

操作系统实践 lab7 实验报告

实验目的：

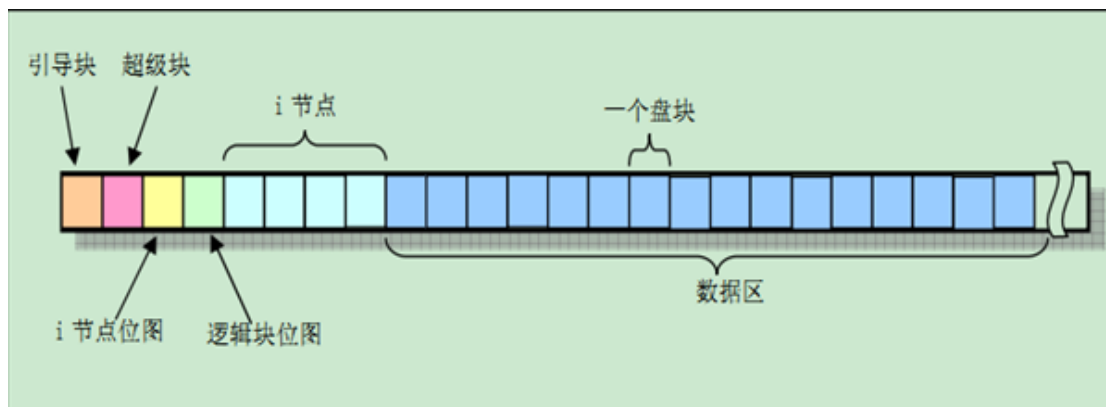
1. 熟悉文件系统理论
2. 了解文件系统主要的数据结构，并学习如何利用这些理论写一个文件系统

实验要求：

1. 设计并实现一个文件系统，拥有以下功能：
 - 1.1. 创造与删除一个文件：touch 与 rm
 - 1.2. 写与读文件
 - 1.3. 目录命令：mkdir, rmdir 与 cd
 - 1.4. 表示目录中的内容：ls

实验内容：

1. 文件系统的主要结构：
 - 1.1. 每个 disk 之中的结构基本示意如下：



其中包括一个 boot block，一个 superblock，一些 inodes 以及一些 data blocks。Root block 用于启动 disk，对于我的文件而言并不需要，所以 superblock 将被设为第 0 个 block。虽然这里将 bitmap 单独列出，但是我这里将 bitmap 被整理在 superblock 之中。

- 1.2. Superblock 的结构：

```

class superblock{
public:
    int inodes_count;        //the number of used inodes
    int blocks_count;        //the number of used blocks
    int free_blocks_count;    //the number of free blocks
    int free_inodes_count;    //the number of free inodes
    int first_data_block;     //the 1st data block
    int magic;                //the check whether the disk is correct format
    int block_size;           //the size of block
    char bitmap[BLOCK_NUM/2]; //the bitmap
    int set_bitmap_true(int);
    int set_bitmap_false(int);
    bool get_bitmap(int);
    int get_size(int);
    int get_empty_inode(void);
    int get_empty_block(void);
};

```

可以到这里有用于计算 inodes 以及 blocks 数量的变量，剩余 inodes 与 block 数量的变量，以及记录第一个 data block 位置的变量。Magic 用于存 magic 值，在我的文件系统中初始化时将被设为 0xef53，与 ext2 的 magic 值相同。遇过 magic 值对不上说明不是我的文件系统格式化的 disk。Block_size 用于记录 block 的大小。初始化时将被设为 1024，即 1kb。Bitmap 数组即用于完成 bitmap 作用，为了方便起见在我的文件系统整合在了 superblock 中。很多书中是位于 superblock 跟 inodes 之间。下面的成员函数将会将在之后详细说明。

1.3. Inode 的结构:

```

..
class inode{
public:
    int size;                //the number of directories in this inode,including . and ..
    int data;                //record the place of beginning address of the data of this inode in blocks
    bool isroot;             //return true when root
    bool isdir;              //return true when directory
    int block;               //record the place of this inode in blocks
};

```

size 用于记录目录 inode 中子目录的数量(包括了“.”与“..”), data 用于记录文件 inode 中其 data block 的为第几个 block。Isroot 与 isdir 分别判断是否为根目录和是否为目录。Block 用于记录这个 inode 本身为第几个 block。

1.4. Inode_map 的结构:

```

struct inode_map{
    char filename[12];        //the name of subdirectories
    int inumber;              //record the place of subdirectories in blocks
};

```

这里的 inode_map 讲吧保存某个目录 inode 的某个子目录及其信息。Filename 为记录该子目录的名称，inumber 为记录该子目录为第几个 block。

1.5. 其他:

```

typedef char data_block[DATA_SIZE];    //to record the data
typedef inode_map dir_block[TABLE_SIZE]; //the subdirectories of a directory

```

data_block 只是为了方便起见，讲一个 DATA_SIZE 长度的 char 进行重定义。同样的 dir_block 用于表示一个目录下的若干子目录以及文件，主要是用于表述当前目录下的子目录和文件的集合。

2. 一些全局变量的说明:

```

superblock nowsb;    //the current superblock
inode nowin;         //the current directory
dir_block nowdir;    //the subdirectories of current directory
FILE * f;            //for write and read from disk

```

nowsb 表示当前的 superblock（其实一直就是一个）。Nowin 表示当前的目录的信息。而 nowdir 表示当前目录的所有子目录和文件的一些基本信息。

3. Superblock 中的成员函数的实现:

3.1. set_bitmap_true:

```

int superblock::set_bitmap_true(int p)
{
    int x=p/2;
    int y=p%2;
    if(y==1)
        bitmap[x] |= 0x01;
    else
        bitmap[x] |= 0x10;
}
//make the pth block true in bitmap

```

使得第 p 个 block 的 bitmap 值为 1。由于一个 char 型占有 16 位，而我们事实上只需要 1 位就足够表示 0 和 1 了。[0 1][2 3][4 5]...如这般的排列，其中一个[]之中为一个 char 型数据，偶数占高位，奇数占低位。

3.2. set_bitmap_false:

```

int superblock::set_bitmap_false(int p)
{
    int x=p/2;
    int y=p%2;
    if(y==1)
        bitmap[x] &= 0x10;
    else
        bitmap[x] &= 0x01;
}
//make teh pth block false in bitmap

```

使得第 p 个 block 的 bitmap 值为 0。基本思路基本同 set_bitmap_true，便不再赘述。

3.3. get_bitmap:

```

bool superblock::get_bitmap(int p)
{
    int x=p/2;
    int y=p%2;
    if(y==1)
        return bitmap[x] & 0x01;
    else
        return (bitmap[x] & 0x10) >> 1;
}
//get the value of bitmap of pth block

```

获得第 p 个位置的 bitmap 值。同样是偶数在高位，奇数在低位，这样便能利用右移跟或完整返回值。

3.4. get_size:

```
int superblock::get_size(int p)
{
    return p*block_size;
}
//for seek to the position of pth block
```

获得第 p 个 block 在 disk 的二进制文件中的具体位置。只需要将 block_size 乘以 p 就能得到在二进制文件中这个 block 的起始地址。这个函数将被使用在 fseek 的第二个参数的获得中。

3.5. get_empty_block:

```
int superblock::get_empty_block()
{
    for(int i=0;i<BLOCK_NUM;i++)
        if(!this->get_bitmap(i))
        {
            blocks_count++;
            free_blocks_count--;
            return i;
        }
    SetColor(RED);
    printf("there has been no empty block to create\n");
    SetColor(GRAY);
    return -1;
}
//get a empty block
```

用于寻找一个空的 block。利用 get_bitmap() 函数不断寻找是否有返回值为 false 的 bitmap 从而确定一个空闲的 block。如果有的话就将第几个 block 的值返回，否则输出警告并返回 -1。因为在添加文件的时候没有更改，所以这个函数同时承担了对 block 跟 free_block 数的更新。

3.6. get_empty_inode:

```
int superblock::get_empty_inode()
{
    if(this->free_inodes_count<=0)
    {
        SetColor(RED);
        printf("there has been no empty space to create an inode\n");
        SetColor(GRAY);
        return -1;
    }
    for(int i=0;i<TABLE_SIZE;i++)
    {
        if(strlen(nowdir[i].filename)==0)
        {
            inodes_count++;
            free_inodes_count--;
            return i;
        }
    }
    SetColor(RED);
    printf("there has been no empty space to create an inode\n");
    SetColor(GRAY);
    return -1;
}
//get a empty inode
```

寻找一个新的 inode。如果 free_inode_count 已经小于等于 0，说明肯定没有了。如果大于 0，则需要通过遍历来寻找空闲的 inode，并将其为第几个 block 返回。如果出错则警告并返回-1。因为该在添加文件的时候没有修改，所以这个函数同时承担了对 inodes 以及 free_inodes 数的更新。

4. 两个需要说明的函数:

4.1. set_inode_table:

```
int superblock::set_inode_table(int i, char * name, int bnum)
{
    strcpy(nowdir[i].filename, name);
    nowdir[i].inumber=bnum;
    return 1;
}
//used in creating directory
```

这个函数将用于新建一个目录的时候，讲这个新的子目录加入到新的 nowdir 数组中。

4.2. name_exist:

```
int name_exist(char *name)
{
    for(int i=0;i<TABLE_SIZE;i++)
    {
        if(strcmp(nowdir[i].filename,name)==0)
            return i;
    }
    return -1;
}
//to check whether the name has been used
```

这个函数用于判断是否这个名字已经被使用过了。由于 nowdir 中存储的是当前的目录下的子目录以及文件，所以只需要 nowdir 中的所有的元素就能够的到结构。如果存在则返回这个文件或子目录的标号，否则返回-1。

5. disk 的创造和载入:

5.1. disk 的创立: format()

```
int format(char filename[]) //to format the disk
{
    superblock sb;
    inode root;
    dir_block root_block;
    sb.block_size=BLOCK_SIZE;
    sb.blocks_count=3;           //the superblock, the inode of root and the block of root.
    sb.inodes_count=1;          //the inode of root
    sb.free_blocks_count=BLOCK_SIZE-sb.blocks_count;
    sb.free_inodes_count=BLOCK_SIZE*INODE_RATIO-sb.inodes_count;
    sb.first_data_block=1;
    sb.magic=EXT2_MAGIC;        //for checking whether the disk is correct
    memset(sb.bitmap,0,sizeof(sb.bitmap));
    nowsb=sb;
    nowsb.set_bitmap_true(0);
    nowsb.set_bitmap_true(1);
    nowsb.set_bitmap_true(2);
    f=fopen(filename,"wb+");
    if(!f)
    {
        SetColor(RED);
        printf("cannot open the file\n");
        SetColor(GRAY);
        return -1;
    }
}
```

```

fseek(f,0,SEEK_SET);
fwrite(&nowsb,sizeof(superblock),1,f); //set the superblock as the 0th block
root.isroot=true;
root.isdir=true;
root.size=1;
root.data=2;
root.block=1;
fseek(f,1*nowsb.block_size,SEEK_SET);
fwrite(&root,sizeof(inode),1,f);
memset(root_block,0,sizeof(root_block));
strcpy(root_block[0].filename,"."); //point to root itself
root_block[0].inumber=1; //set the root as the 1st block
fseek(f,2*nowsb.block_size,SEEK_SET);
fwrite(&root_block,sizeof(dir_block),1,f);
data_block d;
memset(d,0,sizeof(d));
fseek(f,3*nowsb.block_size,SEEK_SET);
for(int i=3;i<nowsb.blocks_count+nowsb.free_blocks_count;i++)
{
    fwrite(&d,sizeof(data_block),1,f);
}
SetColor(YELLOW);
printf("block_size=%d\n",nowsb.block_size); //print the status of the disk
printf("inodes_count=%d\tblocks_count=%d\n",nowsb.inodes_count,nowsb.blocks_count);
printf("free_inodes_count=%d\tfree_block_count=%d\n",nowsb.free_inodes_count,nowsb.free_blocks_count);
printf("format ok!\n");
SetColor(GRAY);
fclose(f);
return 1;
}

```

这个函数用于创造或者格式化一个 disk。一开始创造的时候包括了三个 blocks，包括了 superblock 本身以及 root 根目录的 inode 以及 block。其中 magic 值是用于当载入的时候是否是格式正确。根目录之下只有一个子目录，就是“.”，也就是根目录自身。写完文件后打印出这个 disk 的信息。

5.2. disk 的载入：load()

```

int load(char filename[]) //to open the disk
{
    superblock sb;
    inode root;
    dir_block root_block;
    f=fopen(filename,"rb+");
    if(!f)
    {
        SetColor(RED);
        printf("cannot open the file\n");
        SetColor(GRAY);
        return -1;
    }
    fseek(f,0,SEEK_SET);
    fread(&sb,sizeof(superblock),1,f);
    if(sb.magic!=EXT2_MAGIC) //check the magic value
    {
        SetColor(RED);
        printf("not correct disk!\n");
        SetColor(GRAY);
        return -1;
    }
    nowsb=sb;
    fseek(f,nowsb.get_size(nowsb.first_data_block),SEEK_SET);
    fread(&root,sizeof(inode),1,f);
    if (!root.isdir)
    {
        SetColor(RED);
        printf("not a directory!\n");
        SetColor(GRAY);
        return -1;
    }
}

```

```

    nowsb=sb;
    fseek(f,nowsb.get_size(nowsb.first_data_block),SEEK_SET);
    fread(&root,sizeof(inode),1,f);
    if (!root.isdir)
    {
        SetColor(RED);
        printf("not a directory!\n");
        SetColor(GRAY);
        return -1;
    }
    nowin=root;          //make the current directory inode as the root
    fseek(f,nowsb.get_size(root.data),SEEK_SET);
    fread(&root_block,sizeof(dir_block),1,f);
    for(int i=0;i<TABLE_SIZE;i++)
        nowdir[i]=root_block[i];
    SetColor(YELLOW);
    printf("block_size=%d\n",nowsb.block_size);    //print the status of the disk
    printf("inodes_count=%d\tblocks_count=%d\n",nowsb.inodes_count,nowsb.blocks_count);
    printf("free_inodes_count=%d\tfree_block_count=%d\n",nowsb.free_inodes_count,nowsb.free_blocks_count);
    printf("load file ok!\n");
    SetColor(GRAY);
    return 1;
}

```

载入一个 disk。首先要判断文件是否存在。若存在则读入第 0 个 block，即 superblock 对应的位子。如果其 magic 值为 EXT2_MAGIC，则为正确格式的文件。通过 first_data_block 得到根目录所在的 block 的位子。从而可以获得根目录的信息，并将其设置为 nowin，即现在所在的目录。最后打印出这个 disk 的各项信息。

6. 目录操作:

6.1. 创立目录: create_dir():

```

int create_dir(char dirname[]) //for the command "mkdir"
{
    int space,inode_ptr,newdir_ptr;
    inode in;
    dir_block newdir;
    if(strcmp(dirname,"..")==0||strcmp(dirname,".")==0)
    {
        SetColor(RED);
        printf("cannot create a directory called %s\n",dirname);
        SetColor(GRAY);
        return -1;
    }//the .. and . directories cannot be created
    else if(name_exist(dirname)!=-1)
    {
        SetColor(RED);
        printf("directory with name \"%s\" has existed\n",dirname);
        SetColor(GRAY);
        return -1;
    }
    space=nowsb.get_empty_inode();
    if(space==-1)
        return -1;          //no empty inode
    inode_ptr=nowsb.get_empty_block();
    if(inode_ptr==-1)
        return -1;          //no empty block
    set_inode_table(space,dirname,inode_ptr);
    nowsb.set_bitmap_true(inode_ptr);          //set the bitmap
    nowin.size++;
    in.isroot=false;
    in.isdir=true;
    in.block=inode_ptr;
}

```

```

newdir_ptr=newsb.get_empty_block();           //get a data block
newsb.set_bitmap_true(newdir_ptr);           //set the bitmap for data block
if(newdir_ptr==-1)
    return -1;
in.data=newdir_ptr;
in.size=2;                                   //including . and .. directories
memset(newdir,0,sizeof(newdir));
strcpy(newdir[0].filename,".");              //point to the new directory itself
newdir[0].inumber=inode_ptr;
strcpy(newdir[1].filename,"..");             //point to the father directory
newdir[1].inumber=nowin.block;

fseek(f,0,SEEK_SET);
fwrite(&newsb,sizeof(superblock),1,f);

fseek(f,newsb.get_size(nowin.block),SEEK_SET);
fwrite(&nowin,sizeof(inode),1,f);

fseek(f,newsb.get_size(nowin.data),SEEK_SET);
fwrite(nowdir,sizeof(dir_block),1,f);

fseek(f,newsb.get_size(inode_ptr),SEEK_SET);
fwrite(&in,sizeof(inode),1,f);

fseek(f,newsb.get_size(newdir_ptr),SEEK_SET);
fwrite(newdir,sizeof(dir_block),1,f);
fflush(f);
return 1;
}

```

这个函数英语指令“mkdir XXX”，首先判断目录名是否为“.”或者“..”，如果是则返回-1。然后判断该名字是否已经被使用过。如果没有的话，则申请一个 inode，一个 data block，并设置各自的 bitmap 对应的值为 1。加入讲个子目录，分别为“.”与“..”。最后将内容写入到 disk 中，并刷新文件流。

6.2. 删除目录：remove_dir():

```

int remove_dir(char dirname[])               //for the command "rmdir"
{
    int ptr;
    inode in;
    ptr=name_exist(dirname);
    if((ptr<2&&(ptr==0)||!(ptr==1&&nowin.isroot==true))||strcmp(dirname,"..")==0||strcmp(dirname,".")==0)
    {
        SetColor(RED);//
        printf("the father and directory itself cannot be removed!\n");
        SetColor(GRAY);
        return -1;
    }//the .. and . directory cannot be removed
    if(ptr==1)
    {
        SetColor(RED);
        printf("directory %s does not exist!\n",dirname);
        SetColor(GRAY);
        return -1;
    }//check the name whether exists
    fseek(f,newsb.get_size(nowdir[ptr].inumber),SEEK_SET);
    fread(&in,sizeof(inode),1,f);
    if(!in.isdir)
    {
        SetColor(RED);
        printf("%s is not a directory!\n",dirname);
        SetColor(GRAY);
        return -1;
    }//if not a file ,return
}

```



```

    if(in.size!=2)
    {
        SetColor(RED);
        printf("there are still files in directory %s\n",dirname);
        SetColor(GRAY);
        return -1;
    }//can just remove the directory with nothing inside
    nowsb.set_bitmap_false(in.data);
    strcpy(nowdir[ptr].filename,"");
    nowsb.set_bitmap_false(nowdir[ptr].inumber);
    nowdir[ptr].inumber=0;
    nowin.size--;           //the number of size decreased by one
    nowsb.blocks_count-=2;  //the number of blocks decreased by two
    nowsb.inodes_count--;   //the number of inodes decreased by one
    nowsb.free_blocks_count+=2; //meanwhile the free_blocks and free_inodes increased
    nowsb.free_inodes_count++;

    fseek(f,0,SEEK_SET);
    fwrite(&nowsb,sizeof(superblock),1,f);

    fseek(f,nowsb.get_size(nowin.block),SEEK_SET);
    fwrite(&nowin,sizeof(inode),1,f);

    fseek(f,nowsb.get_size(nowin.data),SEEK_SET);
    fwrite(nowdir,sizeof(dir_block),1,f);
    fflush(f);
    return 1;
}

```

这个函数将被用在“rmdir XXX”指令中。首先判断目录名是否为“.”与“..”，并且是否存在。然后再是否为一个目录，再判断是否为空目录（这个文件系统只能删除空目录，并没有实现递归删除）。如果均为合法的话就将这个目录的 inode 以及 data block 的 bitmap 设为 0，将当前目录的子目录以及文件，即 nowdir 中的对应项的文件名设为“”，即空字符串，表示已经删除。并对 superblock 中的 inode，block 以及空闲的 inode，block 数进行修改。最后将文件写入 disk 中。

6.3. 改变目录路径：change_dir():

```

int change_dir(char dirname[]) //for the command "cd"
{
    int ptr;
    inode in;
    ptr=name_exist(dirname);
    if(ptr==-1)
    {
        SetColor(RED);
        printf("directory %s does not exist\n",dirname);
        SetColor(GRAY);
        return -1;
    }
    fseek(f,nowsb.get_size(nowdir[ptr].inumber),SEEK_SET);
    fread(&in,sizeof(inode),1,f);
    if(in.isdir==false)
    {
        SetColor(RED);
        printf("%s is not a directory\n",dirname);
        SetColor(GRAY);
        return -1;
    }
    nowin=in;
    fseek(f,nowsb.get_size(nowin.data),SEEK_SET);
    fread(nowdir,sizeof(dir_block),1,f);
    return 1;
}

```

用于命令“cd XXX”。首先同样是判断是否存在这个名字的子目录或者文件。通过 nowdir 对子目录以及文件的记录，可以得到每个子目录对应的第几个 block。从而可以找到这个子目录在 disk 中的对应的 inode。如果不是目录则返回错误。如果均合法，只需要将目前的目录 nowin 更改为这个子目录，并将其对应的 data block 中包含的子目录及文件的信息存入 nowdir 中即可。

7. 文件操作

7.1. 创建文件：create_file():

```
int create_file(char filename[])    //for the command "touch"
{
    int space;
    int inode_ptr,newfile_ptr;
    inode in;
    data_block newdata;
    if(strcmp(filename,"..")==0||strcmp(filename,".")==0)    //file .. and . should not be created
    {
        SetColor(RED);
        printf("cannot create a file called %s\n",filename);
        SetColor(GRAY);
        return -1;
    }
    else if(name_exist(filename)!=-1)
    {
        SetColor(RED);
        printf("file with name \"%s\" has existed\n",filename);
        SetColor(GRAY);
        return -1;
    }
    space=nowsb.get_empty_inode();    //get a empty inode
    if(space==-1)
        return -1;
    inode_ptr=nowsb.get_empty_block();    //get the block of the inode
    if(inode_ptr==-1)
        return -1;
    set_inode_table(space,filename,inode_ptr);
    nowsb.set_bitmap_true(inode_ptr);    //set the bitmap of inode true
    nowin.size++;
    in.isroot=false;
    in.isdir=false;    //not a directory,but a file
    in.block=inode_ptr;

    newfile_ptr=nowsb.get_empty_block();    //get a block for data
    if(newfile_ptr==-1)
        return -1;
    in.data=newfile_ptr;
    nowsb.set_bitmap_true(newfile_ptr);    //set the data block true
    memset(newdata,0,sizeof(newdata));
    printf("input the content\n");    //get the content of the file
    fflush(stdin);
    gets(newdata);

    fseek(f,0,SEEK_SET);
    fwrite(&nowsb,sizeof(superblock),1,f);

    fseek(f,nowsb.get_size(nowin.block),SEEK_SET);
    fwrite(&nowin,sizeof(inode),1,f);

    fseek(f,nowsb.get_size(nowin.data),SEEK_SET);
    fwrite(nowdir,sizeof(dir_block),1,f);

    fseek(f,nowsb.get_size(inode_ptr),SEEK_SET);
    fwrite(&in,sizeof(inode),1,f);

    fseek(f,nowsb.get_size(newfile_ptr),SEEK_SET);
    fwrite(newdata,sizeof(data_block),1,f);
    fflush(f);
    return 1;
}
```

这个函数用于指令“touch XXX”。同样的，首先判断是否为“.”与“..”，以及文件名是否已经存在。如果均合法，则为 inode 以及 data block 均申请一个 block。并将两者的 bitmap 设为 1。因为是文件，所以 isroot 以及 isdir 均为 false。在创建文件的时候就要求输入文件的内容，并记录到了 data block 中。以回车键位结束符。最后将修改后的信息写入 disk，并刷新文件流。

7.2. 写文件：write_file():

```
int write_file(char filename[])//for the command "write" to change the content of the file
{
    data_block newdata;
    int ptr;
    inode in;
    ptr=name_exist(filename);
    if(ptr==-1)
    {
        SetColor(RED);
        printf("file %s does not exist!\n",filename);
        SetColor(GRAY);
        return -1;
    }
    //judge the name whether exit
    fseek(f,nowsb.get_size(nowdir[ptr].inumber),SEEK_SET);
    fread(&in,sizeof(inode),1,f);
    if(in.isdir==true)        //judge whether it is a file
    {
        SetColor(RED);
        printf("%s is not a file!\n",filename);
        SetColor(GRAY);
        return -1;
    }
    //this function only for file
    printf("input the content\n");
    gets(newdata);            //get content of file
    fseek(f,nowsb.get_size(in.data),SEEK_SET);
    fwrite(newdata,sizeof(data_block),1,f);
    fflush(f);
    return 1;
}
```

这个函数用于指令“write XXX”，首先要判断这个文件是否存在，并且是否是文件而非目录。如果均合法，则读入新的内容并修改 data block 对应的内容。

7.3. 删除文件：remove_file():

```
int remove_file(char filename[])    //for command "rm"
{
    int ptr;
    inode in;
    ptr=name_exist(filename);
    if(ptr==-1)
    {
        SetColor(RED);
        printf("file %s does not exist!\n",filename);
        SetColor(GRAY);
        return -1;
    }
    fseek(f,nowsb.get_size(nowdir[ptr].inumber),SEEK_SET);
    fread(&in,sizeof(inode),1,f);
    if(in.isdir)
    {
        SetColor(RED);
        printf("%s is not a file!\n",filename);
        SetColor(GRAY);
        return -1;
    }
    nowsb.set_bitmap_false(in.data);                //set the bitmap of data block false
    strcpy(nowdir[ptr].filename,"");                //delete the file from current directory
    nowsb.set_bitmap_false(nowdir[ptr].inumber);    //set the bitmap of inode false
    nowdir[ptr].inumber=0;
    nowin.size--;
    nowsb.blocks_count-=2;
    nowsb.free_blocks_count+=2;
    nowsb.inodes_count--;
    nowsb.free_inodes_count++;
}
```

```

fseek(f,0,SEEK_SET);
fwrite(&nowsb,sizeof(superblock),1,f);

fseek(f,nowsb.get_size(nowin.block),SEEK_SET);
fwrite(&nowin,sizeof(inode),1,f);

fseek(f,nowsb.get_size(nowin.data),SEEK_SET);
fwrite(&nowdir,sizeof(dir_block),1,f);
fflush(f);
}

```

这个函数用于指令“rm XXX”。跟目录删除相似，首先判断是否存在，并且是否确实为文件。如果均合法，则将 data block 以及 inode 对应的 bitmap 均设为 0。然后将 nowdir 中记录的对应文件的文件名设为空字符串吗，表示已经从当前目录中删除。并对 superblock 中的 blocks, inodes，以及空余 blocks 和 inodes 的数量进行更新。最后写入 disk 并刷新文件流。

7.4. 显示文件：concatenate():

```

int concatenate(char filename[])    //for the command "cat"
{
    int ptr;
    inode in;
    data_block data;
    ptr=name_exist(filename);
    if(ptr==-1)
    {
        SetColor(RED);
        printf("there is no file called %s\n",filename);
        SetColor(GRAY);
        return -1;
    }
    fseek(f,nowsb.get_size(nowdir[ptr].inumber),SEEK_SET);
    fread(&in,sizeof(inode),1,f);
    if(in.isdir==true)
    {
        SetColor(RED);
        printf("%s is not a file\n",filename);
        SetColor(GRAY);
        return -1;
    }
    fseek(f,nowsb.get_size(in.data),SEEK_SET);
    fread(&data,sizeof(data_block),1,f);
    printf("%s\n",data);
    return 1;
}

```

用于指令“cat XXX”，将文件中的内容显示出来。只需要首先判断这个文件名是否存在，如果存在再判断是否为文件。如果均合法则取到其 data block 对应的位子，读取数据并打印即可。

8. 展示目录中的内容

8.1. 展示内容：list():

```

int list(void)           //for the command "ls"
{
    inode in;
    int bias=nowin.isroot?1:2;           //check whether a root
    SetColor(BLUE);
    if(bias==1)
        printf(".\t");
    else
        printf(".\t|.\t");
    SetColor(GRAY);
    for(int i=bias;i<TABLE_SIZE;i++)
    {
        if(strlen(nowdir[i].filename)!=0)
        {
            fseek(f,nowsb.get_size(nowdir[i].inumber),SEEK_SET);
            fread(&in,sizeof(inode),1,f);
            if(in.isdir)
            {
                SetColor(BLUE);
                printf("%s\t",nowdir[i].filename);
                SetColor(GRAY);
            }
            else
                printf("%s\t",nowdir[i].filename);
        }
    }
    printf("\n");
    return 1;
}

```

用于“ls”指令。只需要将 nowdir 中文件名（或子目录名）不为空的项一个个打印即可。其中文件夹将被蓝色表示，而普通文件将是初始设定的灰色。

9. 两个用于命令以及当前路径处理的函数：

9.1. 用于分隔字符串的函数：split():

```

void split(char cmd[],char x[],char y[])    //to split a string by a ' '
{
    int flag=0;
    int j=0;
    for(int i=0;cmd[i]!='\0';i++)
    {
        if(flag==0&&cmd[i]==' ')
        {
            flag=1;
            x[i]='\0';
            continue;
        }
        if(flag==0)
            x[i]=cmd[i];
        else
            y[j++]=cmd[i];
    }
    y[j]='\0';
}

```

用于除了“ls”以外的各种“XXX XXX”形式的指令，以第一个空格为分隔符将一个 command 分为两个字符串，分别为一个指令以及指令的目标。

9.2. 用于是路径退回的函数：previous_dir():

```

void previous_dir(char dir[])          //the make the directory address right when "cd .."
{
    int i;
    int j;
    for(i=0;dir[i]!='\0';i++);
    for(j=i-2;j>=0;j--)
    {
        if(dir[j]=='\\')
        {
            dir[j+1]='\0';
            //printf("j=%d\n",j);
            break;
        }
    }
}

```

一个字符串将保留当前的路径，有“cd ..”指令时，我们需要将这个路径缩短到上一个路径，所以以倒数第二个‘\’为界限（最后一个‘\’是一定有的）将路径缩短。

10. 命令的处理

10.1. 命令处理：console():

```

void console()          //the console
{
    char cmd[300];
    bool flag=false;
    char current_dir[50]="\\";
    while(true)
    {
        fflush(stdin);
        //printf("%s\n",cmd);
        if(flag==false)
        {
            printf("input \"create diskname\" to create a new disk\n");          //create new disk and format
            printf("input \"load diskname\" to load a disk\n");          //load the disk
        }
        if(flag==false)
            printf(">>>");
        else
        {
            printf("%s",current_dir);          //print the address of current directory
            printf(">>");
        }
        fflush(stdin);
        gets(cmd);
        char x[100],y[100];
        split(cmd,x,y);
        if(flag==false&&strcmp(x,"create")==0)
        {
            if(format(y)==-1)
                continue;
        }
    }
}

```

```

else if(flag==false&&strcmp(x,"load")==0)
{
    if(load(y)==-1)
        continue;
    flag=true;
}
else if(flag==true)
{
    if(strcmp(cmd,"q")==0||strcmp(cmd,"quit")==0)
    {
        printf("byebye :-)\n");
        return;
    }
    if(strcmp(cmd,"ls")==0)
        list();
    else
    {
        split(cmd,x,y);
        if(strcmp(x,"touch")==0)
            create_file(y);
        else if(strcmp(x,"rm")==0)
            remove_file(y);
        else if(strcmp(x,"cat")==0)
            concatenate(y);
        else if(strcmp(x,"mkdir")==0)
            create_dir(y);
        else if(strcmp(x,"rmdir")==0)
            remove_dir(y);
        else if(strcmp(x,"write")==0)
            write_file(y);

        else if(strcmp(x,"cd")==0)
        {
            if(strcmp(y,".")==0||change_dir(y)==-1)
                continue;
            if(strcmp(y,"..")!=0)
            {
                strcat(current_dir,y);
                strcat(current_dir,"\\");
            }
            else
            {
                previous_dir(current_dir);
            }
        }
        else
        {
            SetColor(RED);
            printf("illegal command!\n");
            SetColor(GRAY);
        }
    }
}
}
}

```

用于解读输入的命令。首先需要 load 一个 disk（若没有需要 create），如果输入不为“q”或“quit”（表退出），则利用 split 函数将命令分解，并调用相应的函数。

11. 颜色处理:

11.1. 全局变量，enum 型数据以及函数设定：

```
HANDLE hCon;

enum Color { DARKBLUE = 1, DARKGREEN, DARKTEAL, DARKRED, DARKPINK,
             DARKYELLOW, GRAY, DARKGRAY, BLUE, GREEN, TEAL, RED, PINK, YELLOW, WHITE };

void SetColor(Color c){
    if(hCon == NULL)
        hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hCon, c);
}
// to change the color of words
```

利用这个 SetColor 便能改变 console 的属性，从而是输出的颜色改变。

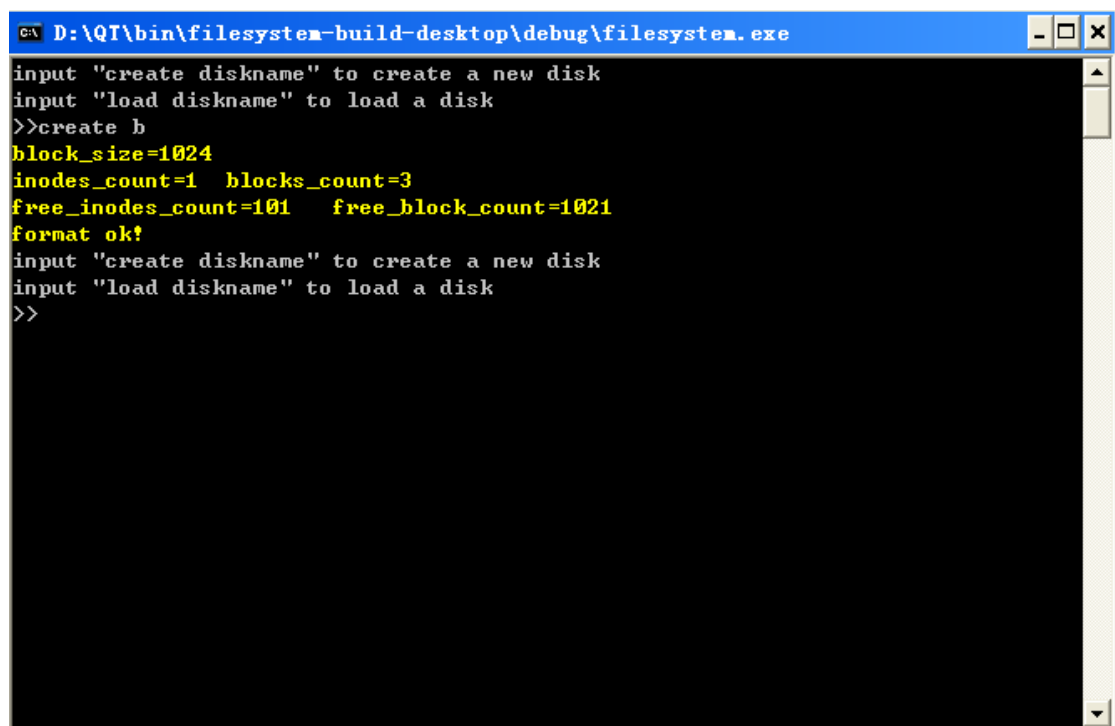
11.2. 颜色设定：

其中一开始的 disk 的信息为黄色，所有的错误以及警示为红色，ls 指令下所有的文件夹为蓝色，其他没有说明均为初始设定的灰色。

实验结果

1. Disk 的载入与初始：

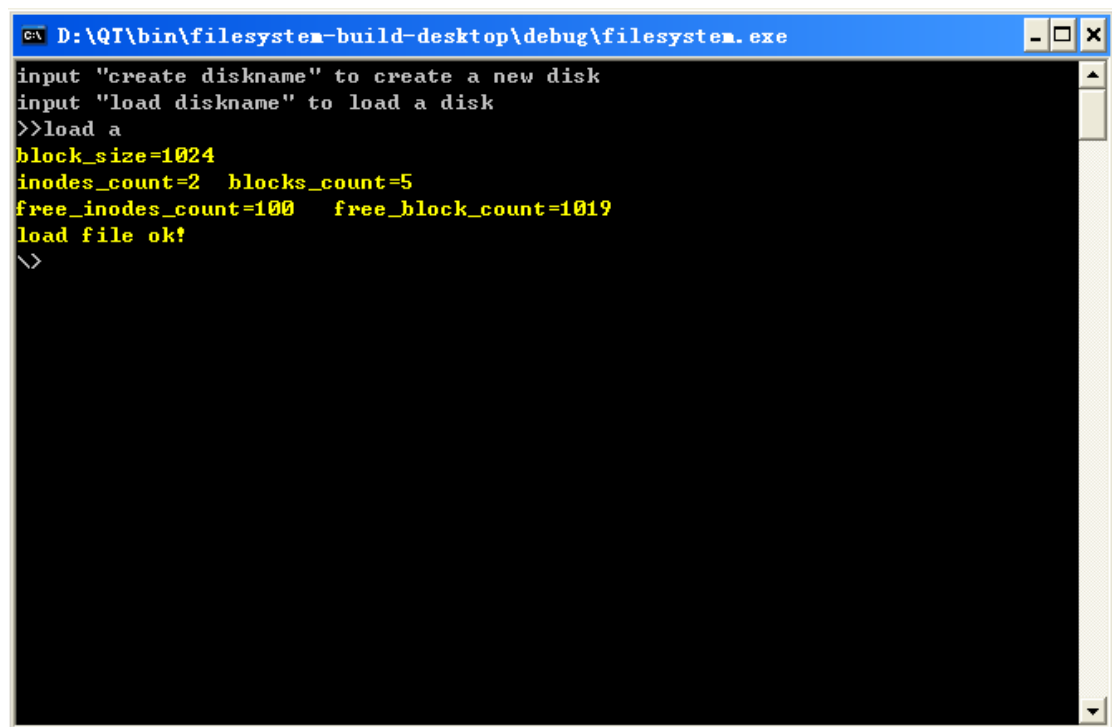
1.1. 创建一个新的 disk 并格式化：



```
C:\ D:\QT\bin\filesystem-build-desktop\debug\filesystem.exe
input "create diskname" to create a new disk
input "load diskname" to load a disk
>>create b
block_size=1024
inodes_count=1 blocks_count=3
free_inodes_count=101 free_block_count=1021
format ok!
input "create diskname" to create a new disk
input "load diskname" to load a disk
>>
```

如图表示创建一个叫 b 的 disk 并初始化。

1.2. 载入一个 disk：

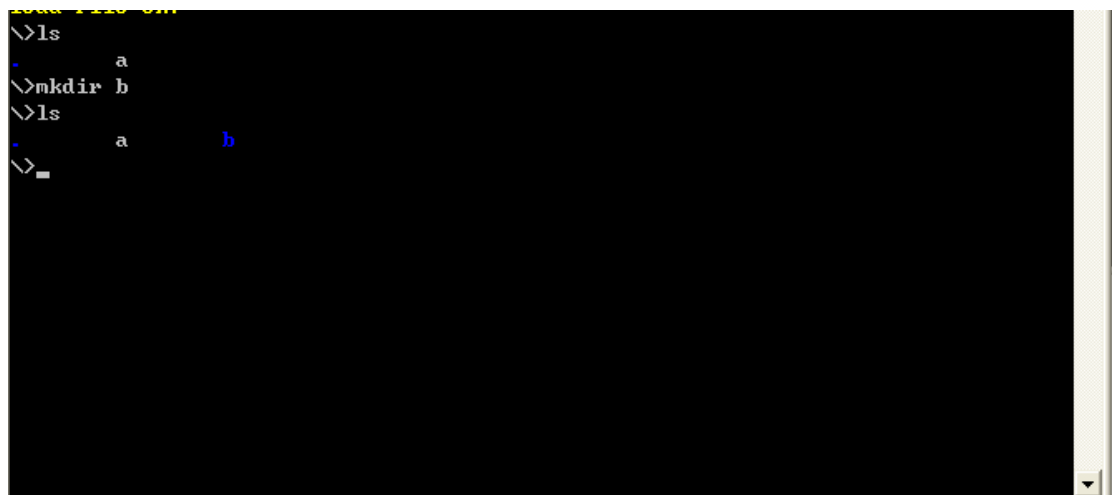


```
D:\QT\bin\filesystem-build-desktop\debug\filesystem.exe
input "create diskname" to create a new disk
input "load diskname" to load a disk
>>load a
block_size=1024
inodes_count=2 blocks_count=5
free_inodes_count=100 free_block_count=1019
load file ok!
>>
```

如果为载入一个 disk a。

2. 目录操作:

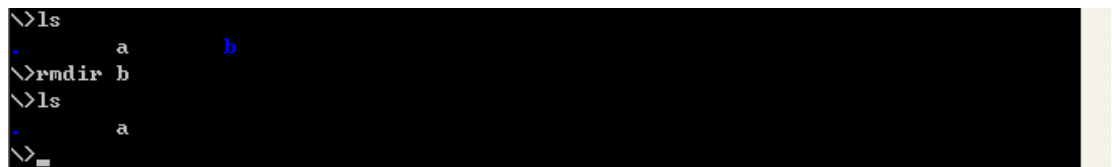
2.1. 新建目录:



```
\>ls
.      a
\>mkdir b
\>ls
.      a      b
\>
```

如图通过 mkdir b 新建了目录 b。

2.2. 删除目录:



```
\>ls
.      a      b
\>rmdir b
\>ls
.      a
\>
```

如图所示，原本还存在的 b 文件夹经 rmdir b 之后被删除了。

2.3. 进入目录:

```
\>ls
.      a      b
\>cd b
\b\>ls
.      ..
\b\>cd ..
\>ls
.      a      b
\>
```

如图所示，通过 `cd b` 指令，进入到了 `b` 目录，路径显示也有所改变。通过 `cd ..` 便返回到了上一级目录。

3. 文件操作：

3.1. 创建文件：

```
\>ls
.      a      b
\>touch c
input the content
alex shi
\>ls
.      a      b      c
\>cat c
alex shi
\>
```

如图所示，可见通过 `touch c` 新建了 `c` 文件，并可以输入内容，并通过 `cat c` 来查看内容。

3.2. 修改文件内容：

```
\>cat c
alex shi
\>write c
input the content
09cs
\>cat c
09cs
\>
```

如图所示，我们通过 `write c` 我们修改了 `c` 文件中的内容。

3.3. 删除文件：

```
\>ls
.      a      b      c
\>rm c
\>ls
.      a      b
\>
```

如果所示，通过 `rm c` 我们删除了原本存在的 `c` 文件。

3.4. 查看文件内容：

对于 `cat` 指令上面的文件功能中已经有所展示，这里不赘述。

3.5. 展示目录内容：

对于 `ls` 指令上面已经有展示，这里不赘述。

4. 警告

由于文件中有各种警告，这里举一个例子，如：

```
\b\b\hh\>ls
.      ..
\b\b\hh\>rmdir ..
the father and directory itself cannot be removed!
\b\b\hh\>
```

这里为试图删除“..”文件夹，即父文件夹。从而有红色字体报错。

实验中遇到的问题：

1. A: 出现可以使用 `cd` 打开文件，用 `rm` 删除目录等情况。
Q: 一开始没有设定 `isroot`, `isdir` 等变量，导致目录与文件指令混乱。
2. A: 输入 `ls` 指令，没有显示“.”与“..”文件夹
Q: 一开始我自己还设定不限时“.”与“..”文件夹，后来经过老师指导发现应该显示这两个文件夹。所以对 `list()` 函数进行了修改。

实验感想：

1. 了解了一般的 `linux` 文件系统中的数据结构。因为我的文件系统是以 `ext2` 为样本的，所以对于 `ext2` 有了更深入的了解。
2. 知道了 `superblock`，以及 `inode` 的具体结构以及作用。
3. 学会了如何在 `windows` 的 `console` 界面中修改字体颜色。