Project 5 File System

姓名：洪语晨
学号：10161511322

目标:

1、设计一个文件系统，最多可以容纳5000个文件，设备容量为250MB,单个文件最大容量为50MB,提供目录树功能，文件名最长为50个字符，每块大小1KB。
2、项目报告应该包含文件系统的详细设计信息和所有实现的源代码文件。

要求:

1、你的代码应该实现下列 API（源自 POSIX 并简化过的 FILE API）：

- int my_open(char * path)，打开一个已经存在的文件以备读或写，返回文件描述符，文件不存在则失败
- int my_creat (char *path),为只写操作打开一个新文件(文件已经存在则失败)，返回文件描述符
- int my_read (int fd,void* buf, int count)，顺序读取一个文件
- int my_write(int fd,void* buf,int count)，顺序写入一个文件
- int my_close(int fd)，关闭文件
- int my_remove(char *path) 删除文件，文件不存在则失败
- int my_rename(char * old ,char * new)文件改名(支持 move 操作)
- int my_mkdir (char *path),创建目录(只有在所有上层目录都已存在的情况下才创建成功，否则失败)
- int my_rmdir(char *path) 删除文件夹(假设文件夹已经为空)
- void my_mkfs(),检查设备是上是否已经存在文件系统，如果不存在，则创建一个。

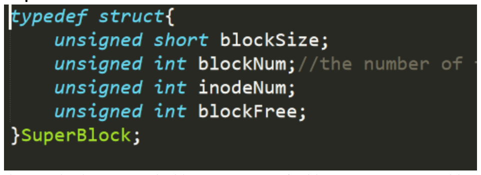
以上这些文件 API 已经在 p5.c 上定义好，但是未实现。如果代码量较大，可以考虑将 p5.c 拆分成多个小 c 文件。为了方便你的实现，已经提供 p5.h 和一个非常简单的测试程序，testp5.c. 这个程序默认情况将会做所有测试。
2、实现需要通过测试程序的测试。
3、提交实验报告，报告内容包括实现思路、设计方案，实现要点和所有的源代码。

实验说明

1. 本实验可以在 minix 下实现，也可以在 linux 下实现，推荐在 minix 下实现。
2. 本实验采用文件来模拟磁盘，采用 dd 命令创建文件，例如：dd if=/dev/zero of=simulated_device bs=1024 count=250000 将创建一个包含 250000 块文件名为 simulated_device 的文件。(这个文件可以满足测试程序中所有测试程序的空间需要)。
3. 本实验不需要自行编写块操作函数，实验预先提供了 block.c 文件，已经实现了打开块设备(dev_open),读块(read_block),写块(write_block)的功能。所以实验只需要在此基础上实现实验具体要求中的文件系统 API 即可。注意：block.c 的实现中默认模拟磁盘文件是 simulated_device,如果你创建了不同名字的模拟磁盘文件，请在 block.c 的 dev_open 函数中修改。

4. 本实验的主要结果是看 testp5.c 中测试程序的输出，所以在提交实验报告和源文件前，请确保 testp5.c 中所有的测试都已通过，或者尽可能多测试的通过。
5. 由于涉及多个文件，建议采用 shell 脚本、Makefile 脚本等方式自动完成编译

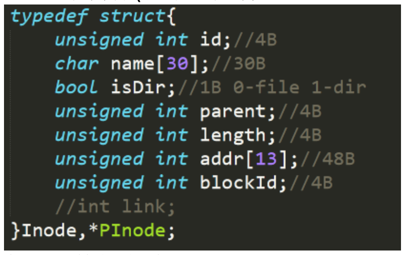
设计信息：
FileSystem.h 中结构体：
SuperBlock 结构：



```c
typedef struct{
    unsigned short blockSize;
    unsigned int blockNum;//the number of
    unsigned int inodeNum;
    unsigned int blockFree;
}SuperBlock;
```

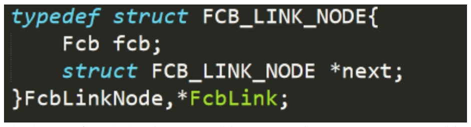
存储块大小，块数，inode 个数，以及空闲快的数目

Inode 结构体(目录和文件)：



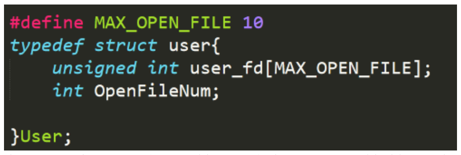
```c
typedef struct{
    unsigned int id;//4B
    char name[30];//30B
    bool isDir;//1B 0-file 1-dir
    unsigned int parent;//4B
    unsigned int length;//4B
    unsigned int addr[13];//48B
    unsigned int blockId;//4B
    //int link;
}Inode,*PInode;
```

存储文件的名称，和是否是目录，以及文件的长度，addr 中存储其包含的块号，blockId 中存储 Inode 的序号



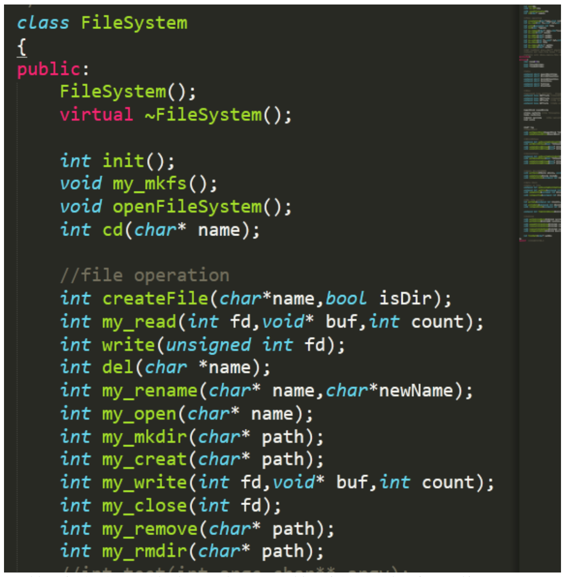
```c
typedef struct FCB_LINK_NODE{
    Fcb fcb;
    struct FCB_LINK_NODE *next;
}FcbLinkNode,*FcbLink;
```

建立一个链表存储文件，通过 Fcblink 对文件进行快速查找



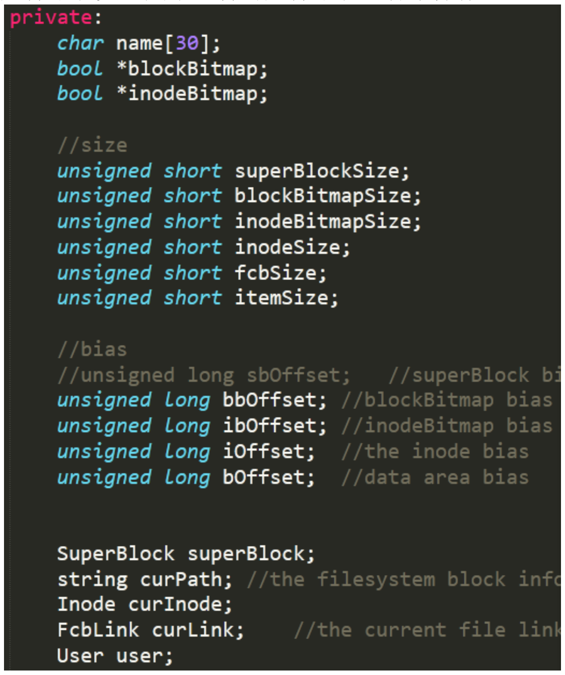
```c
#define MAX_OPEN_FILE 10
typedef struct user{
    unsigned int user_fd[MAX_OPEN_FILE];
    int OpenFileNum;

}User;
```

存储用户打开的文件，最大打开文件数目为 10



```
class FileSystem
{
public:
    FileSystem();
    virtual ~FileSystem();

    int init();
    void my_mkfs();
    void openFileSystem();
    int cd(char* name);

    //file operation
    int createFile(char*name,bool isDir);
    int my_read(int fd,void* buf,int count);
    int write(unsigned int fd);
    int del(char *name);
    int my_rename(char* name,char*newName);
    int my_open(char* name);
    int my_mkdir(char* path);
    int my_creat(char* path);
    int my_write(int fd,void* buf,int count);
    int my_close(int fd);
    int my_remove(char* path);
    int my_rmdir(char* path);
```

文件系统类，其中包含对文件系统的各个操作

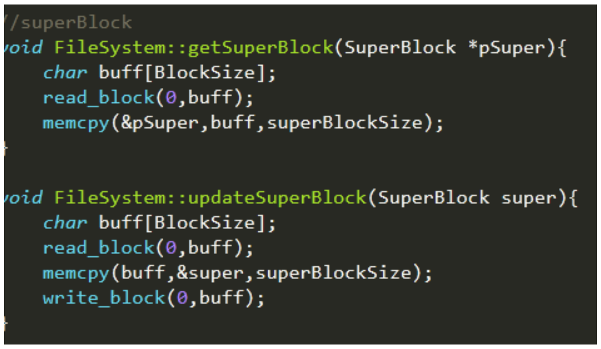


```
private:
    char name[30];
    bool *blockBitmap;
    bool *inodeBitmap;

    //size
    unsigned short superBlockSize;
    unsigned short blockBitmapSize;
    unsigned short inodeBitmapSize;
    unsigned short inodeSize;
    unsigned short fcbSize;
    unsigned short itemSize;

    //bias
    //unsigned long sbOffset;   //superBlock bi
    unsigned long bbOffset; //blockBitmap bias
    unsigned long ibOffset; //inodeBitmap bias
    unsigned long iOffset;  //the inode bias
    unsigned long bOffset;  //data area bias


    SuperBlock superBlock;
    string curPath; //the filesystem block info
    Inode curInode;
    FcbLink curLink;    //the current file link
    User user;
```

磁盘存储结构：

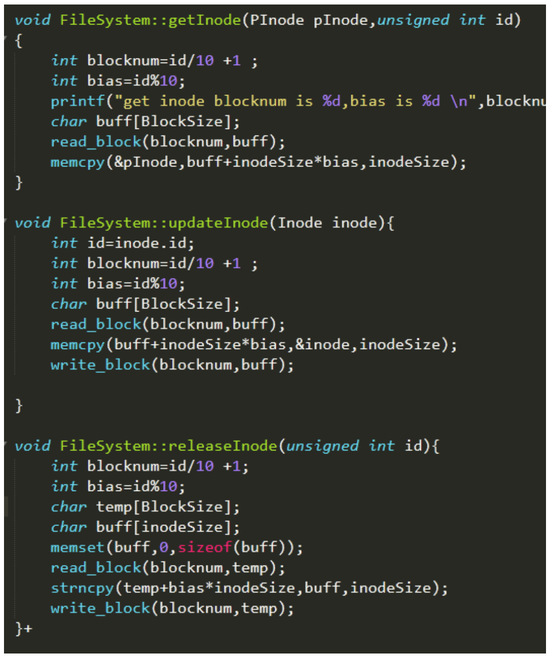
Superblock	Inodebitmap	Blockbitmap	Inode	item

对 superblock 的存储如下：



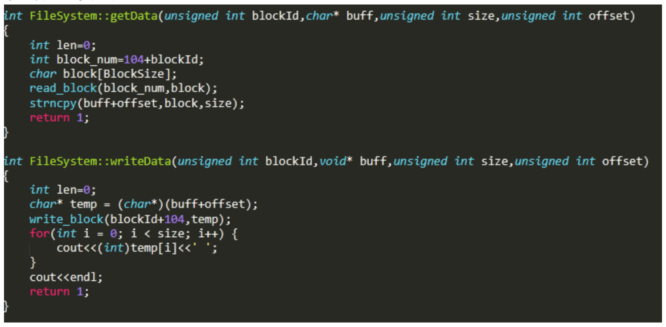
```
/superBlock
oid FileSystem::getSuperBlock(SuperBlock *pSuper){
    char buff[BlockSize];
    read_block(0,buff);
    memcpy(&pSuper,buff,superBlockSize);
}

oid FileSystem::updateSuperBlock(SuperBlock super){
    char buff[BlockSize];
    read_block(0,buff);
    memcpy(buff,&super,superBlockSize);
    write_block(0,buff);
}
```

需要获得新的 inode 或者块时，通过线性搜索找到空闲块
对 inode 的操作：



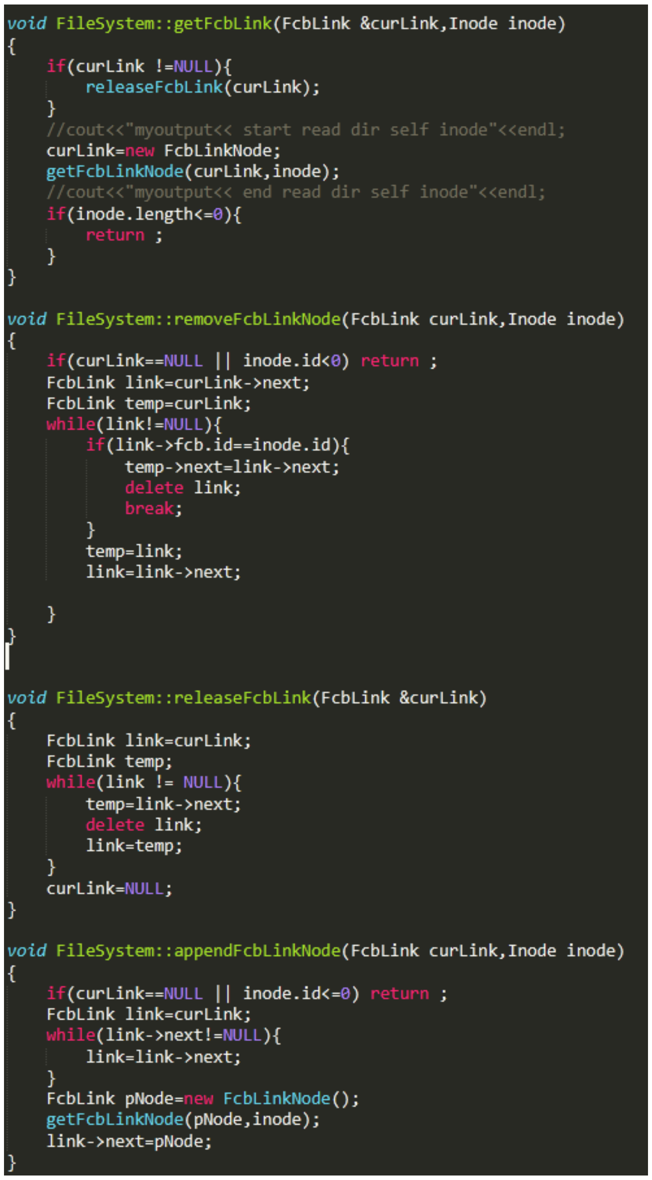
```
void FileSystem::getInode(PInode pInode,unsigned int id)
{
    int blocknum=id/10 +1 ;
    int bias=id%10;
    printf("get inode blocknum is %d,bias is %d \n",blocknu
    char buff[BlockSize];
    read_block(blocknum,buff);
    memcpy(&pInode,buff+inodeSize*bias,inodeSize);
}

void FileSystem::updateInode(Inode inode){
    int id=inode.id;
    int blocknum=id/10 +1 ;
    int bias=id%10;
    char buff[BlockSize];
    read_block(blocknum,buff);
    memcpy(buff+inodeSize*bias,&inode,inodeSize);
    write_block(blocknum,buff);

}

void FileSystem::releaseInode(unsigned int id){
    int blocknum=id/10 +1;
    int bias=id%10;
    char temp[BlockSize];
    char buff[inodeSize];
    memset(buff,0,sizeof(buff));
    read_block(blocknum,temp);
    strncpy(temp+bias*inodeSize,buff,inodeSize);
    write_block(blocknum,temp);
}+
```

数据的读写



```
int FileSystem::getData(unsigned int blockId,char* buff,unsigned int size,unsigned int offset)
{
    int len=0;
    int block_num=104+blockId;
    char block[BlockSize];
    read_block(block_num,block);
    strncpy(buff+offset,block,size);
    return 1;
}

int FileSystem::writeData(unsigned int blockId,void* buff,unsigned int size,unsigned int offset)
{
    int len=0;
    char* temp = (char*)(buff+offset);
    write_block(blockId+104,temp);
    for(int i = 0; i < size; i++) {
        cout<<(int)temp[i]<<' ';
    }
    cout<<endl;
    return 1;
}
```

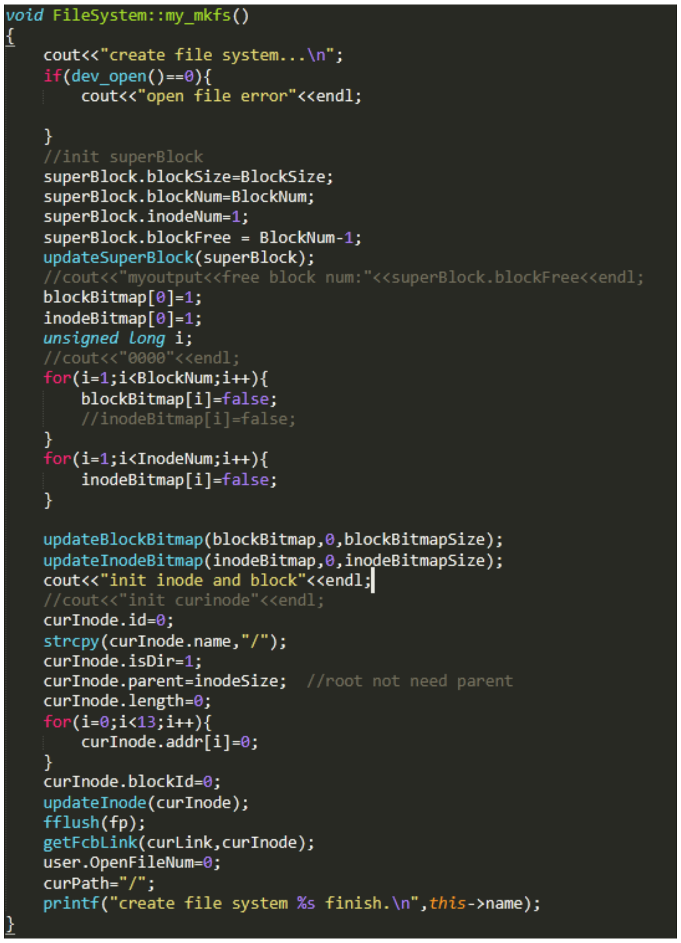
Fcblink：对节点的添加删除



```cpp
void FileSystem::getFcbLink(FcbLink &curLink,Inode inode)
{
    if(curLink !=NULL){
        releaseFcbLink(curLink);
    }
    //cout<<"myoutput<< start read dir self inode"<<endl;
    curLink=new FcbLinkNode;
    getFcbLinkNode(curLink,inode);
    //cout<<"myoutput<< end read dir self inode"<<endl;
    if(inode.length<=0){
        return ;
    }
}

void FileSystem::removeFcbLinkNode(FcbLink curLink,Inode inode)
{
    if(curLink==NULL || inode.id<0) return ;
    FcbLink link=curLink->next;
    FcbLink temp=curLink;
    while(link!=NULL){
        if(link->fcb.id==inode.id){
            temp->next=link->next;
            delete link;
            break;
        }
        temp=link;
        link=link->next;

    }
}

void FileSystem::releaseFcbLink(FcbLink &curLink)
{
    FcbLink link=curLink;
    FcbLink temp;
    while(link != NULL){
        temp=link->next;
        delete link;
        link=temp;
    }
    curLink=NULL;
}

void FileSystem::appendFcbLinkNode(FcbLink curLink,Inode inode)
{
    if(curLink==NULL || inode.id<=0) return ;
    FcbLink link=curLink;
    while(link->next!=NULL){
        link=link->next;
    }
    FcbLink pNode=new FcbLinkNode();
    getFcbLinkNode(pNode,inode);
    link->next=pNode;
}
```

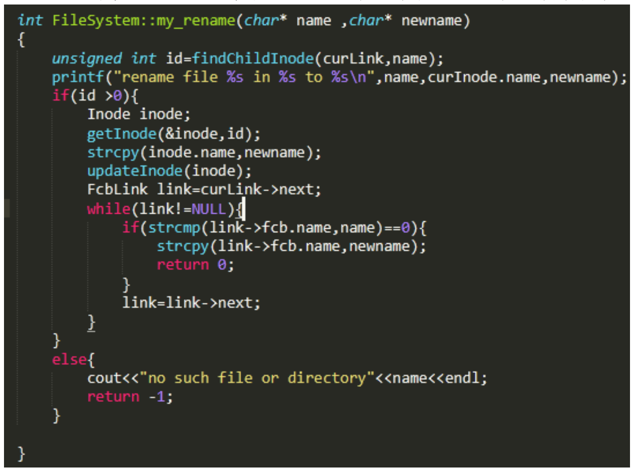
文件系统的创建

打开块设备，更新 superblock 的内容，将第一个块和 inode 节点标记为分配，建立根节点，更新得到 fcblink，更新 user 结构体



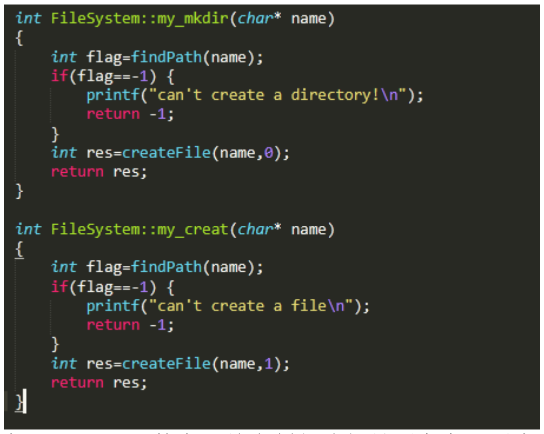
```
void FileSystem::my_mkfs()
{
    cout<<"create file system...\n";
    if(dev_open()==0){
        cout<<"open file error"<<endl;

    }
    //init superBlock
    superBlock.blockSize=BlockSize;
    superBlock.blockNum=BlockNum;
    superBlock.inodeNum=1;
    superBlock.blockFree = BlockNum-1;
    updateSuperBlock(superBlock);
    //cout<<"myoutput<<free block num:"<<superBlock.blockFree<<endl;
    blockBitmap[0]=1;
    inodeBitmap[0]=1;
    unsigned long i;
    //cout<<"0000"<<endl;
    for(i=1;i<BlockNum;i++){
        blockBitmap[i]=false;
        //inodeBitmap[i]=false;
    }
    for(i=1;i<InodeNum;i++){
        inodeBitmap[i]=false;
    }

    updateBlockBitmap(blockBitmap,0,blockBitmapSize);
    updateInodeBitmap(inodeBitmap,0,inodeBitmapSize);
    cout<<"init inode and block"<<endl;
    //cout<<"init curinode"<<endl;
    curInode.id=0;
    strcpy(curInode.name,"/");
    curInode.isDir=1;
    curInode.parent=inodeSize;  //root not need parent
    curInode.length=0;
    for(i=0;i<13;i++){
        curInode.addr[i]=0;
    }
    curInode.blockId=0;
    updateInode(curInode);
    fflush(fp);
    getFcbLink(curLink,curInode);
    user.OpenFileNum=0;
    curPath="/";
    printf("create file system %s finish.\n",this->name);
}
```

rename 函数，通过名字在 fcblink 中查找 inode 号，若找到，则对 inode 进行更改，并且同时更新 fcblink



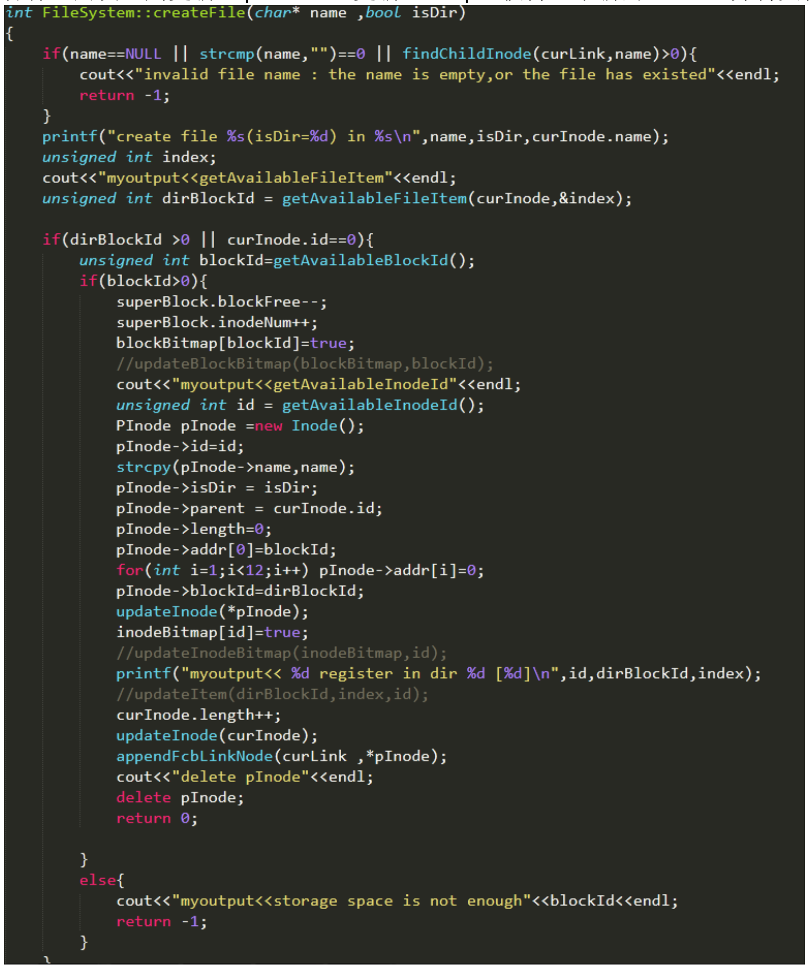
```
int FileSystem::my_rename(char* name ,char* newname)
{
    unsigned int id=findChildInode(curLink,name);
    printf("rename file %s in %s to %s\n",name,curInode.name,newname);
    if(id >0){
        Inode inode;
        getInode(&inode,id);
        strcpy(inode.name,newname);
        updateInode(inode);
        FcbLink link=curLink->next;
        while(link!=NULL){
            if(strcmp(link->fcb.name,name)==0){
                strcpy(link->fcb.name,newname);
                return 0;
            }
            link=link->next;
        }
    }
    else{
        cout<<"no such file or directory"<<name<<endl;
        return -1;
    }

}
```

文件和目录的创建，因为目录是特殊的文件，在创建时调用 createfile 函数，仅传参不同



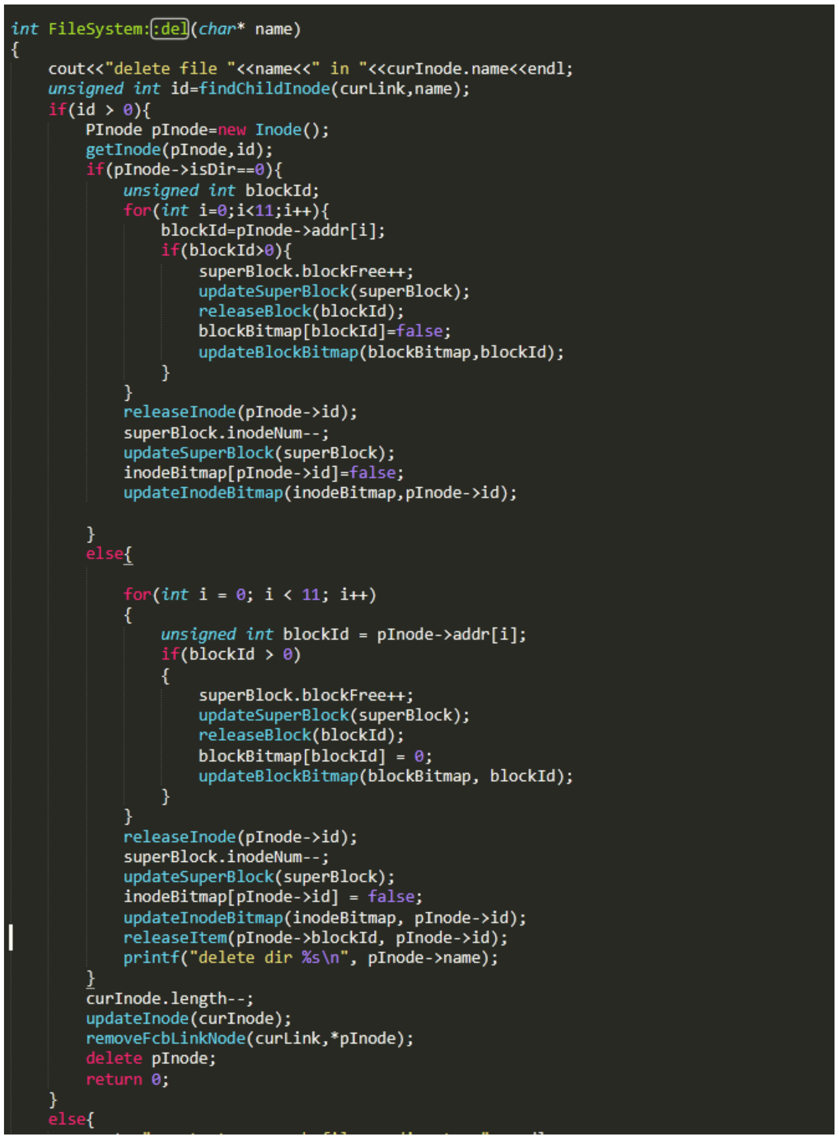
```
int FileSystem::my_mkdir(char* name)
{
    int flag=findPath(name);
    if(flag==-1) {
        printf("can't create a directory!\n");
        return -1;
    }
    int res=createFile(name,0);
    return res;
}

int FileSystem::my_creat(char* name)
{
    int flag=findPath(name);
    if(flag==-1) {
        printf("can't create a file\n");
        return -1;
    }
    int res=createFile(name,1);
    return res;
}
```

在 createfile 函数中，首先判断路径是否存在，再查看是否有空闲块
若有空闲块，则更新 superblock，更新 bitmap，获得一个新的 inode，并将文件信息填入，最后将其写回磁盘



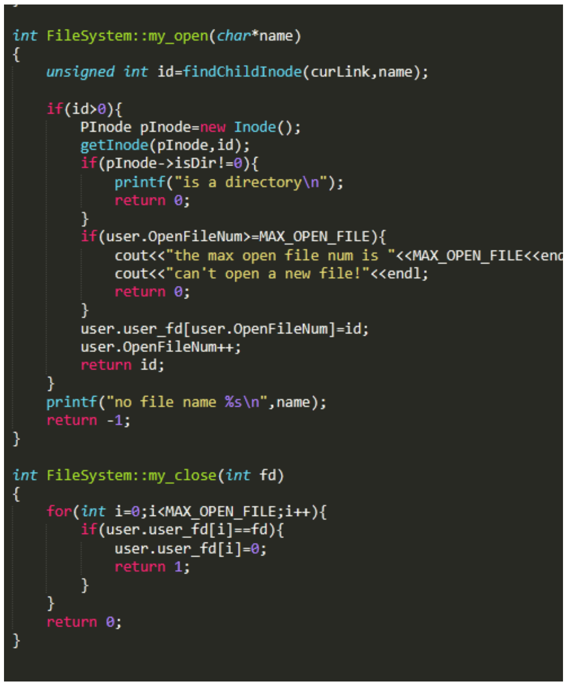
```
int FileSystem::createFile(char* name ,bool isDir)
{
    if(name==NULL || strcmp(name,"")==0 || findChildInode(curLink,name)>0){
        cout<<"invalid file name : the name is empty,or the file has existed"<<endl;
        return -1;
    }
    printf("create file %s(isDir=%d) in %s\n",name,isDir,curInode.name);
    unsigned int index;
    cout<<"myoutput<<getAvailableFileItem"<<endl;
    unsigned int dirBlockId = getAvailableFileItem(curInode,&index);

    if(dirBlockId >0 || curInode.id==0){
        unsigned int blockId=getAvailableBlockId();
        if(blockId>0){
            superBlock.blockFree--;
            superBlock.inodeNum++;
            blockBitmap[blockId]=true;
            //updateBlockBitmap(blockBitmap,blockId);
            cout<<"myoutput<<getAvailableInodeId"<<endl;
            unsigned int id = getAvailableInodeId();
            PInode pInode =new Inode();
            pInode->id=id;
            strcpy(pInode->name,name);
            pInode->isDir = isDir;
            pInode->parent = curInode.id;
            pInode->length=0;
            pInode->addr[0]=blockId;
            for(int i=1;i<12;i++) pInode->addr[i]=0;
            pInode->blockId=dirBlockId;
            updateInode(*pInode);
            inodeBitmap[id]=true;
            //updateInodeBitmap(inodeBitmap,id);
            printf("myoutput<< %d register in dir %d [%d]\n",id,dirBlockId,index);
            //updateItem(dirBlockId,index,id);
            curInode.length++;
            updateInode(curInode);
            appendFcbLinkNode(curLink ,*pInode);
            cout<<"delete pInode"<<endl;
            delete pInode;
            return 0;

        }
        else{
            cout<<"myoutput<<storage space is not enough"<<blockId<<endl;
            return -1;
        }
```

文件和目录的删除，调用 del 函数，完成对 inode 和块的释放，更新 superblock，更新 bitmap



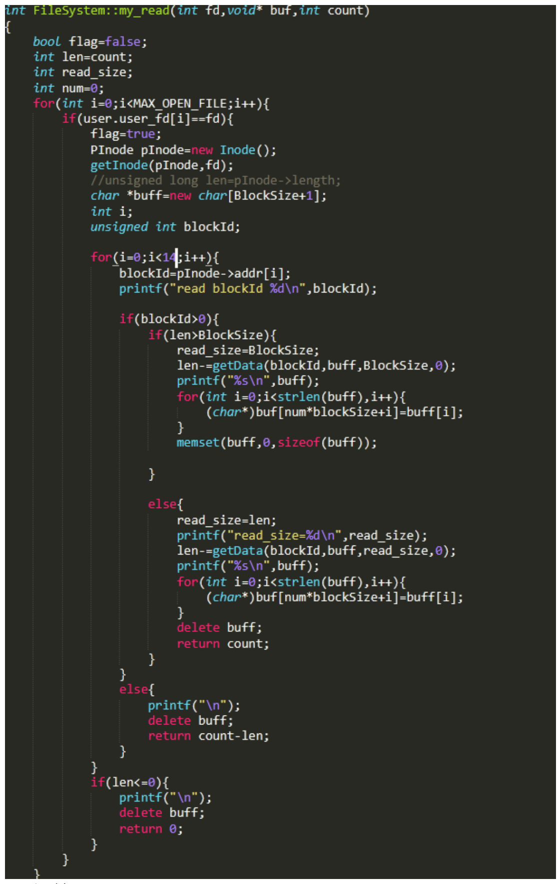
```
int FileSystem::del(char* name)
{
    cout<<"delete file "<<name<<" in "<<curInode.name<<endl;
    unsigned int id=findChildInode(curLink,name);
    if(id > 0){
        PInode pInode=new Inode();
        getInode(pInode,id);
        if(pInode->isDir==0){
            unsigned int blockId;
            for(int i=0;i<11;i++){
                blockId=pInode->addr[i];
                if(blockId>0){
                    superBlock.blockFree++;
                    updateSuperBlock(superBlock);
                    releaseBlock(blockId);
                    blockBitmap[blockId]=false;
                    updateBlockBitmap(blockBitmap,blockId);
                }
            }
            releaseInode(pInode->id);
            superBlock.inodeNum--;
            updateSuperBlock(superBlock);
            inodeBitmap[pInode->id]=false;
            updateInodeBitmap(inodeBitmap,pInode->id);

        }
        else{

            for(int i = 0; i < 11; i++)
            {
                unsigned int blockId = pInode->addr[i];
                if(blockId > 0)
                {
                    superBlock.blockFree++;
                    updateSuperBlock(superBlock);
                    releaseBlock(blockId);
                    blockBitmap[blockId] = 0;
                    updateBlockBitmap(blockBitmap, blockId);
                }
            }
            releaseInode(pInode->id);
            superBlock.inodeNum--;
            updateSuperBlock(superBlock);
            inodeBitmap[pInode->id] = false;
            updateInodeBitmap(inodeBitmap, pInode->id);
            releaseItem(pInode->blockId, pInode->id);
            printf("delete dir %s\n", pInode->name);
        }
        curInode.length--;
        updateInode(curInode);
        removeFcbLinkNode(curLink,*pInode);
        delete pInode;
        return 0;
    }
    else{
```

文件的打开和关闭：
通过在 fcblink 中进行查找，将 inode 号返回，并且记录在用户打开文件的数组当中



```
int FileSystem::my_open(char*name)
{
    unsigned int id=findChildInode(curLink,name);

    if(id>0){
        PInode pInode=new Inode();
        getInode(pInode,id);
        if(pInode->isDir!=0){
            printf("is a directory\n");
            return 0;
        }
        if(user.OpenFileNum>=MAX_OPEN_FILE){
            cout<<"the max open file num is "<<MAX_OPEN_FILE<<enc
            cout<<"can't open a new file!"<<endl;
            return 0;
        }
        user.user_fd[user.OpenFileNum]=id;
        user.OpenFileNum++;
        return id;
    }
    printf("no file name %s\n",name);
    return -1;
}

int FileSystem::my_close(int fd)
{
    for(int i=0;i<MAX_OPEN_FILE;i++){
        if(user.user_fd[i]==fd){
            user.user_fd[i]=0;
            return 1;
        }
    }
    return 0;
}
```

文件的读写：查找 fd 是否在用户打开文件中，然后通过 fd 找到对应 inode，对 inode 中记录的文件块号进行读写
读文件，依次遍历索引块



```
int FileSystem::my_read(int fd,void* buf,int count)
{
    bool flag=false;
    int len=count;
    int read_size;
    int num=0;
    for(int i=0;i<MAX_OPEN_FILE;i++){
        if(user.user_fd[i]==fd){
            flag=true;
            PInode pInode=new Inode();
            getInode(pInode,fd);
            //unsigned long len=pInode->length;
            char *buff=new char[BlockSize+1];
            int i;
            unsigned int blockId;

            for(i=0;i<14;i++){
                blockId=pInode->addr[i];
                printf("read blockId %d\n",blockId);

                if(blockId>0){
                    if(len>BlockSize){
                        read_size=BlockSize;
                        len-=getData(blockId,buff,BlockSize,0);
                        printf("%s\n",buff);
                        for(int i=0;i<strlen(buff),i++){
                            (char*)buf[num*blockSize+i]=buff[i];
                        }
                        memset(buff,0,sizeof(buff));

                    }

                    else{
                        read_size=len;
                        printf("read_size=%d\n",read_size);
                        len-=getData(blockId,buff,read_size,0);
                        printf("%s\n",buff);
                        for(int i=0;i<strlen(buff),i++){
                            (char*)buf[num*blockSize+i]=buff[i];
                        }
                        delete buff;
                        return count;
                    }
                }
                else{
                    printf("\n");
                    delete buff;
                    return count-len;
                }
            }
            if(len<=0){
                printf("\n");
                delete buff;
                return 0;
            }
        }
    }
```

写文件

```
int FileSystem::my_write(int fd,void* buff,int count)
{
        unsigned long len=0;//the total write size
        int num=count;
        int write_size;
        PInode pInode = new Inode();
        getInode(pInode,fd);
        unsigned int blockId;
        for(int i=0;i<12;i++){
                if(num > BlockSize){
                        write_size=BlockSize;
                        blockId=pInode->addr[i];
                        if(blockId>0){
                                writeData(blockId,buff,write_size,0);
                                num-=BlockSize;
                                len+=BlockSize;

                        }
                        else{
                                blockId=getAvailableBlockId();
                                if(blockId>0){
                                        superBlock.blockFree--;
                                        updateSuperBlock(superBlock);
                                        blockBitmap[blockId]=1;
                                        pInode->length=BlockSize;
```

```
                    updateInode(*pInode);
                    writeData(blockId,buff,write_size,0);
                    num-=BlockSize;
                    len+=BlockSize;
                }
            }
        }
        else{
            write_size=num;
            blockId=pInode->addr[i];
            if(blockId>0){
                writeData(blockId,buff,write_size,0);
                len+=num;
                printf("write_size=%d\n",write_size);
                pInode->length=len;
                updateInode(*pInode);
                //delete buff;
                return len;
            }
            else{
                blockId=getAvailableBlockId();
                if(blockId>0){
                    superBlock.blockFree--;
                    updateSuperBlock(superBlock);
                    blockBitmap[blockId]=1;
                    //updateInode(*pInode);
                    printf("write_size is %d\n",write_size);
                    writeData(blockId,buff,write_size,0);
                    len+=num;
                    pInode->length=len;
                    updateInode(*pInode);
                    delete buff;
                    return len;
                }
            }

        }
    }
}
```