文件管理实验报告

夏　梦　　**郑棵丹**

14061175　　14061147

助教：运明纯

2016. 5. 21

目录

ls

前提：对基础实验的修改即 fat 表项和其他基本定义，使得程序可以正常的读取文件内容：

```
#ifndef FILESYS_H
#define FILESYS_H
#include<stddef.h>
#define DEVNAME "data"
#define DIR_ENTRY_SIZE (32)
#define SECTOR_SIZE (512)
#define CLUSTER_SIZE (bdptor.BytesPerSector*bdptor.SectorsPerCluster)
#define FAT_SIZE (bdptor.SectorsPerFAT*bdptor.BytesPerSector)
#define FAT_ONE_OFFSET (SECTOR_SIZE*(bdptor.ReservedSectors))
#define FAT_TWO_OFFSET (FAT_ONE_OFFSET+FAT_SIZE)
#define ROOTDIR_OFFSET (FAT_TWO_OFFSET+FAT_SIZE)
#define DATA_OFFSET (ROOTDIR_OFFSET+bdptor.RootDirEntries*DIR_ENTRY_SIZE)
#define CLUSTER_NUM ((bdptor.LogicSectors*bdptor.BytesPersetor-DATA_OFFSET)/CLUSTER_SIZE)
```

基础要求：设计并实现一个目录列表函数（无须支持选项，如 ls -a、ls -l 等），用来显示当前目录下包含的文件信息。

```
功能：显示当前目录的内容
返回值：1，成功；-1，失败
/
nt fd_ls()
{
    int ret, offset,cluster_addr;
    struct Entry entry;
    unsigned char buf[DIR_ENTRY_SIZE];
    if( (ret = read(fd,buf,DIR_ENTRY_SIZE))<0)
        perror("read entry failed");
    if(curdir==NULL)
        printf("Root_dir\n");
    else
        printf("%s_dir\n",curdir->short_name);
    printf("\tname\tdate\t\t time\t\tcluster\tsize\t\tattr\n");

    if(curdir==NULL)  /*显示根目录区*/
    {
        /*将fd定位到根目录区的起始地址*/
        if((ret= lseek(fd,ROOTDIR_OFFSET,SEEK_SET))<0)
            perror("lseek ROOTDIR_OFFSET failed");

        offset = ROOTDIR_OFFSET;

        /*从根目录区开始遍历，直到数据区起始地址*/
        while(offset < (DATA_OFFSET))
        {
            ret = GetEntry(&entry);

            offset += abs(ret);
            if(ret > 0)
            {
                printf("%12s\t"
                       "%d:%d:%d\t"
                       "%d:%d:%d   \t"
                       "%d\t"
                       "%d\t\t"
                       "%s\n",
                       entry.short_name,
                       entry.year,entry.month,entry.day,
                       entry.hour,entry.min,entry.sec,
                       entry.FirstCluster,
                       entry.size,
                       (entry.subdir) ? "dir":"file");
            }
        }
    }

    else /*显示子目录*/
    {
        int current_cluster;                                                    1.
        for(current_cluster = curdir->FirstCluster;current_cluster!=0xffff;current_cluster =
RevByte(fatbuf[current_cluster<<1],fatbuf[current_cluster<<1|1])){
printf("%d %d\n",fatbuf[current_cluster<<1],fatbuf[current_cluster<<1|1]);
            cluster_addr = DATA_OFFSET + (current_cluster-2) * CLUSTER_SIZE ;
            if((ret = lseek(fd,cluster_addr,SEEK_SET))<0)
                perror("lseek cluster_addr failed");

            offset = cluster_addr;

            /*只读一簇的内容*/
            while(offset<cluster_addr +CLUSTER_SIZE)
            {
                ret = GetEntry(&entry);
                offset += abs(ret);
                if(ret > 0)
                {
                    printf("%12s\t"
                           "%d:%d:%d\t"
                           "%d:%d:%d   \t"
                           "%d\t"
                           "%d\t\t"
                           "%s\n",
                           entry.short_name,
                           ▪
                           ▪
                           ▪
```

提高要求：对 ud_ls()函数进行改进，增加对全部非根目录信息的读取（基本要求中仅读取一个扇区的非根目录细信息）：for 循环实现对全部簇的遍历然后再对其中的每一簇进行读取。[详见上图红框 1.]

cd

基础要求：设计并实现一个改变目录函数（无须处理路径名，如../../directoryName 等），用来把当前目录切换到上一层目录

```
/*
*参数：dir，类型：char
*返回值：1，成功；-1，失败
*功能：改变目录到父目录或子目录
*/
int fd_cd(char *dir)
{
        struct Entry *pentry,*curdir1;
        int ret,retu,dirno1;
        int i,flag = 0;

        if (dir[0] == '/'){
                curdir1 = curdir; dirno1 = dirno;
                curdir = NULL; dirno = 0; retu = fd_cd(dir+1);
                if (retu < 0) {curdir = curdir1; dirno = dirno1;}
                return retu;
        }
        for (i = 0; dir[i] && dir[i] != '/'; i++) ;
        printf("%d %c\n",i,dir[i]);
        if (dir[i] == '/'){
                if (dir[i+1] == '/') {printf("no such dir\n"); return -1;}
                flag = 1;
                dir[i] = '\0';
        }

        if(!strcmp(dir,".")){
                if (flag == 1) return fd_cd(dir+(i+1)); else return 1;
        }
        if(!strcmp(dir,"..") && curdir==NULL){
                if (flag == 1) return fd_cd(dir+(i+1)); else return 1;
        }
        /*返回上一级目录*/
        if(!strcmp(dir,"..") && curdir!=NULL)
        {
                curdir1 = curdir;
                curdir = fatherdir[dirno];
                dirno--;
                if (flag == 1) {
                        retu = fd_cd(dir+(i+1));
                        if (retu < 0) {
                                dirno++; curdir = curdir1;
                        }
                        return retu;
                }
                return 1;
        }
        pentry = (struct Entry*)malloc(sizeof(struct Entry));

        ret = ScanEntry(dir,pentry,1);
        if(ret < 0)
        {
                printf("no such dir\n");
                free(pentry);
                return -1;
        }
        dirno ++;
        fatherdir[dirno] = curdir;
        curdir = pentry;
        if (flag == 1) { retu = fd_cd(dir+(i+1));
        if (retu < 0) {
                curdir = fatherdir[dirno--];
                return retu;
        }
        return 1;
}
```

或当前目录的子目录。

提高要求：增加绝对路径和多级目录的支持：这里需要对输入的目录路径字符串进行解析，然后逐级查找目录。详见上图，flag 用来记录目录是否正确，如果当前这一级目录不正确，则返回上一级目录。

df

基础要求：设计并实现一个删除文件函数， 使用当前目录中的要删除的文件名作为参数，用来删除指定的文件，要注意文件的隐藏、只读和其他系统属性。

```
/*
*参数：filename，类型：char
*返回值：1，成功；-1，失败
*功能:删除当前目录下的文件
*/
int fd_df(char *filename)
{
        struct Entry *pentry;
        int ret,offset,cluster_addr,current_cluster;
        unsigned char c;
        unsigned short seed,next;

        pentry = (struct Entry*)malloc(sizeof(struct Entry));

        /*扫描当前目录查找文件*/
        ret = ScanEntry(filename,pentry,0);
        if(ret<0)
        {
                ret = ScanEntry(filename,pentry,1);
                if(ret<0){
                        printf("no such file\n");
                        free(pentry);
                        return -1;
                }
        }
        if (pentry->subdir){
                int ret;
                printf("%d\n",fd_cd(filename));
                for(current_cluster = curdir->FirstCluster;current_cluster!=0xffff;current_cluster =
RevByte(fatbuf[current_cluster<<1],fatbuf[current_cluster<<1|1])){
                        cluster_addr = DATA_OFFSET + (current_cluster-2) * CLUSTER_SIZE ;
                        if((ret = lseek(fd,cluster_addr,SEEK_SET))<0)
                                perror("lseek cluster_addr failed");

                        offset = cluster_addr;

                        while(offset<cluster_addr +CLUSTER_SIZE)
                        {
                                struct Entry entry;
                                ret = GetEntry(&entry);
                                offset += abs(ret);
                                if(ret > 0 && strcmp(entry.short_name,".") && strcmp
(entry.short_name,".."))
                                {
                                        puts(entry.short_name);
                                        fd_df(entry.short_name);
                                }
                        }

                }
                fd_cd("..");
        }
        /*清除fat表项*/
        seed = pentry->FirstCluster;
        while((next = GetFatCluster(seed))!=0xffff)
        {
                ClearFatCluster(seed);
                seed = next;

        }

        ClearFatCluster( seed );

        /*清除目录表项*/
        c=0xe5;


        if(lseek(fd,ret-0x20,SEEK_SET)<0)
                perror("lseek fd_df failed");
        if(write(fd,&c,1)<0)
                perror("write failed");


/*      if(lseek(fd,ret-0x40,SEEK_SET)<0)
                perror("lseek fd_df failed");
        if(write(fd,&c,1)<0)
                perror("write failed");
*/
        free(pentry);
        if(WriteFat()<0)
                exit(1);
        return 1;
}
```

提高要求：增加删除目录的功能：通常需要先判断目录是否为空目录，若目录不为空，则需给出提示，并删除其包含的所有子目录和文件；若是空目录则可以直接删除。详见上图，进入目录时判断其中是否存在非空目录，若存在，则进入下一级非空目录，若不存在，则将空目录视为一个文件，遍历当前目录，删除每一个文件。

cf

基础要求：设计并实现一个创建文件函数，使用要创建的文件名和文件大小作为参数，用来创建一个新的文件。

提高要求：对 ud_cf()函数进行改进，使其可以向文件中写入实际内容，并根据写入的内容计算文件的实际大小。在传参的时候顺便将要写入文件的内容传入，这样在创建文件的同时也将内容写入到创建的文件中去。将写入的内容和根据内容计算出的文件大小传入，然后将内容写入该文件中。（min 函数是用来判断看写入的大小是一簇还是内容大小，如果规定文件大小则按照文件大小否则按照写入文件内容来计算文件大小）

增加创建目录的功能[类同创建有内容文件]:

```
void fd_makedir(char *menuname)
{
        dirflag = 1;
        fd_cf(menuname,0);
        dirflag = 0;
}
```