FAT12 文件提取

顾郑林 201180055

一、定义

1、单位字节

```
typedef unsigned char u8;
typedef unsigned short u16;
typedef unsigned int u32;
```

分别定义 8 位（1 字节），16 位（2 字节），32 位（4 字节）。

定义为无符号数，采用逻辑移位，防止右移时补 1。

2、全局变量（引导扇区）

```
int BytsPerSec;     //每扇区字节数
int SecPerClus;     //每簇扇区数
int RsvdSecCnt;     //Boot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
```

常用信息写入全局变量，减少函数传参。

3、结构体 BPB

```
struct BPB
{
    u16 BPB_BytsPerSec;     //每扇区字节数
    u8  BPB_SecPerClus;     //每簇扇区数
    u16 BPB_RsvdSecCnt;     //Boot扇区数
    u8  BPB_numFATs;        //FAT表个数
    u16 BPB_RootEntCnt;     //根目录最大文件数
    u16 BPB_TotSec16;       //逻辑扇区总数
    u8  BPB_Media;          //介质描述符
    u16 BPB_FATSz16;        //FAT扇区数
    u16 BPB_SecPerTrk;      //每磁道扇区数
    u16 BPB_NumHeads;       //磁头数
    u32 BPB_HiddSec;        //隐藏扇区数
    u32 BPB_TotSec32;       //逻辑扇区总数
};
```

启动区有用信息，偏移量 11，长度 25。

4、根目录

```
truct RootEntry
{
```

```
    char DIR_Name[11];      //文件名+拓展名
    u8   DIR_Attr;          //文件属性
    char DIR_NTRes[10];     //保留
    u16  DIR_WrtTime;   //最后写入时间
    u16  DIR_WrtDate;   //最后写入日期
    u16  DIR_FstClus;   //起始簇号
    u32  DIR_FileSize;      //文件大小
};
```

单个文件的条目信息。

二、函数

1、取 BPB 信息

```
void getBPBvalue(FILE* fat12,struct BPB* bpb_ptr)
{
    int flag;
    //指向BPB第11字节
    flag = fseek(fat12, 11, SEEK_SET);
    if(flag==-1)        //错误处理
    {
        cout << "fseek error!" << endl;
        exit(0);
    }
    //读取25字节的数据
    flag = fread(bpb_ptr, 1, 25, fat12);
    if(flag==-1)        //错误处理
    {
        cout << "fread error ! " << endl;
        exit(0);
    }
}
```

将映像中的数据读到结构体 BPB 中。

2、从根目录读取文件名和位置

```
void getFilename(FILE* fat12,struct RootEntry* rootEntry_ptr,int* filelocation)
{
    int offset = (RsvdSecCnt + NumFATs * FATSz) * BytsPerSec;//根目录偏移量
    int flag;
    int index = 0;
    for(int i=0;i<RootEntCnt;i++)
    {
        flag = fseek(fat12, offset+i*32, SEEK_SET);     //读取第i个条目
        if(flag==-1)                //错误处理
        {
```

```
            cout << "fseek error! line:84" << endl;
            exit(0);
        }
        flag = fread(rootEntry_ptr, 1, 32, fat12);
        if(flag!=32)
        {
            cout << "fread error! line:90" << endl;
            exit(0);
        }
        if(rootEntry_ptr->DIR_Name[0]=='\0')          //跳过无效条目
            continue;
        int sign = 0;
        for(int j=0;j<11;j++)                         //判断是否为文件
        {
            if (!(((rootEntry_ptr->DIR_Name[j] >= '0') && (rootEntry_ptr->DIR_Name[j]
<= '9')) ||((rootEntry_ptr->DIR_Name[j] >= 'A') && (rootEntry_ptr->DIR_Name[j] <= 'Z'))
||((rootEntry_ptr->DIR_Name[j] >= 'a') && (rootEntry_ptr->DIR_Name[j] <= 'z'))
||rootEntry_ptr->DIR_Name[j] == ' '))
            {
                sign = 1;
                break;
            }
        }
        if(sign==1)
            continue;
        filelocation[index++] = i;
    }
}
```

遍根目录，找出所有可用文件，将文件在根目录中的位置写入索引。

3、从 FAT 表的内容读出当前簇的下一簇号

```
u16 getValue(FILE* fat12,int num)             //从FAT表读取下一簇号
{
    int fat1_ptr = RsvdSecCnt * BytsPerSec;
    int fatpos = fat1_ptr + num * 3 / 2;
    int type = -1;
    if (num % 2 == 0)
        type = 0;
    else
        type = 1;

    u16 bytes;
    u16* bytes_ptr = &bytes;
    int flag;
```

```
    flag = fseek(fat12, fatpos, SEEK_SET);
    if(flag==-1)
    {
        cout << "fseek error! line:129" << endl;
        exit(0);
    }
    flag = fread(bytes_ptr, 1, 2, fat12);
    if(flag!=2)
    {
        cout << "fread error! ;line:135" << endl;
        exit(0);
    }
    u16 byte1 = bytes << 4;
    byte1 = byte1 >> 4;
    if (type == 0)
        return byte1;       //返回byte2的低4位和byte1
    else
        return (bytes >> 4)&0x0FFF;         //返回byte2和byte1的高4位
}
```

簇号的读取：

例如 0001 0011 0111 1111

若簇号为偶数，需要低 12 位，先左移 4 位再右移 4 位，删除第 2 字节的高 4 位，得到 0011 0111 1111。

若簇号为奇数，需要高 12 位，右移 4 位并与 0x0FFF 位与，删除第 1 字节的低 4 位，得到 0001 0011 0111。

4、读取 pdf 文件。

```
void getfile(FILE* fat12,int startClus,string name)
{
    FILE* outfile;
    outfile = fopen(name.c_str(), "ab");
    //数据簇开始
    int data_ptr = BytsPerSec * (RsvdSecCnt + FATSz * NumFATs + (RootEntCnt * 32 +
BytsPerSec - 1) / BytsPerSec);
    u16 nextClus;
    u16 Clus = startClus;
    while(Clus<0xFF8)
    {
        nextClus = getValue(fat12, Clus);
        if(nextClus==0xFF7)
        {
```

```
            cout << "bad cluster!" << endl;
            exit(1);
        }
        else
        {
            char* str = (char*)malloc(SecPerClus * BytsPerSec);
            char* content = str;
            int startByte = data_ptr + (Clus - 2) * SecPerClus * BytsPerSec;
            int flag;
            flag = fseek(fat12, startByte, SEEK_SET);
            if(flag==-1)
            {
                cout << "fseek error! line:171" << endl;
                exit(0);
            }
            flag = fread(content, 1, SecPerClus * BytsPerSec, fat12);
            if (flag != SecPerClus * BytsPerSec)
            {
                cout << "fread error! line:177" << endl;
                exit(0);
            }
            fwrite(str, 1, SecPerClus * BytsPerSec, outfile);
        }
        Clus = nextClus;
    }
    fclose(outfile);
}
```

由于 pdf 文件特有的文件结构，用二进制的方式写入才不会损坏文件。使用 fwrite 函数，从每一簇读取二进制信息，追加到文件末尾。

5、读取其他文本文件

```
void getother(FILE* fat12, int startClus, string name)
{
    ofstream outfile(name, ios::app | ios::binary);
    if (!outfile.is_open())
    {
        cout << "outfile open error!" << endl;
        exit(0);
    }
    //数据簇开始
    int data_ptr = BytsPerSec * (RsvdSecCnt + FATSz * NumFATs + (RootEntCnt * 32 +
BytsPerSec - 1) / BytsPerSec);
    u16 nextClus;
```

```
    u16 Clus = startClus;
    while (Clus < 0xFF8)
    {
        nextClus = getValue(fat12, Clus);
        if (nextClus == 0xFF7)
        {
            cout << "bad cluster!" << endl;
            exit(1);
        }
        else
        {
            char* str = (char*)malloc(SecPerClus * BytsPerSec);
            char* content = str;
            int startByte = data_ptr + (Clus - 2) * SecPerClus * BytsPerSec;
            int flag;
            flag = fseek(fat12, startByte, SEEK_SET);
            if (flag == -1)
            {
                cout << "fseek error!" << endl;
                exit(0);
            }
            flag = fread(content, 1, SecPerClus * BytsPerSec, fat12);
            if (flag != SecPerClus * BytsPerSec)
            {
                cout << "fread error! ;line:222" << endl;
                exit(0);
            }
            cout << str << endl;
            outfile << str;
        }
        Clus = nextClus;
    }
    outfile.close();
}
```

由于 fopen 的方式不能正确识别文本文件的结束，必须采用字符的方式写入文本文件。因此选择字符串的形式写入。

三、主函数

1、定义、初始化并写入全局变量

```
FILE* fat12;
    fat12 = fopen("fat12.img", "rb");

    struct BPB bpb;
```

```
    struct BPB* bpb_ptr = &bpb;
    struct RootEntry rootEntry;
    struct RootEntry* rootEntry_ptr = &rootEntry;

    getBPBvalue(fat12, bpb_ptr);
    //写入全局变量
    BytsPerSec = bpb_ptr->BPB_BytsPerSec;
    SecPerClus = bpb_ptr->BPB_SecPerClus;
    RsvdSecCnt = bpb_ptr->BPB_RsvdSecCnt;
    NumFATs = bpb_ptr->BPB_NumFATs;
    RootEntCnt = bpb_ptr->BPB_RootEntCnt;
    //FATSz可能出现在两处
    if (bpb_ptr->BPB_FATSz16 != 0)
    {
        FATSz = bpb_ptr->BPB_FATSz16;
    }
    else
    {
        FATSz = bpb_ptr->BPB_TotSec32;
    }

    string *filenames=(string*)new string[RootEntCnt];
    int *filelocation = (int*)new int[RootEntCnt];          //创建文件索引
    memset(filelocation, -1, RootEntCnt);                   //初始化为-1
```

2、遍历根目录，读取文件名并打印在终端以供选择。

```
getFilename(fat12,rootEntry_ptr,filelocation);

    int flag,idx = 0;

    while (filelocation[idx] != -1)
    {
        char* name = new char[12];
        flag = fseek(fat12, (RsvdSecCnt + NumFATs * FATSz) * BytsPerSec +
filelocation[idx] * 32, SEEK_SET);
        if (flag == -1)                 //错误处理
        {
            cout << "fseek error! line:275" << endl;
            exit(0);
        }
        flag = fread(rootEntry_ptr, 1, 32, fat12);
        if (flag != 32)
        {
            cout << "fread error! line:281" << endl;
```

```
            exit(0);
        }
        int temp = -1;
        for (int k = 0; k < 11; k++)
        {
            if (rootEntry_ptr->DIR_Name[k] != ' ') {
                temp++;
                name[temp] = rootEntry_ptr->DIR_Name[k];
            }
            else {
                temp++;
                name[temp] = '.';
                while (rootEntry_ptr->DIR_Name[k] == ' ')
                    k++;
                k--;
            }
        }
        temp++;
        name[temp] = '\0';
        for (int i = 0; i < temp; i++)
        {
            filenames[idx] += name[i];
        }
        cout << idx<<":"<<name << endl;
        idx++;
        delete[]name;
    }
    filecount = idx;
```

3、根据输入提取文件，同时文本文件打印在终端。

```
int select;
    cout << "Input the number of file you want to read:" << endl;
    cin >> select;
    while (select >= 0 && select < filecount) {
        flag = fseek(fat12, (RsvdSecCnt + NumFATs * FATSz) * BytsPerSec +
filelocation[select] * 32, SEEK_SET);
        flag = fread(rootEntry_ptr, 1, 32, fat12);
        if (strstr(filenames[select].c_str(), "PDF"))
            getfile(fat12, rootEntry_ptr->DIR_FstClus, filenames[select]);
        else
            getother(fat12, rootEntry_ptr->DIR_FstClus, filenames[select]);
        cout << "Input the number of file you want to read:" << endl;
        cin >> select;
    }
```

```
    cout << "Input over!" << endl;
    delete[]filenames;
    return 0;
```

四、调试

1、禁用警告

```
#define _CRT_SECURE_NO_WARNINGS
```

在 vs 编译器中，fopen 被认为不安全，不能使用。在文件开头添加定义。

2、对齐方式

```
#pragma pack (1)     //1字节对齐
```

改变编译器的对齐方式。默认为 8 字节，用结构体读取数据将出现错误。

3、定位

```
cout << "fseek error! line:61" << endl;
```

调试过程中，由于偏移量的错误，fseek 和 fread 报错多，在输出中打印错误位置，便于调试。