[k] != ' '; k++)
        fname_tmp[tmplen++] = transletter(dirname[k]);
    fname_tmp[tmplen] = '\0';
}
void getFname(const char *dirname)
{
    int tmplen = 0;
    for (int k = 0; k < 11; k++)
    {
        if (dirname[k] != ' ')
            fname_tmp[tmplen++] = transletter(dirname[k]);
        else
        {
            fname_tmp[tmplen++] = '.';
            while ((dirname[k] == ' ') && k < 11) // 过滤空格
                k++;
            k--;
        }
    }
    fname_tmp[tmplen] = '\0';
}
char transletter(char x)
{
    if ((x <= 'z') && (x >= 'a'))
    {
        x -= 32;
    }
    return x;
}
void help() {
    cout << "                                        -ls 展开文件树" <<
endl;
    cout << "                                        -dir 展开文件详情" <<
endl;
    cout << "                                        -creat 创建文本文件" <<
endl;
    cout << "                                        -edit 编辑文本文件" <<
endl;
    cout << "                                        -del 删除文件" <<
endl;
    cout << "                                        -op 打开文件" << endl;
    cout << "                                        -q 退出" << endl;
}
int main() {
    FILE *fat12;
    fat12 = fopen("a.img", "r+");
    ini(fat12);
    help();
    printf(">");
    string cmd, tname;
    cin >> cmd ;
    while (1) {
        if (cmd == "q" || cmd == "Q") {
            cout << "Quit." << endl;
            return 0;
```

```
        }
        else if (cmd == "creat") {
            cin >> tname;
            froot.clear();
            ftree.clear();
            ini(fat12);
            getWritrEntry(fat12, tname);
            cin.clear();
            printf("\n>");
            cin >> cmd;
        }
        else if (cmd == "ls") {
            froot.clear();
            ftree.clear();
            ini(fat12);
            ls(fat12);
            printf("\n>");
            cin >> cmd;
        }
        else if (cmd == "dir") {
            cin >> tname;
            froot.clear();
            ftree.clear();
            ini(fat12);
            dir(fat12, tname);
            printf("\n>");
            cin >> cmd;
        }
        else if (cmd == "del") {
            cin >> tname;
            froot.clear();
            ftree.clear();
            ini(fat12);
            del(fat12, tname);
            printf("\n>");
            cin >> cmd;
        }
        else if (cmd == "edit") {
            cin >> tname;
            froot.clear();
            ftree.clear();
            ini(fat12);
            edit(fat12, tname);
            printf("\n>");
            cin >> cmd;
        }
        else if (cmd == "op") {
            cin >> tname;
            froot.clear();
            ftree.clear();
            ini(fat12);
            op(fat12, tname);
            printf("\n>");
            cin >> cmd;
        }
        else {
            cin.clear();
            printf("\n>");
```

```
            cin >> cmd;
        }
    }
    return 0;
}

void ini(FILE *fat12)
{
    struct BPB bpb;
    struct BPB *bpb_ptr = &bpb;    //载入BPB
    fseek(fat12, 11, SEEK_SET);    //BPB从偏移11个字节处开始
    fread(bpb_ptr, 1, 25, fat12); //BPB长度为25字节

    BytsPerSec = bpb_ptr->BPB_BytsPerSec; //初始化各个全局变量
    SecPerClus = bpb_ptr->BPB_SecPerClus;
    RsvdSecCnt = bpb_ptr->BPB_RsvdSecCnt;
    NumFATs = bpb_ptr->BPB_NumFATs;
    RootEntCnt = bpb_ptr->BPB_RootEntCnt;
    if (bpb_ptr->BPB_FATSz16 != 0)
        FATSz = bpb_ptr->BPB_FATSz16;
    else
        FATSz = bpb_ptr->BPB_TotSec32;
    fatBase = RsvdSecCnt * BytsPerSec;
    fileRootBase = (RsvdSecCnt + NumFATs * FATSz) * BytsPerSec; //根目录首字节的偏移数=boot+fat1&2的总字节数
    dataBase = BytsPerSec * (RsvdSecCnt + FATSz * NumFATs + (RootEntCnt * 32 + BytsPerSec - 1) / BytsPerSec);
    BytsPerClus = SecPerClus * BytsPerSec; //每簇的字节数
    struct RootEntry rootEntry;
    fptr rootEntry_ptr = &rootEntry;
    getRootFiles(fat12, rootEntry_ptr);
    for (auto ele : froot) {
        if ((ele.fentry.DIR_Attr & 0x10) == 0)
            continue;
        char tmp[50];
        memset(tmp, 0, sizeof(tmp));
        strcat(tmp, ele.rname);
        strcat(tmp, ele.fname);
        ftree.push_back(ele);
        dfs(fat12, ele, tmp);
    }
}
void getRootFiles(FILE *fat12, fptr rootEntry_ptr) {
    int frbase = fileRootBase;
    for (int i = 0; i < RootEntCnt; i++)
    {
        fseek(fat12, frbase, SEEK_SET);
        fread(rootEntry_ptr, 1, 32, fat12);
        frbase += 32;
        if ((rootEntry_ptr->DIR_Name[0] == '\0') || (checkFile(rootEntry_ptr->DIR_Name, 0) == 0)) //过滤非法条目
            continue;
        fnode f;
        if ((rootEntry_ptr->DIR_Attr & 0x10) == 0) //此条目是文件
        {
            getFname(rootEntry_ptr->DIR_Name);
        }
        else //目录    则放进队列
```

```
        {
            getRname(rootEntry_ptr->DIR_Name);
            f.rappend("/");
        }
        f.append(fname_tmp);
        f.fentry = *rootEntry_ptr;
        froot.push_back(f);
    }
}
void dfs(FILE *fat12, fnode cur, char *rname)
{
    int curClus = cur.fentry.DIR_FstClus;
    int startByte;
    while (curClus < 0xFF8)
    {
        if (curClus == 0xFF7)
        {
            printf("Bad cluster\n");
            break;
        }
        startByte = dataBase + (curClus - 2) * BytsPerClus;
        for (int loop = 0; loop < BytsPerClus; loop += 32)
        {
            RootEntry roottmp;
            fptr rootptr = &roottmp;
            fseek(fat12, startByte + loop, SEEK_SET);
            fread(rootptr, 1, 32, fat12);
            if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr->DIR_Name, 0) == 0) || (rootptr->DIR_Attr & 0x10) == 0)
                continue;
            getRname(rootptr->DIR_Name);
            fnode f;
            f.append(fname_tmp);
            f.rappend(rname);
            f.rappend("/");
            f.fentry = *rootptr;
            char tmp[50];
            memset(tmp, 0, sizeof(tmp));
            strcat(tmp, f.rname);
            strcat(tmp, f.fname);
            ftree.push_back(f);
            dfs(fat12, f, tmp);
        }
        curClus = getFATValue(fat12, curClus); //获取fat项的内容
    }
}
/*
 * 指令格式：creat /文件路径/文件名 （es. /USER/HOULAI.TXT；ps.根目录下为/HOULAI.TXT）
 * 指令功能：生成文本文件，并可编辑内容
 */
RootEntry WirteIntext(FILE *fat12) {
    int curClus;
    int startByte;
    RootEntry tpEntry;
    //fptr tpEntryptr = (fptr)malloc(sizeof(RootEntry));
    string tpLine;
    int fsize = 0;
```

```
    vector<string> tpText;
    getchar();
    while (getline(cin, tpLine)) {
        tpText.push_back(tpLine);
        fsize += tpLine.length();
        fsize++; // \n
    }
    tpEntry.DIR_FileSize = fsize;
    for (int i = 0; i < 3072; i++) {
        if (getFATValue(fat12, i) == 0) {
            int startByte = dataBase + (i - 2) * BytsPerClus;
            fseek(fat12, startByte, SEEK_SET);
            for (auto ele : tpText) {
                char* tp = (char*)malloc(ele.length() + 1);
                strcpy(tp, ele.c_str());
                fwrite(tp, strlen(tp), 1, fat12);
                free(tp);
                fwrite("\n", 1, 1, fat12);
            }

            tpEntry.DIR_FstClus = (short)i;
            tpEntry.DIR_WrtDate = (unsigned short)getDate();
            tpEntry.DIR_WrtTime = (unsigned short)getTime();
            tpEntry.DIR_Attr = (char)0;

            u16 bytes;
            u16 bytes2 = 0xff8;
            u16 *bytes_ptr = &bytes;
            fseek(fat12, fatBase + i * 3 / 2, SEEK_SET);
            fread(bytes_ptr, 1, 2, fat12);
            if (i & 1) {
                bytes2 = (bytes2 << 4) + (bytes & 15);
            }
            else {
                bytes2 = bytes2 + (bytes&(61440));
            }
            fseek(fat12, fatBase + i * 3 / 2, SEEK_SET);
            fwrite(&bytes2, sizeof(u16), 1, fat12);
            fseek(fat12, fatBase + i * 3 / 2 + FATSz * 512, SEEK_SET);
            fwrite(&bytes2, sizeof(u16), 1, fat12);
            //free(tpEntryptr);
            return tpEntry;
        }
    }
}
void getWritrEntry(FILE *fat12, string tname) {
    RootEntry newEntry;
    RootEntry tpEntry = WirteIntext(fat12);
    fptr newEptr = &newEntry;
    string sroot, stext;
    newEptr->DIR_Attr = 1;
    int pos = tname.rfind('/');//  /name.txt
    if (pos < 0) {
        printf("Root is not find,please retype it\n");
        return;
    }
    if (pos == 0) {
        sroot = "/";
```

```
        stext = tname.substr(pos + 1);
        int frbase = fileRootBase;
        fptr rootEntry_ptr = &newEntry;
        for (int i = 0; i < RootEntCnt; i++) {
            fseek(fat12, frbase, SEEK_SET);
            fread(rootEntry_ptr, 1, 32, fat12);
            frbase += 32;
            if ((rootEntry_ptr->DIR_Name[0] != '\0')) //过滤非法条目
                continue;
            pos = stext.rfind('.');
            string prefix, postfix;
            if (pos >= 8) {
                prefix = stext.substr(0, 8);
                postfix = stext.substr(pos + 1).substr(0, 3);
                stext = prefix + postfix;
            }
            else {
                prefix = stext.substr(0, pos);
                postfix = stext.substr(pos + 1).substr(0, 3);
                for (int i = 0; i <= 7 - pos; i++) {
                    prefix += ' ';
                }
                stext = prefix + postfix;
            }
            strcpy(rootEntry_ptr->DIR_Name, stext.c_str());
            rootEntry_ptr->DIR_Attr = 0x20;
            rootEntry_ptr->DIR_FileSize = tpEntry.DIR_FileSize;
            rootEntry_ptr->DIR_FstClus = tpEntry.DIR_FstClus;
            rootEntry_ptr->DIR_WrtDate = tpEntry.DIR_WrtDate;
            rootEntry_ptr->DIR_WrtTime = tpEntry.DIR_WrtTime;
            fseek(fat12, -32, SEEK_CUR);
            fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
            return;
        }

    }
    else {
        sroot = tname.substr(0, pos);
        stext = tname.substr(pos + 1);
        for (auto ele : ftree) {
            string x = ele.rname;
            x += ele.fname;
            if (x == sroot) {
                fptr rootEntry_ptr = &newEntry;
                int startByte = dataBase + (ele.fentry.DIR_FstClus - 2) *
BytsPerClus;
                fseek(fat12, startByte, SEEK_SET);
                fread(rootEntry_ptr, 1, 32, fat12);
                while (rootEntry_ptr->DIR_Name[0] != '\0') {
                    startByte += 32;
                    fseek(fat12, startByte, SEEK_SET);
                    fread(rootEntry_ptr, 1, 32, fat12);
                }
                pos = stext.rfind('.');
                string prefix, postfix;
                if (pos >= 8) {
                    prefix = stext.substr(0, 8);
                    postfix = stext.substr(pos + 1).substr(0, 3);
```

```
                    stext = prefix + postfix;
                }
                else {
                    prefix = stext.substr(0, pos);
                    postfix = stext.substr(pos + 1).substr(0, 3);
                    for (int i = 0; i <= 7 - pos; i++) {
                        prefix += ' ';
                    }
                    stext = prefix + postfix;
                }
                strcpy(rootEntry_ptr->DIR_Name, stext.c_str());
                rootEntry_ptr->DIR_Attr = 0x01;
                rootEntry_ptr->DIR_FileSize = tpEntry.DIR_FileSize;
                rootEntry_ptr->DIR_FstClus = tpEntry.DIR_FstClus;
                rootEntry_ptr->DIR_WrtDate = tpEntry.DIR_WrtDate;
                rootEntry_ptr->DIR_WrtTime = tpEntry.DIR_WrtTime;
                fseek(fat12, startByte, SEEK_SET);
                fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
            }
        }
    }
    return;
}
/*
 * 指令格式： ls
 * 指令功能： 展开文件树
 */
struct lsfile
{
    bool flag;
    char fname[12];
    void append(char *ch) { strcat(fname, ch); }
    void ini()
    {
        memset(fname, 0, sizeof(fname));
    }
    int fsize;
};
vector<lsfile> vcfile;
void Traverse(fnode cur, FILE *fat12)
{
    vcfile.clear();
    int curClus = cur.fentry.DIR_FstClus, startByte;
    while (curClus < 0xFF8)
    {
        if (curClus == 0xFF7)
        {
            printf("bad cluster,read failed\n");
            break;
        }
        startByte = dataBase + (curClus - 2) * BytsPerClus;
        for (int loop = 0; loop < BytsPerClus; loop += 32)
        {
            RootEntry roottmp;
            fptr rootptr = &roottmp;
            fseek(fat12, startByte + loop, SEEK_SET);
            fread(rootptr, 1, 32, fat12);
```

```
            if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr->DIR_Name, 0) == 0))
                continue;
            if ((rootptr->DIR_Attr & 0x10) == 0)
            {
                lsfile tpfile;
                tpfile.ini();
                tpfile.flag = 0;
                getFname(rootptr->DIR_Name);
                tpfile.append(fname_tmp);
                tpfile.fsize = rootptr->DIR_FileSize;
                vcfile.push_back(tpfile);
            }
            else
            {
                lsfile tpfile;
                tpfile.ini();
                tpfile.flag = 1;
                getRname(rootptr->DIR_Name);
                tpfile.append(fname_tmp);
                vcfile.push_back(tpfile);
            }
        }
        curClus = getFATValue(fat12, curClus);
    }
    return;
}
void ls(FILE *fat12) {
    int mode;
    printf("/:");
    for (auto ele : froot)
    {
        print(ele.fname,strlen(ele.fname));
        printf(" ");
    }
    printf("\n");

    for (auto ele : ftree)
    {
        Traverse(ele, fat12);
        print(ele.rname, strlen(ele.rname));
        print(ele.fname, strlen(ele.fname));
        printf("/:");
        printf("\n");
        printf("%s", dot1);
        for (auto obj : vcfile)
        {
            print(obj.fname, strlen(obj.fname));
            printf(" ");
        }
        printf("\n");
    }
}
/*
 * 指令格式：dir /文件路径 （es. /USER/；ps.根目录下为/）
 * 指令功能：展示指定路径下文件项
 */
void dir(FILE *fat12, string rname) {
```

```
661     int flag = 0;
662     RootEntry newEntry;
663     fptr rootEntry_ptr = &newEntry;
664     if (rname == "/") {
665         cout << left << setw(20) <<setfill(' ')<< "name" << right <<
    setw(15) << "size" << setw(39) << "date" << endl;
666         for (auto ele : froot) {
667             rootEntry_ptr = &ele.fentry;
668             if (rootEntry_ptr->DIR_Name[0] != '\0') {
669                 memset(fname_tmp, ' ', sizeof(fname_tmp));
670                 if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
671                     getFname(rootEntry_ptr->DIR_Name);
672                 else
673                     getRname(rootEntry_ptr->DIR_Name);
674                 print(fname_tmp, strlen(fname_tmp));
675                 if (strlen(fname_tmp) < 13) {
676                     for (int i = 0; i < 13 - strlen(fname_tmp); i++)
677                         printf(" ");
678                 }
679                 cout << setfill(' ') << left << setw(20) << "" << setw(6)
    << rootEntry_ptr->DIR_FileSize << setw(25) << " ";
680                 showDate(rootEntry_ptr->DIR_WrtDate);
681                 showTime(rootEntry_ptr->DIR_WrtTime);
682                 cout << endl;
683             }
684         }
685     }
686     else
687         for (auto ele : ftree) {
688             string x = ele.rname;
689             x += ele.fname;
690             if (x == rname) {
691                 flag = 1;
692                 int startByte = dataBase + (ele.fentry.DIR_FstClus - 2) *
    BytsPerClus + 64;
693                 fseek(fat12, startByte, SEEK_SET);
694                 fread(rootEntry_ptr, 1, 32, fat12);
695                 cout << left << setw(20) << "name" << right << setw(15) <<
    "size" << setw(39) << "date" << endl;
696                 while (rootEntry_ptr->DIR_Name[0] != '\0') {
697                     memset(fname_tmp, ' ', sizeof(fname_tmp));
698                     if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
699                         getFname(rootEntry_ptr->DIR_Name);
700                     else
701                         getRname(rootEntry_ptr->DIR_Name);
702                     print(fname_tmp, strlen(fname_tmp));
703                     if (strlen(fname_tmp) < 13) {
704                         for (int i = 0; i < 13 - strlen(fname_tmp); i++)
705                             printf(" ");
706                     }
707                     cout << setfill(' ') << left << setw(20) << "" <<
    setw(6) << rootEntry_ptr->DIR_FileSize << setw(25) << " ";
708                     showDate(rootEntry_ptr->DIR_WrtDate);
709                     showTime(rootEntry_ptr->DIR_WrtTime);
710                     cout << endl;
711                     startByte += 32;
712                     fseek(fat12, startByte, SEEK_SET);
713                     fread(rootEntry_ptr, 1, 32, fat12);
```

```
                }
            }
        }
    if (flag = 0) {
        cout << "The root rount is not found, please retype it.\n";
    }
}
/*
 * 指令格式：del /文件路径/文件名 （es. /USER/HOULAI.TXT；ps.根目录下为/HOULAI.TXT）
 * 指令功能：删除指定文件
 */
void del(FILE *fat12, string fname) {
    RootEntry newEntry;
    fptr newEptr = &newEntry;
    string sroot, stext;
    newEptr->DIR_Attr = 1;
    int pos = fname.rfind('/');//  /name.txt
    if (pos < 0) {
        printf("The root rount is not found, please retype it.\n");
        return;
    }
    if (pos == 0) {
        sroot = "/";
        stext = fname.substr(pos + 1);
        //stext = sroot + stext;
        int frbase = fileRootBase;
        fptr rootEntry_ptr = &newEntry;
        for (int i = 0; i < RootEntCnt; i++) {
            fseek(fat12, frbase, SEEK_SET);
            fread(rootEntry_ptr, 1, 32, fat12);
            frbase += 32;
            memset(fname_tmp, ' ', sizeof(fname_tmp));
            if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
                getFname(rootEntry_ptr->DIR_Name);
            else
                getRname(rootEntry_ptr->DIR_Name);
            string nn = fname_tmp;
            nn = nn.substr(0, fname.size() - 1);
            if (nn == stext) {
                int clus_num = rootEntry_ptr->DIR_FstClus;
                fseek(fat12, fatBase + clus_num * 3 / 2, SEEK_SET);
                unsigned short w = 0;
                unsigned short* pw = &w;
                fwrite(pw, sizeof(u16), 1, fat12);
                fseek(fat12, fatBase + clus_num * 3 / 2 + FATSz * 512, SEEK_SET);
                fwrite(pw, sizeof(u16), 1, fat12);
                fseek(fat12, frbase - 32, SEEK_SET);
                char n = '\0';
                char *np = &n;
                for (int i = 0; i < 32; i++)
                    fwrite(np, 1, 1, fat12);

            }

        }

```

```
770     }
771     else {
772         sroot = fname.substr(0, pos);
773         stext = fname.substr(pos + 1);
774         for (auto ele : ftree) {
775             string x = ele.rname;
776             x += ele.fname;
777             if (x == sroot) {
778                 fptr rootEntry_ptr = &newEntry;
779                 int startByte = dataBase + (ele.fentry.DIR_FstClus - 2) *
    BytsPerClus;
780                 fseek(fat12, startByte, SEEK_SET);
781                 fread(rootEntry_ptr, 1, 32, fat12);
782                 memset(fname_tmp, ' ', sizeof(fname_tmp));
783                 if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
784                     getFname(rootEntry_ptr->DIR_Name);
785                 else
786                     getRname(rootEntry_ptr->DIR_Name);
787                 string nn = fname_tmp;
788                 nn = nn.substr(0, fname.size());
789                 if (strcmp(nn.c_str(), stext.c_str()) == 0) {
790                     int clus_num = rootEntry_ptr->DIR_FstClus;
791                     fseek(fat12, fatBase + clus_num * 3 / 2, SEEK_SET);
792                     fwrite(0, 1, sizeof(u16), fat12);
793                     fseek(fat12, fatBase + clus_num * 3 / 2 + FATSz * 512,
    SEEK_SET);
794                     fwrite(0, 1, sizeof(u16), fat12);
795                     fseek(fat12, startByte, SEEK_SET);
796                     fwrite(0, 1, 32, SEEK_SET);
797                 }
798             }
799         }
800     }
801     return;
802 }
803 /*
804  * 指令格式：edit /文件路径/文件名 （es. /USER/HOULAI.TXT；ps.根目录下
    为/HOULAI.TXT）
805  * 指令功能：编辑指定文件内容或重命名
806  */
807 void edit(FILE *fat12, string fname) {
808     RootEntry newEntry;
809     fptr newEptr = &newEntry;
810     string sroot, stext;
811     newEptr->DIR_Attr = 1;
812     RootEntry tpEntry;
813     bool flag = 0;
814     string tpLine;
815     vector<string> tpText;
816     int fsize = 0;
817     int pos = fname.rfind('/');//  /name.txt
818     /*if (pos < 0) {
819         printf("%s", warn5);
820         return;
821     }*/
822     if (pos <= 0) {
823         int frbase = fileRootBase;
824         fptr rootEntry_ptr = &newEntry;
```

```
        for (int i = 0; i < RootEntCnt; i++) {
            fseek(fat12, frbase, SEEK_SET);
            fread(rootEntry_ptr, 1, 32, fat12);
            frbase += 32;
            memset(fname_tmp, ' ', sizeof(fname_tmp));
            if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
                getFname(rootEntry_ptr->DIR_Name);
            else
                getRname(rootEntry_ptr->DIR_Name);
            string nn = fname_tmp;
            nn = nn.substr(0, fname.size());
            if (strcmp(nn.c_str(), fname.c_str()) == 0) {
                int clus_num = rootEntry_ptr->DIR_FstClus;
                getchar();
                while (getline(cin, tpLine)) {
                    tpText.push_back(tpLine);
                    fsize += tpLine.length();
                    fsize++; // \n
                }
                int startByte = dataBase + (clus_num - 2) * BytsPerClus;
                fseek(fat12, startByte, SEEK_SET);
                for (auto ele : tpText) {
                    char* tp = (char*)malloc(ele.length() + 1);
                    strcpy(tp, ele.c_str());
                    fwrite(tp, strlen(tp), 1, fat12);
                    free(tp);
                    fwrite("\n", 1, 1, fat12);
                }
                fseek(fat12, frbase - 32, SEEK_SET);
                rootEntry_ptr->DIR_WrtDate = getDate();
                rootEntry_ptr->DIR_WrtTime = getTime();
                printf("Would you rename the file ? (EOF to giv up)\n");
                printf("Enter the new name:\n");
                //getchar();
                cin.clear();
                if (getline(cin, tpLine)) {
                    string stext = tpLine;
                    int pos = stext.rfind('.');//未做鲁棒性处理
                    string prefix, postfix;
                    if (pos >= 8) {
                        prefix = stext.substr(0, 8);
                        postfix = stext.substr(pos + 1).substr(0, 3);
                        stext = prefix + postfix;
                    }
                    else {
                        prefix = stext.substr(0, pos);
                        postfix = stext.substr(pos + 1).substr(0, 3);
                        for (int i = 0; i <= 7 - pos; i++) {
                            prefix += ' ';
                        }
                        stext = prefix + postfix;
                    }
                    strcpy(rootEntry_ptr->DIR_Name, stext.c_str());

                }
                cin.clear();
                rootEntry_ptr->DIR_FileSize = fsize;
                fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
```

```
                return;
            }

        }

    }
    else {
        printf("The file is not found, please retype it.\n");
    }
    return;
}
/*
 * 指令格式：op /文件路径/文件名 （es. /USER/HOULAI.TXT；ps.根目录下为/HOULAI.TXT）
 * 指令功能：打开文本文件
 */
void op(FILE *fat12, string tname)
{
    string str = tname, sroot, stext;
    bool flag = 0;
    int pos = str.rfind('/');
    if (pos < 0)
    {
        stext = str;
        for (auto ele : froot)
        {
            if (strcmp(ele.fname, tname.c_str()) == 0)
            {
                getTextData(fat12, ele.fentry);
                flag = 1;
                break;
            }
        }
    }
    sroot = str.substr(0, pos);
    stext = str.substr(pos + 1);
    for (auto ele : ftree)
    {
        string x = ele.rname;
        x += ele.fname;
        if (x == sroot)
        {
            getTextEntry(fat12, ele.fentry, stext);
            flag = 1;
            break;
        }
    }
    if (flag == 0)
    {
        printf("The file is not found, please retype it.\n");
    }
}

```