

# 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	0x1
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：

32 Bytes	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

- ☒ 模拟读取软盘映像，并展示其中文件内容
- ☒ 打开文本文件
- ☒ 创建新文本文件并编辑内容
- ☒ 编辑文本文件内容，或重命名
- ☒ 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

```
1 struct BPB
2 {
3     u16 BPB_BytsPerSec; //每扇区字节数
4     u8 BPB_SecPerClus; //每簇扇区数
5     u16 BPB_RsvdSecCnt; //Boot记录占用的扇区数
6     u8 BPB_NumFATs; //FAT表个数
7     u16 BPB_RootEntCnt; //根目录最大文件数
8     u16 BPB_TotSec16;
9     u8 BPB_Media;
10    u16 BPB_FATSz16; //FAT扇区数
11    u16 BPB_SecPerTrk;
12    u16 BPB_NumHeads;
```

```

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

```

RootEntry: //根目录项

```

1 struct RootEntry
2 {
3     char DIR_Name[11];
4     u8 DIR_Attr; //文件属性
5     char reserved[10];
6     u16 DIR_WrtTime;
7     u16 DIR_WrtDate;
8     u16 DIR_FstClus; //开始簇号
9     u32 DIR_FileSize;
10 }; // size = 32Bytes

```

fnode: //存储节点

```

1 struct fnode
2 {
3     char rname[40] = { 0 };
4     char fname[12] = { 0 };
5     RootEntry fentry;
6     void append(const char *ch) { strcat(fname, ch); }
7     void rappend(const char *ch) { strcat(rname, ch); }
8 };

```

## 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:

```

1 void getRootFiles(FILE *fat12, fptr rootEntry_ptr) {
2     int frbase = fileRootBase;
3     for (int i = 0; i < RootEntCnt; i++)
4     {
5         fseek(fat12, frbase, SEEK_SET);
6         fread(rootEntry_ptr, 1, 32, fat12);
7         frbase += 32;
8         if ((rootEntry_ptr->DIR_Name[0] == '\0') ||
9             (checkFile(rootEntry_ptr->DIR_Name, 0) == 0)) //过滤非法条目
10             continue;
11         fnode f;
12         if ((rootEntry_ptr->DIR_Attr & 0x10) == 0) //此条目是文件
13         {
14             getFname(rootEntry_ptr->DIR_Name);
15         }
16         else //目录 则放进队列
17         {
18             getRname(rootEntry_ptr->DIR_Name);
19         }
20     }
21 }

```

```

18         f.rappend("/");
19     }
20     f.append(fname_tmp);
21     f.fentry = *rootEntry_ptr;
22     froot.push_back(f);
23 }
24 }

```

DFS:

```

1     getRootFiles(fat12, rootEntry_ptr);
2     for (auto ele : froot) {
3         if ((ele.fentry.DIR_Attr & 0x10) == 0)
4             continue;
5         char tmp[50];
6         memset(tmp, 0, sizeof(tmp));
7         strcat(tmp, ele.rname);
8         strcat(tmp, ele.fname);
9         ftree.push_back(ele);
10        dfs(fat12, ele, tmp);
11    }

```

- 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项结构可以得知，当*i*为偶数时——去掉高四位，*i*为奇数——去掉低四位。

```

1     int getFATValue(FILE *fat12, int num) //3072
2     {
3         u16 bytes;
4         u16 *bytes_ptr = &bytes;
5         fseek(fat12, fatBase + num * 3 / 2, SEEK_SET);
6         fread(bytes_ptr, 1, 2, fat12);
7         return (num & 1) ? (bytes >> 4) : (bytes & ((1 << 12) - 1));
8     }

```

- 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:

```

1     while (curClus < 0xFF8)
2     {
3         if (curClus == 0xFF7)
4         {
5             printf("bad cluster,read failed\n");
6             break;
7         }
8         // operation should continue
9     }

```

- 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:

```

1  u16 DIR_WrtTime;
2  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:

```

1  void showTime(unsigned short time) {
2      u16 tm_min = 0b11111100000;
3      u16 tm_hour = 0b1111100000000000;
4      tm_min = ((tm_min&time) >> 5);
5      tm_hour = ((tm_hour&time) >> 11);
6      cout << setw(2) << tm_hour << ":" << setw(2) << tm_min;
7  }

```

Date:

```

1  void showDate(unsigned short date) {
2      u16 year = 0b1111111000000000;
3      u16 month = 0b111100000;
4      u16 day = 0b11111;
5      year = ((year&date) >> 9) + 1980;
6      month = ((month&date) >> 5);
7      day = (day & date);
8      cout << year << "/" << right << setw(2) << setfill('0') << month << "/"
9      << setw(2) << setfill('0') << day << "    ";
10 }

```

- 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。

```

1  RootEntry WirteIntext(FILE *fat12); //读取输入内容，并返回由此内容创建的目录项
2  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：

```
E:\DATA\OS\LAB_FAT12\FAT12\Debug\FAT12.exe
-ls 展开文件树
-dir 展开文件详情
-creat 创建文本文件
-edit 编辑文本文件
-del 删除文件
-op 打开文件
-q 退出
>
```

ls:

```
E:\DATA\OS\LAB_FAT12\FAT12\Debug\FAT12.exe
-ls 展开文件树
-dir 展开文件详情
-creat 创建文本文件
-edit 编辑文本文件
-del 删除文件
-op 打开文件
-q 退出
>ls
/;IO.SYS MSDOS.SYS COMMAND.COM DRVSPAC.BIN USER
/USER/:
. . . JOIN JACK LUCY
/USER/JOIN/:
. . . MUSIC PICTURE BOOK
/USER/JOIN/MUSIC/:
. . . HOULAI.TXT
/USER/JOIN/PICTURE/:
. . .
/USER/JOIN/BOOK/:
. . .
/USER/JACK/:
. . .
/USER/LUCY/:
. . .
>
_
```

dir:

```
>dir /
name                                size                                date                                12:52
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                                12:52
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/qg\\_39654127/article/details/88429461#main-toc](https://blog.csdn.net/qg_39654127/article/details/88429461#main-toc)

## 附录:

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <string>
4  #include <stdlib.h>
5  #include <vector>
6  #include <string.h>
7  #include <ctime>
8  #include <cstdio>
9  #include <iomanip>
10 using namespace std;
11 typedef unsigned char u8;    //1字节
12 typedef unsigned short u16; //2字节
13 typedef unsigned int u32;    //4字节
14 int BytsPerSec;              //每扇区字节数
15 int SecPerClus;              //每簇扇区数
16 int RsvdSecCnt;              //Boot记录占用的扇区数
17 int NumFATS;                 //FAT表个数
18 int RootEntCnt;              //根目录最大文件数
19 int FATSz;                   //FAT扇区数
20 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
21 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
22 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
23 int BytsPerClus;
24 #pragma pack(1) /*指定按1字节对齐*/
25 struct BPB
26 {
27     u16 BPB_BytsPerSec; //每扇区字节数
28     u8 BPB_SecPerClus;  //每簇扇区数
29     u16 BPB_RsvdSecCnt; //Boot记录占用的扇区数
30     u8 BPB_NumFATS;     //FAT表个数
31     u16 BPB_RootEntCnt; //根目录最大文件数
32     u16 BPB_TotSec16;
33     u8 BPB_Media;
34     u16 BPB_FATSz16;    //FAT扇区数
35     u16 BPB_SecPerTrk;
36     u16 BPB_NumHeads;
37     u32 BPB_HiddSec;
```

```

38     u32 BPB_TotSec32; //如果BPB_FATSize16为0, 该值为FAT扇区数
39 }; // size = 25Bytes
40 //根目录条目
41 struct RootEntry
42 {
43     char DIR_Name[11];
44     u8 DIR_Attr; //文件属性
45     char reserved[10];
46     u16 DIR_WrtTime;
47     u16 DIR_WrtDate;
48     u16 DIR_FstClus; //开始簇号
49     u32 DIR_FileSize;
50 }; // size = 32Bytes
51 #pragma pack() /*取消指定对齐, 恢复缺省对齐*/
52 typedef struct RootEntry *fptr;
53 struct fnode
54 {
55     char rname[40] = { 0 };
56     char fname[12] = { 0 };
57     RootEntry fentry;
58     void append(const char *ch) { strcat(fname, ch); }
59     void rappend(const char *ch) { strcat(rname, ch); }
60 };
61 int checkFile(char *, int);
62 void getFName(const char *);
63 void getRName(const char *);
64 unsigned short getTime();
65 unsigned short getDate();
66 void showTime(unsigned short);
67 void showDate(unsigned short);
68 // 写入生成文本文件目录项内容
69 RootEntry writeIntext(FILE *fat12);
70 // 写入生成文本文件的目录项并返回此目录项地址
71 void getWriteEntry(FILE *fat12, string tname);
72 // 初始化根目录项
73 void getRootFiles(FILE *fat12, struct RootEntry *rootEntry_ptr);
74 int getFATValue(FILE *fat12, int num);
75 void dfs(FILE *fat12, fnode cur, char *rname);
76 void op(FILE *fat12, string tname);
77 void edit(FILE *fat12, string fname);
78 void del(FILE *fat12, string fname);
79 void dir(FILE *fat12, string rname);
80 void print(char *, int);
81 void ls(FILE *fat12);
82 char transletter(char x);
83 // 从BPB表初始化FAT信息
84 void ini(FILE* fat12);
85 char dot1[10] = ". . . ";
86 char dot2[10] = ".\n..\n";
87 char colon[2] = ":";
88 char tdata[100000];
89 char fname_tmp[13];
90 // 根目录文件队列
91 vector<fnode> froot;
92 // 文件树
93 vector<fnode> ftree;
94 void print(char *x, int a) {
95     x[a] = '\0';

```

```

96     printf("%s", x);
97 }
98 unsigned short getTime() {
99     time_t nowtime;
100     struct tm* p;;
101     time(&nowtime);
102     p = localtime(&nowtime);
103     unsigned short ans = (p->tm_hour << 11) + (p->tm_min << 5);
104     return ans;
105 }
106 void showTime(unsigned short time) {
107     u16 tm_min = 0b11111100000;
108     u16 tm_hour = 0b1111100000000000;
109     tm_min = ((tm_min&time) >> 5);
110     tm_hour = ((tm_hour&time) >> 11);
111     cout << setw(2) << tm_hour << ":" << setw(2) << tm_min;
112 }
113 unsigned short getDate() {
114     time_t nowtime;
115     struct tm* p;;
116     time(&nowtime);
117     p = localtime(&nowtime);
118     unsigned short year = p->tm_year - 80;
119     unsigned short mon = p->tm_mon + 1;
120     unsigned short ans = (year << 9) + (mon << 5) + p->tm_mday;
121     return ans;
122 }
123 void showDate(unsigned short date) {
124     u16 year = 0b1111111000000000;
125     u16 month = 0b111100000;
126     u16 day = 0b11111;
127     year = ((year&date) >> 9) + 1980;
128     month = ((month&date) >> 5);
129     day = (day & date);
130     cout << year << "/" << right << setw(2) << setfill('0') << month <<
131     "/" << setw(2) << setfill('0') << day << "    ";
132 }
133 void getTextData(FILE *fat12, RootEntry re)
134 {
135     int curClus = re.DIR_FstClus;
136     int startByte;
137     int fsize = re.DIR_FileSize;
138     while (curClus < 0xFF8)
139     {
140         if (curClus == 0xFF7)
141         {
142             printf("bad cluster,read failed\n");
143             break;
144         }
145         startByte = dataBase + (curClus - 2) * BytsPerClus;
146         fseek(fat12, startByte, SEEK_SET);
147         if (fsize >= BytsPerClus) {
148             fread(tdata, 1, BytsPerClus, fat12);
149             fsize -= BytsPerClus;
150         }
151         else
152             fread(tdata, 1, fsize, fat12);

```

```

153     tdata[fsize - 1] = '\0';
154     cout << tdata;
155     curClus = getFATValue(fat12, curClus); //获取fat项的内容
156 }
157 }
158 void getTextEntry(FILE *fat12, RootEntry re, string tname)
159 {
160     int curClus = re.DIR_FstClus;
161     int startByte;
162     while (curClus < 0xFF8)
163     {
164         if (curClus == 0xFF7)
165         {
166
167             printf("bad cluster,read failed\n");
168             break;
169         }
170         startByte = dataBase + (curClus - 2) * BytsPerClus;
171         for (int loop = 0; loop < BytsPerClus; loop += 32)
172         {
173             RootEntry roottmp;
174             fptr rootptr = &roottmp;
175             fseek(fat12, startByte + loop, SEEK_SET);
176             fread(rootptr, 1, 32, fat12);
177             if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr-
>DIR_Name, 0) == 0) || (rootptr->DIR_Attr & 0x10) != 0)
178                 continue;
179             getFname(rootptr->DIR_Name);
180             string tmp = fname_tmp;
181             if (tmp == tname)
182             {
183                 getTextData(fat12, *rootptr);
184                 break;
185             }
186         }
187         curClus = getFATValue(fat12, curClus); //获取fat项的内容
188     }
189 }
190 int getFATValue(FILE *fat12, int num) //3072
191 {
192     u16 bytes;
193     u16 *bytes_ptr = &bytes;
194     fseek(fat12, fatBase + num * 3 / 2, SEEK_SET);
195     fread(bytes_ptr, 1, 2, fat12);
196     return (num & 1) ? (bytes >> 4) : (bytes & ((1 << 12) - 1));
197 }
198 int checkFile(char *fname_tmp, int pos)
199 {
200     for (int j = pos; j < pos + 11; j++)
201     {
202         if (!(((fname_tmp[j] >= 48) && (fname_tmp[j] <= 57)) ||
203             ((fname_tmp[j] >= 65) && (fname_tmp[j] <= 90)) ||
204             ((fname_tmp[j] >= 97) && (fname_tmp[j] <= 122)) ||
205             (fname_tmp[j] == ' ')))
206             return 0;
207     }
208     return 1;
209 }

```

```

210 void getRname(const char *dirname)
211 {
212     int tmpLen = 0;
213     for (int k = 0; k < 12 && dirname[k] != ' '; k++)
214         fname_tmp[tmpLen++] = transletter(dirname[k]);
215     fname_tmp[tmpLen] = '\0';
216 }
217 void getFname(const char *dirname)
218 {
219     int tmpLen = 0;
220     for (int k = 0; k < 11; k++)
221     {
222         if (dirname[k] != ' ')
223             fname_tmp[tmpLen++] = transletter(dirname[k]);
224         else
225         {
226             fname_tmp[tmpLen++] = '.';
227             while ((dirname[k] == ' ') && k < 11) // 过滤空格
228                 k++;
229             k--;
230         }
231     }
232     fname_tmp[tmpLen] = '\0';
233 }
234 char transletter(char x)
235 {
236     if ((x <= 'z') && (x >= 'a'))
237     {
238         x -= 32;
239     }
240     return x;
241 }
242 void help() {
243     cout << "                                -ls 展开文件树" <<
endl;
244     cout << "                                -dir 展开文件详情" <<
endl;
245     cout << "                                -creat 创建文本文件" <<
endl;
246     cout << "                                -edit 编辑文本文件" <<
endl;
247     cout << "                                -del 删除文件" <<
endl;
248     cout << "                                -op 打开文件" << endl;
249     cout << "                                -q 退出" << endl;
250 }
251 int main() {
252     FILE *fat12;
253     fat12 = fopen("a.img", "r+");
254     ini(fat12);
255     help();
256     printf(">");
257     string cmd, tname;
258     cin >> cmd ;
259     while (1) {
260         if (cmd == "q" || cmd == "Q") {
261             cout << "Quit." << endl;
262             return 0;

```

```
263     }
264     else if (cmd == "creat") {
265         cin >> tname;
266         froot.clear();
267         ftree.clear();
268         ini(fat12);
269         getWritrEntry(fat12, tname);
270         cin.clear();
271         printf("\n>");
272         cin >> cmd;
273     }
274     else if (cmd == "ls") {
275         froot.clear();
276         ftree.clear();
277         ini(fat12);
278         ls(fat12);
279         printf("\n>");
280         cin >> cmd;
281     }
282     else if (cmd == "dir") {
283         cin >> tname;
284         froot.clear();
285         ftree.clear();
286         ini(fat12);
287         dir(fat12, tname);
288         printf("\n>");
289         cin >> cmd;
290     }
291     else if (cmd == "del") {
292         cin >> tname;
293         froot.clear();
294         ftree.clear();
295         ini(fat12);
296         del(fat12, tname);
297         printf("\n>");
298         cin >> cmd;
299     }
300     else if (cmd == "edit") {
301         cin >> tname;
302         froot.clear();
303         ftree.clear();
304         ini(fat12);
305         edit(fat12, tname);
306         printf("\n>");
307         cin >> cmd;
308     }
309     else if (cmd == "op") {
310         cin >> tname;
311         froot.clear();
312         ftree.clear();
313         ini(fat12);
314         op(fat12, tname);
315         printf("\n>");
316         cin >> cmd;
317     }
318     else {
319         cin.clear();
320         printf("\n>");
```

```

321         cin >> cmd;
322     }
323 }
324 return 0;
325 }
326
327 void ini(FILE *fat12)
328 {
329     struct BPB bpb;
330     struct BPB *bpb_ptr = &bpb;    //载入BPB
331     fseek(fat12, 11, SEEK_SET);    //BPB从偏移11个字节处开始
332     fread(bpb_ptr, 1, 25, fat12); //BPB长度为25字节
333
334     BytsPerSec = bpb_ptr->BPB_BytsPerSec; //初始化各个全局变量
335     SecPerClus = bpb_ptr->BPB_SecPerClus;
336     RsvdSecCnt = bpb_ptr->BPB_RsvdSecCnt;
337     NumFATS = bpb_ptr->BPB_NumFATS;
338     RootEntCnt = bpb_ptr->BPB_RootEntCnt;
339     if (bpb_ptr->BPB_FATSz16 != 0)
340         FATSz = bpb_ptr->BPB_FATSz16;
341     else
342         FATSz = bpb_ptr->BPB_TotSec32;
343     fatBase = RsvdSecCnt * BytsPerSec;
344     fileRootBase = (RsvdSecCnt + NumFATS * FATSz) * BytsPerSec; //根目录首字
节的偏移数=boot+fat1&2的总字节数
345     dataBase = BytsPerSec * (RsvdSecCnt + FATSz * NumFATS + (RootEntCnt *
32 + BytsPerSec - 1) / BytsPerSec);
346     BytsPerClus = SecPerClus * BytsPerSec; //每簇的字节数
347     struct RootEntry rootEntry;
348     fptr rootEntry_ptr = &rootEntry;
349     getRootFiles(fat12, rootEntry_ptr);
350     for (auto ele : froot) {
351         if ((ele.fentry.DIR_Attr & 0x10) == 0)
352             continue;
353         char tmp[50];
354         memset(tmp, 0, sizeof(tmp));
355         strcat(tmp, ele.rname);
356         strcat(tmp, ele.fname);
357         ftree.push_back(ele);
358         dfs(fat12, ele, tmp);
359     }
360 }
361 void getRootFiles(FILE *fat12, fptr rootEntry_ptr) {
362     int frbase = fileRootBase;
363     for (int i = 0; i < RootEntCnt; i++)
364     {
365         fseek(fat12, frbase, SEEK_SET);
366         fread(rootEntry_ptr, 1, 32, fat12);
367         frbase += 32;
368         if ((rootEntry_ptr->DIR_Name[0] == '\0') ||
369             (checkFile(rootEntry_ptr->DIR_Name, 0) == 0)) //过滤非法条目
370             continue;
371         struct fnode f;
372         if ((rootEntry_ptr->DIR_Attr & 0x10) == 0) //此条目是文件
373         {
374             getFname(rootEntry_ptr->DIR_Name);
375         }
376         else //目录 则放进队列

```

```

376     {
377         getName(rootEntry_ptr->DIR_Name);
378         f.rappend("/");
379     }
380     f.append(fname_tmp);
381     f.fentry = *rootEntry_ptr;
382     froot.push_back(f);
383 }
384 }
385 void dfs(FILE *fat12, fnode cur, char *rname)
386 {
387     int curClus = cur.fentry.DIR_FstClus;
388     int startByte;
389     while (curClus < 0xFF8)
390     {
391         if (curClus == 0xFF7)
392         {
393             printf("Bad cluster\n");
394             break;
395         }
396         startByte = dataBase + (curClus - 2) * BytsPerClus;
397         for (int loop = 0; loop < BytsPerClus; loop += 32)
398         {
399             RootEntry roottmp;
400             fptr rootptr = &roottmp;
401             fseek(fat12, startByte + loop, SEEK_SET);
402             fread(rootptr, 1, 32, fat12);
403             if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr->
404 >DIR_Name, 0) == 0) || (rootptr->DIR_Attr & 0x10) == 0)
405                 continue;
406             getName(rootptr->DIR_Name);
407             fnode f;
408             f.append(fname_tmp);
409             f.rappend(rname);
410             f.rappend("/");
411             f.fentry = *rootptr;
412             char tmp[50];
413             memset(tmp, 0, sizeof(tmp));
414             strcat(tmp, f.rname);
415             strcat(tmp, f.fname);
416             ftree.push_back(f);
417             dfs(fat12, f, tmp);
418         }
419         curClus = getFATValue(fat12, curClus); //获取fat项的内容
420     }
421 }
422 /*
423  * 指令格式: creat /文件路径/文件名 (es. /USER/HOULAI.TXT; ps.根目录下
424 为/HOULAI.TXT)
425  * 指令功能: 生成文本文件, 并可编辑内容
426  */
427 RootEntry WirteIntext(FILE *fat12) {
428     int curClus;
429     int startByte;
430     RootEntry tpEntry;
431     //fptr tpEntryptr = (fptr)malloc(sizeof(RootEntry));
432     string tpLine;
433     int fsize = 0;

```



```

432     vector<string> tpText;
433     getchar();
434     while (getline(cin, tpLine)) {
435         tpText.push_back(tpLine);
436         fsize += tpLine.length();
437         fsize++; // \n
438     }
439     tpEntry.DIR_FileSize = fsize;
440     for (int i = 0; i < 3072; i++) {
441         if (getFATValue(fat12, i) == 0) {
442             int startByte = dataBase + (i - 2) * BytsPerClus;
443             fseek(fat12, startByte, SEEK_SET);
444             for (auto ele : tpText) {
445                 char* tp = (char*)malloc(ele.length() + 1);
446                 strcpy(tp, ele.c_str());
447                 fwrite(tp, strlen(tp), 1, fat12);
448                 free(tp);
449                 fwrite("\n", 1, 1, fat12);
450             }
451
452             tpEntry.DIR_FstClus = (short)i;
453             tpEntry.DIR_WrtDate = (unsigned short)getDate();
454             tpEntry.DIR_WrtTime = (unsigned short)getTime();
455             tpEntry.DIR_Attr = (char)0;
456
457             u16 bytes;
458             u16 bytes2 = 0xff8;
459             u16 *bytes_ptr = &bytes;
460             fseek(fat12, fatBase + i * 3 / 2, SEEK_SET);
461             fread(bytes_ptr, 1, 2, fat12);
462             if (i & 1) {
463                 bytes2 = (bytes2 << 4) + (bytes & 15);
464             }
465             else {
466                 bytes2 = bytes2 + (bytes & (61440));
467             }
468             fseek(fat12, fatBase + i * 3 / 2, SEEK_SET);
469             fwrite(&bytes2, sizeof(u16), 1, fat12);
470             fseek(fat12, fatBase + i * 3 / 2 + FATSz * 512, SEEK_SET);
471             fwrite(&bytes2, sizeof(u16), 1, fat12);
472             //free(tpEntryPtr);
473             return tpEntry;
474         }
475     }
476 }
477 void getWritrEntry(FILE *fat12, string tname) {
478     RootEntry newEntry;
479     RootEntry tpEntry = WirteIntext(fat12);
480     fptr newEptr = &newEntry;
481     string sroot, stext;
482     newEptr->DIR_Attr = 1;
483     int pos = tname.rfind('/'); // /name.txt
484     if (pos < 0) {
485         printf("Root is not find, please retype it\n");
486         return;
487     }
488     if (pos == 0) {
489         sroot = "/";

```

```

490     stext = tname.substr(pos + 1);
491     int frbase = fileRootBase;
492     fptr rootEntry_ptr = &newEntry;
493     for (int i = 0; i < RootEntCnt; i++) {
494         fseek(fat12, frbase, SEEK_SET);
495         fread(rootEntry_ptr, 1, 32, fat12);
496         frbase += 32;
497         if ((rootEntry_ptr->DIR_Name[0] != '\0')) //过滤非法条目
498             continue;
499         pos = stext.rfind('.');
500         string prefix, postfix;
501         if (pos >= 8) {
502             prefix = stext.substr(0, 8);
503             postfix = stext.substr(pos + 1).substr(0, 3);
504             stext = prefix + postfix;
505         }
506         else {
507             prefix = stext.substr(0, pos);
508             postfix = stext.substr(pos + 1).substr(0, 3);
509             for (int i = 0; i <= 7 - pos; i++) {
510                 prefix += ' ';
511             }
512             stext = prefix + postfix;
513         }
514         strcpy(rootEntry_ptr->DIR_Name, stext.c_str());
515         rootEntry_ptr->DIR_Attr = 0x20;
516         rootEntry_ptr->DIR_FileSize = tpEntry.DIR_FileSize;
517         rootEntry_ptr->DIR_FstClus = tpEntry.DIR_FstClus;
518         rootEntry_ptr->DIR_WrtDate = tpEntry.DIR_WrtDate;
519         rootEntry_ptr->DIR_WrtTime = tpEntry.DIR_WrtTime;
520         fseek(fat12, -32, SEEK_CUR);
521         fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
522         return;
523     }
524
525 }
526 else {
527     sroot = tname.substr(0, pos);
528     stext = tname.substr(pos + 1);
529     for (auto ele : ftree) {
530         string x = ele.rname;
531         x += ele.fname;
532         if (x == sroot) {
533             fptr rootEntry_ptr = &newEntry;
534             int startByte = dataBase + (ele.fentry.DIR_FstClus - 2) *
BytsPerClus;
535             fseek(fat12, startByte, SEEK_SET);
536             fread(rootEntry_ptr, 1, 32, fat12);
537             while (rootEntry_ptr->DIR_Name[0] != '\0') {
538                 startByte += 32;
539                 fseek(fat12, startByte, SEEK_SET);
540                 fread(rootEntry_ptr, 1, 32, fat12);
541             }
542             pos = stext.rfind('.');
543             string prefix, postfix;
544             if (pos >= 8) {
545                 prefix = stext.substr(0, 8);
546                 postfix = stext.substr(pos + 1).substr(0, 3);

```

```

547         stext = prefix + postfix;
548     }
549     else {
550         prefix = stext.substr(0, pos);
551         postfix = stext.substr(pos + 1).substr(0, 3);
552         for (int i = 0; i <= 7 - pos; i++) {
553             prefix += ' ';
554         }
555         stext = prefix + postfix;
556     }
557     strcpy(rootEntry_ptr->DIR_Name, stext.c_str());
558     rootEntry_ptr->DIR_Attr = 0x01;
559     rootEntry_ptr->DIR_FileSize = tpEntry.DIR_FileSize;
560     rootEntry_ptr->DIR_FstClus = tpEntry.DIR_FstClus;
561     rootEntry_ptr->DIR_WrtDate = tpEntry.DIR_WrtDate;
562     rootEntry_ptr->DIR_WrtTime = tpEntry.DIR_WrtTime;
563     fseek(fat12, startByte, SEEK_SET);
564     fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
565 }
566     }
567 }
568 return;
569 }
570 /*
571  * 指令格式: ls
572  * 指令功能: 展开文件树
573  */
574 struct lsfile
575 {
576     bool flag;
577     char fname[12];
578     void append(char *ch) { strcat(fname, ch); }
579     void ini()
580     {
581         memset(fname, 0, sizeof(fname));
582     }
583     int fsize;
584 };
585 vector<lsfile> vcfile;
586 void Traverse(fnode cur, FILE *fat12)
587 {
588     vcfile.clear();
589     int curClus = cur.fentry.DIR_FstClus, startByte;
590     while (curClus < 0xFF8)
591     {
592         if (curClus == 0xFF7)
593         {
594             printf("bad cluster, read failed\n");
595             break;
596         }
597         startByte = dataBase + (curClus - 2) * BytsPerClus;
598         for (int loop = 0; loop < BytsPerClus; loop += 32)
599         {
600             RootEntry roottmp;
601             fptr rootptr = &roottmp;
602             fseek(fat12, startByte + loop, SEEK_SET);
603             fread(rootptr, 1, 32, fat12);

```

```

604         if ((rootptr->DIR_Name[0] == '\0') || (checkFile(rootptr->
        >DIR_Name, 0) == 0))
605             continue;
606         if ((rootptr->DIR_Attr & 0x10) == 0)
607         {
608             lsfile tpfile;
609             tpfile.ini();
610             tpfile.flag = 0;
611             getFname(rootptr->DIR_Name);
612             tpfile.append(fname_tmp);
613             tpfile.fsize = rootptr->DIR_FileSize;
614             vcfile.push_back(tpfile);
615         }
616         else
617         {
618             lsfile tpfile;
619             tpfile.ini();
620             tpfile.flag = 1;
621             getRname(rootptr->DIR_Name);
622             tpfile.append(fname_tmp);
623             vcfile.push_back(tpfile);
624         }
625     }
626     curclus = getFATValue(fat12, curclus);
627 }
628 return;
629 }
630 void ls(FILE *fat12) {
631     int mode;
632     printf("/:");
633     for (auto ele : froot)
634     {
635         print(ele.fname, strlen(ele.fname));
636         printf(" ");
637     }
638     printf("\n");
639
640     for (auto ele : ftree)
641     {
642         Traverse(ele, fat12);
643         print(ele.rname, strlen(ele.rname));
644         print(ele.fname, strlen(ele.fname));
645         printf("/:");
646         printf("\n");
647         printf("%s", dot1);
648         for (auto obj : vcfile)
649         {
650             print(obj.fname, strlen(obj.fname));
651             printf(" ");
652         }
653         printf("\n");
654     }
655 }
656 /*
657  * 指令格式: dir /文件路径 (es. /USER/; ps.根目录下为/)
658  * 指令功能: 展示指定路径下文件项
659  */
660 void dir(FILE *fat12, string rname) {

```



```

714     }
715 }
716 }
717 if (flag == 0) {
718     cout << "The root rount is not found, please retype it.\n";
719 }
720 }
721 /*
722  * 指令格式: del /文件路径/文件名 (es. /USER/HOULAI.TXT; ps.根目录下
    为/HOULAI.TXT)
723  * 指令功能: 删除指定文件
724  */
725 void del(FILE *fat12, string fname) {
726     RootEntry newEntry;
727     fptr newEpPtr = &newEntry;
728     string sroot, stext;
729     newEpPtr->DIR_Attr = 1;
730     int pos = fname.rfind('/'); // /name.txt
731     if (pos < 0) {
732         printf("The root rount is not found, please retype it.\n");
733         return;
734     }
735     if (pos == 0) {
736         sroot = "/";
737         stext = fname.substr(pos + 1);
738         //stext = sroot + stext;
739         int frbase = fileRootBase;
740         fptr rootEntry_ptr = &newEntry;
741         for (int i = 0; i < RootEntCnt; i++) {
742             fseek(fat12, frbase, SEEK_SET);
743             fread(rootEntry_ptr, 1, 32, fat12);
744             frbase += 32;
745             memset(fname_tmp, ' ', sizeof(fname_tmp));
746             if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
747                 getFName(rootEntry_ptr->DIR_Name);
748             else
749                 getRName(rootEntry_ptr->DIR_Name);
750             string nn = fname_tmp;
751             nn = nn.substr(0, fname.size() - 1);
752             if (nn == stext) {
753                 int clus_num = rootEntry_ptr->DIR_FstClus;
754                 fseek(fat12, fatBase + clus_num * 3 / 2, SEEK_SET);
755                 unsigned short w = 0;
756                 unsigned short* pw = &w;
757                 fwrite(pw, sizeof(u16), 1, fat12);
758                 fseek(fat12, fatBase + clus_num * 3 / 2 + FATSz * 512,
    SEEK_SET);
759                 fwrite(pw, sizeof(u16), 1, fat12);
760                 fseek(fat12, frbase - 32, SEEK_SET);
761                 char n = '\0';
762                 char *np = &n;
763                 for (int i = 0; i < 32; i++)
764                     fwrite(np, 1, 1, fat12);
765             }
766         }
767     }
768 }
769

```

```

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;

```

```

825     for (int i = 0; i < RootEntCnt; i++) {
826         fseek(fat12, frbase, SEEK_SET);
827         fread(rootEntry_ptr, 1, 32, fat12);
828         frbase += 32;
829         memset(fname_tmp, ' ', sizeof(fname_tmp));
830         if ((rootEntry_ptr->DIR_Attr & 0x10) == 0)
831             getFName(rootEntry_ptr->DIR_Name);
832         else
833             getRName(rootEntry_ptr->DIR_Name);
834         string nn = fname_tmp;
835         nn = nn.substr(0, fname.size());
836         if (strcmp(nn.c_str(), fname.c_str()) == 0) {
837             int clus_num = rootEntry_ptr->DIR_FstClus;
838             getchar();
839             while (getline(cin, tpLine)) {
840                 tpText.push_back(tpLine);
841                 fsize += tpLine.length();
842                 fsize++; // \n
843             }
844             int startByte = dataBase + (clus_num - 2) * BytsPerClus;
845             fseek(fat12, startByte, SEEK_SET);
846             for (auto ele : tpText) {
847                 char* tp = (char*)malloc(ele.length() + 1);
848                 strcpy(tp, ele.c_str());
849                 fwrite(tp, strlen(tp), 1, fat12);
850                 free(tp);
851                 fwrite("\n", 1, 1, fat12);
852             }
853             fseek(fat12, frbase - 32, SEEK_SET);
854             rootEntry_ptr->DIR_WrtDate = getDate();
855             rootEntry_ptr->DIR_WrtTime = getTime();
856             printf("Would you rename the file ? (EOF to giv up)\n");
857             printf("Enter the new name:\n");
858             //getchar();
859             cin.clear();
860             if (getline(cin, tpLine)) {
861                 string stext = tpLine;
862                 int pos = stext.rfind('.'); //未做鲁棒性处理
863                 string prefix, postfix;
864                 if (pos >= 8) {
865                     prefix = stext.substr(0, 8);
866                     postfix = stext.substr(pos + 1).substr(0, 3);
867                     stext = prefix + postfix;
868                 }
869                 else {
870                     prefix = stext.substr(0, pos);
871                     postfix = stext.substr(pos + 1).substr(0, 3);
872                     for (int i = 0; i <= 7 - pos; i++) {
873                         prefix += ' ';
874                     }
875                     stext = prefix + postfix;
876                 }
877                 strcpy(rootEntry_ptr->DIR_Name, stext.c_str());
878             }
879             cin.clear();
880             rootEntry_ptr->DIR_FileSize = fsize;
881             fwrite(rootEntry_ptr, sizeof(RootEntry), 1, fat12);
882

```



```

883         return;
884     }
885
886     }
887
888     }
889     else {
890         printf("The file is not found, please retype it.\n");
891     }
892     return;
893 }
894 /*
895  * 指令格式: op /文件路径/文件名 (es. /USER/HOULAI.TXT; ps.根目录下
    为/HOULAI.TXT)
896  * 指令功能: 打开文本文件
897  */
898 void op(FILE *fat12, string tname)
899 {
900     string str = tname, sroot, stext;
901     bool flag = 0;
902     int pos = str.rfind('/');
903     if (pos < 0)
904     {
905         stext = str;
906         for (auto ele : froot)
907         {
908             if (strcmp(ele.fname, tname.c_str()) == 0)
909             {
910                 getTextData(fat12, ele.fentry);
911                 flag = 1;
912                 break;
913             }
914         }
915     }
916     sroot = str.substr(0, pos);
917     stext = str.substr(pos + 1);
918     for (auto ele : ftree)
919     {
920         string x = ele.rname;
921         x += ele.fname;
922         if (x == sroot)
923         {
924             getTextEntry(fat12, ele.fentry, stext);
925             flag = 1;
926             break;
927         }
928     }
929     if (flag == 0)
930     {
931         printf("The file is not found, please retype it.\n");
932     }
933 }
934

```